Survey on Attribute and Predicate Based Encryption and Improvement of the Existing Security Proof for Hierarchical Hidden Vector Encryption Scheme

Report Submitted

by

Avik Chakraborti

M.Tech (Computer Science)

Roll No. CS0905

Indian Statistical Institute

AS A FULFILLMENT OF THE DISSERTATION

Under The Guidance

of

Prof. Rana Barua

Indian Statistical Institute

203 B.T. Road, Kolkata : 700108

July 16, 2011

Indian Statistical Institute

203, B.T. Road. Kolkata : 700108

CERTIFICATE

This is to certify that the thesis entitled "**Survey on Attribute and Predicate Based Encryption and Improvement of the Existing Security Proof for Hierarchical Hidden Vector Encryption Scheme** " is submitted in the fulfilment of the degree of M.Tech. in Computer Science at Indian Statistical Institute, kolkata. It is fully adequate, in scope and quality as a dissertation for the required degree.

The thesis is faithfully record of bonafide research work carried out by Avik Chakraborti under my supervision and guidance. It is further certified that no part of this thesis has been submitted to any other university or Institute for the award of any degree or diploma.

Prof. Rana Barua
(Supervisor)

Countersigned
(External Examinar)
Date:

Acknowledgement

I take this opportunity to express my sincere gratitude to the supervisor of this dissertation work, Prof. Rana Barua. His instructions guided me towards the learning process and the analysis and also helped me in all other aspects required for thesis. He has always been really motivating and inspiring for me throughout my dissertation. He has also given me the full freedom to think and explore new ideas and implementing those ideas.

I also take the opportunity to specially thank Tapasda, for guiding me towards the discovery of problem and giving his continuous suggestions, motivation, encouragement and helping me for the preparation of this manuscript.

I would also like to thank all my teachers who have taught me throughout the semesters in M.Tech course, specially Kishan Sir, Sumit da and Ashish da for teaching and giving me the basic knowledge of Cryptology, which gave me the confidence to work in this field. I also thank my family members and friends for their endless support.

Place: Kolkata

Date: Avik Chakraborti

Abstract

Predicate encryption is a new paradigm generalizing, among other things, identity-based encryption. In a predicate encryption scheme, secret keys correspond to predicates and ciphertexts are associated with attributes; the secret key $\mathbf{SK_f}$ corresponding to a predicate f can be used to decrypt a ciphertext associated with attribute I if and only if $f(I) = 1$. Constructions of such schemes are currently known for relatively few classes of predicates.In 2008 Jonathon Katz et. al [14]. proposed Predicate Encryption scheme Supporting Disjunctions, Polynomial Equations, and Inner Products in [5]. In 2010 Angelo De Caro et. al [7]. proposed three secure Hidden Vector Encryption Scheme and demonstrated their security under certain hard Assumptions in [7].

In this work we have pointed out some redundancy in the security proof of the Hierarchical Hidden Vector Encryption given in [7] and have given a new security proof for the scheme that removes the flaw. We have also done a brief survey on some important schemes of Attribute Based Encryption and Predicate Based Encryption.

Chapter 1

Introduction

In a traditional Public Key Cryptosystem (PKC), the association between a user's identity and his public key is obtained through a digital certificate issued by a Certifying Authority (CA). The CA checks the credentials of a user before issuing a certificate to him. If Alice wants to send a signed message to Bob, first she obtains a digital certificate for her public key from a CA. Alice then signs a message using her private key and sends the signed message along with her certificate to Bob. Bob first verifies the validity of the certificate by checking the certificate revocation list published by the CA, then he verifies the signature using public key in the certificate. If many CAs are involved between Alice and Bob the entire certificate path has to be verified. Hence, the process of certificate management requires high computational and storage efforts.

To simplify the certificate management process, Shamir [18] introduced the concept of Identity-based cryptosystem in 1984. In such cryptosystems the public key of a user is derived from his identity information and his private key is generated by a trusted third party called Private Key Generator (PKG). The advantage of ID-based cryptosystems is that it simplifies the key management process which is a heavy burden in the traditional certificate based cryptosystems. In these cryptosystems Alice can send an encrypted message to Bob by using Bob's identity information even before Bob obtains his private key from the PKG. In the case of signature Bob can verify Alice's signature just by using her identity information. In general, an identity based cryptosystem has the following properties:

1) user's public key is his identity (or derived from identity).

2) no requirement of public key directories

3) message encryption and signature verification processes require only receivers' and sign ers' identity respectively along with some system parameters.

These properties make ID-based cryptosystems advantageous over the traditional PKCs, as key distribution is far simplified. It needs a directory only for authenticated public system parameters of the PKG, which is clearly less burdensome than maintaining a public key directory for total users. This setup is also much more secure than traditional PKC because Adversary can always forge the Public Key Directory by updating some entry with his public key and can easily get the secret message (The message is encrypted with Adversary's public key) and decrypt it with his own key. However, PKC also suffer from an inherent drawback of key escrow i.e. PKG knows the users' private keys. They also require a secure channel for key issuance between PKG and user. The ID-based cryptosystems require the users to authenticate themselves to their PKG in the same way as they would authenticate themselves to a CA in traditional PKC.

Identity-Based Encryption (IBE) allows for a sender to encrypt a message to an identity without access to a public key certificate. The ability to do public key encryption without certificates has many practical applications. For example, a user can send an encrypted mail to a recipient, e.g. **avikchkrbrti@yahoo.co.in**, without the requiring either the existence of a Public-Key Infrastructure or that the recipient be on-line at the time of creation.

One common feature of all previous Identity-Based Encryption systems is that they view identities as a string of characters. But Amit Sahai and Brent Waters proposed a new IBE scheme at EuroCrypt, 2005 called Fuzzy Identity Based Encryption [16] where identities of a user as a set of descriptive attributes. In a Fuzzy Identity-Based Encryption scheme, a user with the secret key for the identity ω is able to decrypt a ciphertext encrypted with the public key ω' if and only if ω and ω' are within a certain distance of each other as judged by some metric. Therefore, our system allows for a certain amount of error-tolerance in the identities.

Fuzzy-IBE gives rise to two interesting new applications. The first is an Identity-Based En- cryption system that uses *biometric identities*. That is we can view a user's biometric, for example an iris scan, as that user's identity described by several attributes and then encrypt to

the user using their biometric identity. Secondly, Fuzzy IBE can be used for an application that we call *attribute-based encryption*. In this application a party will wish to encrypt a document to all users that have a certain set of attributes. For example, in an Institute, the director might want to encrypt a document to all the "Professors" who are in the "Placement Committee" in the department of "Computer Science" in his institute. In this case he would encrypt to the identity $\{\text{"}\mathbf{Professor}\text{"}, \text{"}\mathbf{PlacementCommittee}\text{"}, \text{"}\mathbf{ComputerScience}\text{"}\}$. Any user who has an identity that contains all of these attributes could decrypt the document.

Here, we describe a much richer type of attribute-based encryption cryptosystem due to Goyal et al. and demonstrate its applications. In this system each ciphertext is labeled by the encryptor with a set of descriptive attributes. Each private key is associated with an access structure that specifies which type of ciphertexts the key can decrypt.This scheme is called **Key-Policy Attribute Based Encryption (KP ABE)** [11] since the access structure is specified in the private key, while the ciphertexts are simply labeled with a set of descriptive attributes. This setting is also reminiscent of secret sharing schemes . Using known techniques one can build a secret sharing scheme that specifies that a set of parties must cooperate in order to reconstruct a secret. For example, one can specify a tree access structure where the interior nodes consist of $\mathbf{AND}$ and $\mathbf{OR}$ gates and the leaves consist of different parties. Any set of parties that satisfy the tree can reconstruct the secret.

Another possibility is to have the reverse situation: user keys are associated with sets of attributes, whereas ciphertexts are associated with policies. Such systems are called **Ciphertext Policy Attribute Based Encryption (CP ABE)** [17] systems. The construction of Sahai and Waters [SW05] was most naturally considered in this framework. CP-ABE systems that allow for complex policies (like those con- sidered here) have a number of applications. An important example is a kind of *sophisticated Broadcast Encryption*, where users are described by various attributes. Then, one could create a ciphertext that can be opened only if the attributes of a user match a policy. For example, in a military setting, one could broadcast a message that is meant to be read only by users who have a rank of Lieutenant or higher, and who were deployed in South Korea in the year 2005.

Further modification on CP-ABE systems is presented by Goyal et al. [12]. Previous CP-ABE systems could either support only very limited access structures or had a proof of security only in the generic group model. This construction can support access structures which can be

represented by a bounded size access tree with threshold gates as its nodes. The bound on the size of the access trees is chosen at the time of the system setup represented by a tuple *(d, num)* where d represents the maximum depth of the access tree and num represents the maximum number of children each non-leaf node of the tree might have.

In 2007 Ling Chuang et al [8]. observed that attributes can be arranged into logical hierarchies, which in turn can be used to improve the efficiency of the scheme. Essentially, a hierarchy allows us to use fewer group elements to represent all attributes in the system, thereby reducing the ciphertext size, the number of exponentiations in encryption and the number of pairings in decryption.

Clearly Their are two types of secrecy *Payload Hiding* and *Attribute Hiding*. Roughly speaking attribute hiding requires that the ciphertext conceal the attribute as well as the plaintext, while payload hiding only requires that the ciphertext conceals the plaintext The current constructions of Attribute Based Encryptions do not hide the set of attributes under which the data is encrypted. Hence they exhibits Payload hiding not attribute hiding. However, if it were possible to hide the attributes, then viewing attributes as keywords in such a system would lead to the first general keyword-based search on encrypted data [1]. A search query could potentially be any monotone boolean formula of any number of keywords. Some progress towards this goal has been made in [2] [6], in a notion generalizing ABE called Functional Encryption (or Predicate Encryption). In Functional Encryption, secret keys correspond to functions that can be used to decrypt a ciphertext (with attributes) if and only if the attributes satisfy the function. The works [2] [6] construct schemes that answer this problem for certain classes of functions.

In 2007 Jonathon Katz et al. [14] proposed Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Product for predicates corresponding to the evaluation of inner products over $\mathbb{Z}_N$ (for some large integer N). This, in turn, enables constructions in which predicates correspond to the evaluation of disjunctions, polynomials, $\mathbf{CNF/DNF}$ formulae, or threshold predicates. In 2010 Angelo De Caro et al. proposed fully secure implementations for Conjunctions (also called Hidden Vector Encryption in the literature), Disjunctions and $\mathbf{k-CNF/DNF}$ predicates that guarantee the security of the plaintext and of the attribute.Their constructions for Disjunctions and Conjunctions are linear in the number of variables in contrast to the previous fully secure constructions for Disjunction required time

exponential in the number of variables while for Conjunctions the best previous construction was quadratic in the number of variables.

The thesis is organized as follows :

- **Chapter2**: This chapter discusses the existing important works in Attribute Based Encryption. Specifically, the existing Ciphertext Policy Attribute Based Encryption systems are discussed stating their construction and security achieved .
- **Chapter3**: This chapter discusses the existing important works in Predicate Based Encryption by stating their construction and security achieved .
- **Chapter4**: This chapter discusses the redundancy of existing security proof for Hierarchical Hidden Vector Encryption(HHVE) and our improved security proof for that scheme.
- **Chapter5**: This chapter gives a brief conclusion to the thesis .

Chapter 2

A Brief Survey on Attribute Based Encryption

In this chapter we are giving a brief survey on Attribute Based Encryption.

The main properties of Attribute Based Encryption are

$\mathbf{1}$) Which users can decrypt a ciphertext will be decided by the attributes and policies associated with the message and the user.
$\mathbf{2}$) A central authority will create secret keys for the users based on attributes/policies foreach user.
$\mathbf{3}$) Ciphertexts can be created (by anyone) by incorporating attributes/policies.

In the next sections we are describing some important schemes in Attribute Based Encryption with their properties .

2.1 Some Important Definitions

In this section we are giving some important definitions

Definition 1 (Access Structure) *Let $\{P_1, P_2, ..., P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1,P_2,...,P_n\}}$ is monotone if $\forall$ B, C : if B $\in \mathbb{A}$ and B $\subseteq$ C then C $\in \mathbb{A}$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, P_2, ..., P_n\}$, i.e., $\mathbb{A} \subseteq 2^{P1,P2,...,Pn} \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called the* authorized sets, *and the sets not in $\mathbb{A}$ are called the* unauthorized sets.

Definition 2 (Bilinear group) *Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be two multiplicative cyclic groups of prime order p. Let g be a generator of G_0 and e be a bilinear map, e : $\mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$. The bilinear map e has the following properties:*

1. Bilinearity: for all u, v $\in \mathbb{G}_0$ and a, b $\in \mathbb{Z}_p$, we have e(u^b , v^b) = $e(u, v)^{ab}$.
2. Non-degeneracy: e(g, g) $\neq$ 1.

We say that $\mathbb{G}_0$ is a bilinear group if the group operation in $\mathbb{G}_0$ and the bilinear map e *: $\mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ are both efficiently computable. Notice that the map e is symmetric since e(g^a , g^b) = $e(g, g)^{ab}$ = e(g^b , g^a).*

2.2 Fuzzy Identity-Based Encryption

2.2.1 Construction

Recall that identities are viewed as sets of attributes and let d represent the error-tolerance in terms of minimal set overlap. When an authority is creating a private key for a user, he will associate a random $d-1$ degree polynomial, $q(x)$, with each user with the restriction that each polynomial have the same valuation at point 0, that is $q(0) = y$.

For each attributes associated with a user's identity the key generation algorithm will issue a private key component that is tied to the user's random polynomial $q(x)$. If the user is able to "match" at least d components of the ciphertext with their private key components, then they will be able to perform decryption. However, since the private key components are tied

to random polynomials, multiple user's are unable to combine them in anyway that allows for collusion attacks.

Recall that an IBE scheme has to create such that a ciphertext created using identity ω can be decrypted only by a secret key ω' where $|\omega \cap \omega'| \geq d$.

Let $\mathbb{G}_1$ be bilinear group of prime order p, and let g be a generator of $\mathbb{G}_1$. Additionally, let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ denote the bilinear map. A security parameter, k, will determine the size of the groups.

The Lagrange coefficient $\Delta_{i,S}$ for $i \in \mathbb{Z}_p$ and a set, S, of element in $\mathbb{Z}_p$ is defined as:

$$\Delta_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$$

Identities will be element subsets of some universe, U, of size $|U|$. Each element is associated with a unique integer in $\mathbb{Z}_p^*$.

Setup(d). First, the universe, U of elements are defined. For simplicity, the fist $|U|$ elements of $\mathbb{Z}_p$ are taken to be the universe. Namely, the integers $1, ..., |U|(modp)$.

Next, choose $t_1, ..., t_{|U|}$ uniformly at random from $\mathbb{Z}_p$. Finally, choose y uniformly at random in $\mathbb{Z}_p$. The published public parameters are:

$\mathbf{PP} =< \mathbf{T_1} = \mathbf{g^{t_1}}, ..., \mathbf{T_{|U|}} = \mathbf{g^{t_{|U|}}}, \mathbf{Y} = \mathbf{e(g,g)^y} >$

The master key is:

$\mathbf{MK} =< \mathbf{t_1}, ..., \mathbf{t_{|U|}}, \mathbf{y} >$

KeyGen($\mathbf{MK}, \omega$) To generate a private key for identity $\omega \subseteq U$, the following steps are taken. A $(d-1)$ degree polynomial q is randomly chosen such that $q(0) = y$. The private key consists of component, $(D_i)_{i \in \omega}$, where $D_i = g^{\frac{q(i)}{t_i}}$ for every $i \in \omega$. That is secret key $\mathbf{Sk}$ is given by

$\mathbf{SK} =< \mathbf{D_i}, \ \forall \mathbf{i} \in \omega >$

Encryption($\mathbf{PP}, \mathbf{M}, \omega'$) Encryption with public key ω' and message $M \in \mathbb{G}_2$ proceed as follows. First, a random value $s \in \mathbb{Z}_p$ is chosen. The ciphertext is then published as:

$$\mathbf{E} = (\omega', \mathbf{E}' = \mathbf{M}\mathbf{Y}^{\mathbf{s}}, \{\mathbf{E_i} = \mathbf{T_i^s}\}_{\mathbf{i}\in\omega'})$$

Decryption$(\mathbf{E}, \mathbf{SK})$ Suppose that a ciphertext, $\mathbf{E}$, is encrypted with a key for identity ω' and the identity ω corresponding to the private key $\mathbf{SK}$ satisfies the d-component matching i.e, $|\omega \cap \omega'| \geq d$. An arbitrary d-element subset, S, of $\omega \cap \omega'$. Then the ciphertext can be decrypted as:

$$\begin{aligned}
& E'/\prod_{i\in S}(e(D_i, E_i))^{\Delta_{i,S}(0)} \\
&= Me(g,g)^{sy}/\prod_{i\in S}(e(g^{\frac{q(i)}{t_i}}, g^{st_i}))^{\Delta_{i,S}(0)} \\
&= Me(g,g)^{sy}/\prod_{i\in S}(e(g,g)^{sq(i)})^{\Delta_{i,S}(0)} \\
&= M \ \ (using\ polynomial\ interpolation)
\end{aligned}$$

2.2.2 Proof of Security

(**Decisional Bilinear Diffie-Hellman (BDH) Assumption**). Suppose a challenger chooses $a, b, c, z \in \mathbb{Z}_p$ at random. The Decisional BDH assumption is that no polynomial-time adversary is to be able to distinguish the tuple($A = g^a, B = g^b, C = g^c, Z = e(g,g)^{abc}$) from the tuple($A = g^a, B = g^b, C = g^c, Z = e(g,g)^z$) with more than a negligible advantage. (**Decisional Modified Bilinear Diffie-Hellman (MBDH) Assumption**). Suppose a challenger chooses $a, b, c, z \in \mathbb{Z}_p$ at random. The Decisional BDH assumption is that no polynomial-time adversary is to be able to distinguish the tuple($A = g^a, B = g^b, C = g^c, Z = e(g,g)^{\frac{ab}{c}}$) from the tuple($A = g^a, B = g^b, C = g^c, Z = e(g,g)^z$) with more than a negligible advantage.

Theorem 2.2.1 *Assume that MBDH assumption holds. Then The Fuzzy Identity Based Encryption is IND-CPA secure.*

2.3 Large Universe Construction

In the previous section the size of the public parameters grows linearly with the number of possibles attributes in the universe. In this second scheme which uses all elements of $\mathbb{Z}_p^*$ as the universe, yet the public parameters only grow linearly in a parameter n, which is fixed as the maximum size identity, it can be decrypted to.

In addition to decreasing the public parameter size, having a large universe allows us to apply a collision-resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_p$ and use arbitrary string as attributes.

Let $\mathbb{G}_1$ be a bilinear group of prime order p, and let g be a generator of $\mathbb{G}_1$. Additionally, let $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ denote the bilinear map. The length of encryption identity is restricted to be n for some fixed n.

The Lagrange coefficient $\Delta_{i,S}$ for $i \in \mathbb{Z}_p$ and a set, S, of element in $\mathbb{Z}_p$ is defined as:

$$\Delta_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$$

Identities will be sets of n elements of $\mathbb{Z}_p^*$. Alternatively an identity can be described as a collection of n strings of arbitrary length and a collision resistant hash function H is used to hash strings into members of $\mathbb{Z}_p^*$. The construction is given below.

Setup(n, d) First, choose $g_1 = g^y, g_2 \in \mathbb{G}_1$.

Next, choose $t_1, ..., t_{n+1}$ uniformly at random from $\mathbb{G}_1$. Let N be the set $\{1, ..., n+1\}$ and a function, T is defined as:

$$T(x) = g_2^{x^n} \prod_{i=1}^{n+1} t_i^{\Delta_{i,N}(x)}$$

The function can be viewed as the function $g_2^{x^n} g^{h(x)}$ for some n degree polynomial h. The public parameter $\mathbf{PP}$ is published as: $g_1, g_2, t_1, ..., t_{n+1}$ and the master key $\mathbf{MK}$ is y.

KeyGen$(\mathbf{MK}, \omega)$. To generate a private key for identity ω the following steps are taken. A $d-1$ degree polynomial q is randomly chosen such that $q(0) = y$. The secret key $\mathbf{SK}$ will consist of two sets. The first set, $\{D_i\} i \in \omega$, where the elements are constructed as

$$D_i = g_2^{q(i)} T(i)^{r_i}$$

where r_i is a random member of $\mathbb{Z}_p$ defined for all all $i \in \omega$.

The other set is $\{d_i\} i \in \omega$ where the elements are constructed as

$$d_i = g^{r_i}$$

Encryption$(\mathbf{PP}, \mathbf{M}, \omega')$. Encryption with the public key ω' and message $M \in \mathbb{G}_2$ proceeds as follows.
First, a random value $s \in \mathbb{Z}_p$ is chosen. The ciphertext is then published as:

$$E \;=\; (\omega', E' = Me(g_1, g_2)^s, E'' = g^s, \{E_i = T(i)^s\}_{i \in \omega'})$$

Decryption$(\mathbf{E}, \mathbf{SK})$ The ciphertext, $\mathbf{E}$, is encrypted with a key for identify ω' and the secret key $\mathbf{SK}$ is associated with identity ω such that $|\omega \cap \omega'| \geq d$. An arbitrary d-element subset, S, of $\omega \cap \omega'$. Then, the ciphertext can be decrypted as:

$$\begin{aligned}
& E' / \prod_{i \in S} \left(\frac{e(d_i, E_i)}{e(D_i, E'')} \right)^{\Delta_{i,S}(0)} \\
&= Me(g_1, g_2)^s / \prod_{i \in S} \left(\frac{e(g^{r_i}, T(i)^s)}{e(g_2^{q(i)} T(i)^{r_i}, g^s)} \right)^{\Delta_{i,S}(0)} \\
&= Me(g_1, g_2)^s / \prod_{i \in S} \left(\frac{e(g^{r_i}, T(i)^s)}{e(g_2^{q(i)}, g^s) e(T(i)^{r_i}, g^s)} \right)^{\Delta_{i,S}(0)} \\
&= Me(g_1, g_2)^s / \prod_{i \in S} \frac{1}{e(g, g_2)^{q(i) s \Delta_{i,S}(0)}} \\
&= M \;\; (using\ polynomial\ interpolation)
\end{aligned}$$

2.3.1 Proof of Security

Theorem 2.3.1 *Assume that Decisional BDH assumption holds. Then the Fuzzy Identity Based Encryption is IND-CPA secure.*

2.4 Attribute-Based Encryption for Fine Grained Access Control of Encrypted Data(Key-Policy Attribute Based Encryption)

2.4.1 Access Trees

In this construction, user decryption keys will be identified with a set γ of attributes. A party who wishes to encrypt a message will specify through an access tree structure, a policy that private keys must satisfy in order to decrypt.

Access Tree

Let τ be a tree representing an access structure. Each non-leaf node of the tree represents a threshold gate, described by its children and a threshold value. If num_x is the number of children of a node x and k_x is its threshold value, then $0 < k_s \leq num_x$. Each leaf node x of the tree is described by an attribute and a threshold value $k_x = 1$

The parent of a node x in the tree is denoted by $parent(x)$. The access tree τ also defines an ordering between the children of every node, that is, the children of a node x are numbered from 1 to num_x. The function $index(x)$ returns such a number associated with a node x, where the index values are uniquely assigned to nodes in an arbitrary manner for a given access structure. Also for a leaf-node x, index is defined as $index(x) = attr(x)$ and $attr(x)$ is the attribute associated with it.

2.4.2 Satisfying an Access Tree

Let τ be an access tree with root r. denote by τ_x the sub-tree of τ rooted at the node x. Hence τ is the same as τ_r. If a set of attribute γ satisfies the access tree τ_x, it is denoted as $\tau_x(\gamma) = 1$. $\tau_x(\gamma)$ is computed recursively as follows. If x is a non-leaf node, evaluate $\tau_z(\gamma)$ for all children z of a node x. $\tau_x(\gamma)$ returns 1 if and only if at least k_x children return 1. If x is leaf node, then $\tau_x(\gamma)$ returns 1 iff $att(x) \in \gamma$.

2.4.3 Construction

Let $\mathbb{G}_1$ be a bilinear group of order p, and let g be a generator of $\mathbb{G}_1$. In addition, let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ denote the bilinear map. A security parameter, k, will determine the size of the groups. The Lagrange coefficient $\Delta_{i,S}$ for $i \in \mathbb{Z}_p$ and a set, S, of elements in $\mathbb{Z}_p$ is defined: $\Delta_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$. Each attribute will be associated with a unique element in $\mathbb{Z}_p^*$. Here the construction has four algorithms given below. **Setup** The universe of attribute is defined by $U = \{1, 2, ..., n\}$. Now, for each attribute $i \in U$, choose a number t_i uniformly at random from $\mathbb{Z}_p$. Finally, choose y uniformly at random in $\mathbb{Z}_p$. The published public parameter PP are

$\mathbf{PP} = T_1 = g^{t_1}, ..., T_{|U|} = g^{t_{|U|}}, Y = e(g, g)^y$ The master key MK is:

$\mathbf{MK} = t_1, ..., t_{|U|}, y.$

Encryption$(M, \gamma, \mathbf{PP})$. To encrypt a message $M \in \mathbb{G}_2$ under a set of attributes γ, choose a random value $s \in \mathbb{Z}_p$ and publish the ciphertext as:

$$E = (\gamma, E' = MY^s, \{E_i = T_i^s\}_{i \in \gamma}).$$

Key Generation(γ, MK) The algorithm outputs a key that enables the user to decrypt a message encrypted under a set of attribute γ if and only if $_(\gamma) = 1$. The algorithm proceed as follows. First choose a polynomial q_x for each node x(including the leaves) in the tree T. These polynomials are chosen in the following way in a top-down manner, starting from the root node r

For each node x in the tree, set the degree d_x of the polynomial q_x to be one less that the threshold value k_x of that node, that is, $d_x = k_x - 1$. Now, for the root node r, set $q_r(0) = y$ and d_r other points of the polynomial q_r randomly to define it completely. For any other node x, set $q_x(0) = q_{parent(x)} index(x)$ and choose d_x other points randomly to completely define q_x.

Once the polynomial have been decided, for each leaf node x, the following secret value is given to the user:

$$D_x = g^{\frac{q_x(0)}{t_i}} \ where \ i = attr(x).$$

The set of above secret values is the decryption key D.

Decryption(E, D) The decryption procedure is a recursive algorithm. First a recursive algorithm $DecryptNode(\mathbf{E}, \mathbf{D}, \mathbf{x})$ at a node x is defined that takes as input the ciphertext $E = (\gamma, E', \{E_i\}_{i\in\gamma})$, the private key D(it is assumed that the access tree T is embedded in the private key), and a node x in the tree. It outputs a group element of $\mathbb{G}_2$ or $\perp$.

Let $i = attr(x)$. If the node x is a leaf node then:

$$DecryptNode(E, D, x) \begin{cases} e(D^x, E_i) = e(g^{\frac{q_x(0)}{t_i}}) = e(g,g)^{s.q_x(0)} & \text{if } i \in \gamma \\ \perp & \text{otherwise} \end{cases}$$

Now the recursive case is considered when x is a non leaf node. The algorithm $DecryptNode(\mathbf{E}, \mathbf{D}, \mathbf{x})$ then proceeds as follows: For all nodes z that are children of x, it calls $DecryptNode(\mathbf{E}, \mathbf{D}, \mathbf{z})$ and stores the output as F_z. Let S_x be an arbitrary k_x-sized set of child nodes z such that $F_z \neq \perp$. If no such set exists then the node was not satisfied and the function returns $\perp$.

Otherwise, it is computed as:

$$\begin{aligned} F_x &= \prod_{z\in S_x} F_z^{\Delta_{i,s'_x}(0)},\ where\ i = index(z),\ S'_x = \{index(z) : z \in S_x\} \\ &= \prod_{z\in S_x} (e(g,g)^{s.q_z(0)})^{\Delta_{i,s'_x}(0)} \\ &= \prod_{z\in S_x} (e(g,g)^{s.q_{parent(z)}(index(z))})^{\Delta_{i,s'_x}(0)}\ \ (by\ construction) \\ &= \prod_{z\in S_x} e(g,g)^{s.q_z(0).\Delta_{i,s'_x}(0)} \\ &= e(g,g)^{sq_x(0)}\ \ (using\ polynomial\ interpolation) \end{aligned}$$

and return the result.

Now, the decryption algorithm simply calls the function on the root of the tree. $DecryptNode(\mathbf{E}, \mathbf{D}, \mathbf{r}) = \mathbf{e(g,g)^{ys}} = \mathbf{Y^s}$ if and only if the ciphertext satisfies the tree. Since, $E' = MY^s$ the decryption algorithm simply divides out Y^s and recovers the message M

2.4.4 Proof of Security

Theorem 2.4.1 *Assume that Decisional BDH assumption holds. Then Key-Policy Attribute Based Encryption is IND-CPA secure.*

2.5 Ciphertext-Policy Attribute-Based Encryption

2.5.1 Construction of CP-ABE scheme

In this section construction of the CP-ABE scheme in [[17] will be provided. First we describe the model of access trees and attributes for describing ciphertexts and private keys respectively. Next, the scheme will be described. Finally, we discuss security, efficiency, and key revocation of the scheme.

In this scheme private keys are attached with a set S of attributes. A party who wants to encrypt a message must specify an access structure along with the ciphertext and the attribute set corresponding to the private key must satisfy the access tree to decrypt the ciphertext.

Each interior node of the tree is a threshold gate and the leaves are associated with attributes. A user will be able to decrypt a ciphertext with a given key if and only if his/her attribute set satisfy the access tree. In this construction the attributes are used to identify the keys.

Access tree (τ) . Let τ be a tree representing an access structure. Clearly each non-leaf node of τ represents a threshold gate(described by its children and a threshold value.) If num_x is the number of children of a node x and k_x is its threshold value, then 0 ¡ $k_x \leq num_x$ (For example k_x = 1, describes an OR gate). Each leaf node x of the tree is described by an attribute and a threshold value k_x = 1. We also denote the parent of the node x in the tree by $\mathbf{parent(x)}$. The function $\mathbf{att(x)}$ is defined for a leaf node x and denotes the attribute associated with the leaf node x in τ. The access tree τ also defines an ordering between the children of every node, that is, the children of a node are numbered from 1 to *num*. The function $\mathbf{index(x)}$ returns

such a number associated with the node x which assigns unique value to nodes in an arbitrary manner.

Satisfying an access tree. Let τ be an access tree with root r. Denote by τ_x the subtree of τ rooted at the node x. Hence τ is the same as τ_r . If a set of attributes γ satisfies the access tree τ_x , we denote it as $\tau_x(\gamma) = 1$. We compute $\tau_x()$ recursively as follows. If x is a non-leaf node, evaluate $\tau_{x'}(\gamma)$ for all children $x^{'}$ of node x. $\tau_x(\gamma)$ returns 1 if and only if at least k_x children return 1. If x is a leaf node, then $\tau_x(\gamma)$ returns 1 if and only if att(x) $\in \gamma$.

Construction. Let $\mathbb{G}_0$ be a bilinear group of prime order p, and let g be a generator of $\mathbb{G}_0$. In addition, let e : $\mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$ denote the bilinear map. A security parameter, , will determine the size of the groups. We also define the Lagrange coefficient $\Delta_{i,S}$ for $i \in \mathbb{Z}_p$ and a set, S, of elements in $\mathbb{Z}_p$: $\Delta_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$. We will additionally employ a hash function $\mathbf{H} : \{0,1\}^* \rightarrow \mathbb{G}_0$ that we will model as a random oracle.

Setup. The setup algorithm will choose a bilinear group $\mathbb{G}_0$ of prime order p with generator g. Next it will choose two random exponents $\alpha, \beta \in \mathbb{Z}_p$. The public key is published as:

PK = ($\mathbb{G}_0$, g, h =g^β, f = $g^{1/\beta}$, $e(g,g)\alpha$)

MK = (β, g^α) is the master key

Encrypt(**PK**, M, τ). The encryption algorithm encrypts a message M under the tree access structure . The algorithm first chooses a polynomial q_x for each node x (including the leaves) in the tree τ . These polynomials are chosen in the following way in a topdown manner, starting from the root node R. For each node x in the tree, set the degree d_x of the polynomial q_x to be one less than the threshold value k_x of that node, that is, $d_x = k_x - 1$.

Starting with the root node R the algorithm chooses a random s $\in Z_p$ and sets $q_R(0)$ = s. Then, it chooses d_R other points of the polynomial q_R randomly to define it completely. For

any other node x, it sets $q_x(0) = q_{parent(x)}$ (index(x)) and chooses d_x other points randomly to completely define q_x .

Let, Y be the set of leaf nodes in τ . The ciphertext is then constructed by giving the tree access structure τ and computing

$\mathbf{CT}$ =(τ , $\tilde{C} = Me(g,g)^{\alpha s}$, C = h^s , $\forall$y $\in$ Y : $C_y = g^{q_y(0)}$, $C'_y = H(att(y))^{q_y(0)}$).

KeyGen ($\mathbf{MK}$, S). The key generation algorithm will take as input a set of attributes S and output a key that identifies with that set. The algorithm first chooses a random $r \in \mathbb{Z}_p$, and then random $r_j \in \mathbb{Z}_p$ for each attribute $j \in$ S.
Then it computes the key as

$\mathbf{SK}$ = (D = $g^{(\alpha+r)/\beta}$, $\forall j \in$ S : $D_j = g^r \cdot H(j)^{r_j}$, $D'_j = g^{r_j}$) .

Delegate($\mathbf{SK}$, $\tilde{S}$). The delegation algorithm takes in a secret key $\mathbf{SK}$, which is for a set S of attributes, and another set $\tilde{S}$ such that $\tilde{S} \subseteq$ S. The secret key is of the form $\mathbf{SK}$ = (D, $\forall j \in$ S : D_j , D'_j). The algorithm chooses random $\tilde{r}$ and $\tilde{r}_k$ $\forall k \in$ S. Then it creates a new secret key as

$\tilde{\mathbf{SK}}$ = ($\tilde{D} = Df^{\tilde{r}}$, $\forall$k $\in \tilde{S}$: $\tilde{D}_k$ = $D_k g^{\tilde{r} H(k)^{\tilde{r}_k}}$,$\tilde{D}'_k$ = $D'^{g^{\tilde{r}_k}}_k$)

The resulting secret key is a secret key for the set $\tilde{S}$. Since the algorithm re-randomizes the key, a delegated key is equivalent to one received directly from the authority.

Decrypt($\mathbf{CT}, \mathbf{SK}$). First a recursive algorithm $DecryptNode(\mathbf{CT}, \mathbf{SK}, \mathbf{x})$ is defined that takes as input a ciphertext $\mathbf{CT} = (\tau, \tilde{\mathbf{C}}, \mathbf{C}, \forall \mathbf{y} \in \mathbf{Y} : \mathbf{C_y}, \mathbf{C'_y})$, a private key $\mathbf{SK}$, which is associated with a set S of attributes, and a node x from .

If the node x is a leaf node and if $i = att(x)$ then $DecryptNode(\mathbf{CT}, \mathbf{SK}, \mathbf{x})$ is defined follows: If $i \in S$, then

$$\begin{aligned} DecryptNode(CT, SK, x) &= \frac{e(D_i, C_x)}{e(D'_i, C'_x)} \\ &= \frac{e(g^r.H(i)^{r_i}, g^{q_x(0)})}{e(g^{r_i}, H(i)^{q_x(0)})} \\ &= e(g,g)^{rq_x(0)} \end{aligned}$$

If $i \notin S$, then $DecryptNode(CT, SK, x) = \perp$.

Let x is a non-leaf node. The recursive algorithm $DecryprNode(\mathbf{CT}, \mathbf{SK}, \mathbf{x})$ is proceeded as follows: For all nodes z that are children of x, it calls $DecryprNode(\mathbf{CT}, \mathbf{SK}, \mathbf{x})$ and stores the output as F_z. Let S_x be an arbitrary k_x-sized set of child nodes z such that $F_z \neq \perp$.If no such set exists then the node was not satisfied and the function returns $\perp$.
Otherwise F_x is computed as follows.

$$\begin{aligned} F_x &= \prod_{z \in S_x} F_z^{\Delta_{i,s'_x}(0)},\ where\ i = index(z),\ S'_x = \{index(z) : z \in S_x\} \\ &= \prod_{z \in S_x} (e(g,g)^{r.q_z(0)})^{\Delta_{i,s'_x}(0)} \\ &= \prod_{z \in S_x} (e(g,g)^{r.q_{parent(z)}(index(z))})^{\Delta_{i,s'_x}(0)}\ \ (by\ construction) \\ &= \prod_{z \in S_x} e(g,g)^{r.q_z(0).\Delta_{i,s'_x}(0)} \\ &= e(g,g)^{rq_x(0)}\ \ (using\ polynomial\ interpolation) \end{aligned}$$

and return the result.

Now the algorithm begins by simply calling the function on the root node R of the tree τ. If the tree is satisfied by S, set $A = DecryptNode(\mathbf{CT}, \mathbf{SK}, \mathbf{x})$=$e(g,g)^{rq_R(0)} = e(g,g)^{rs}$. The algorithm now decrypts by computing

$\tilde{C}/(e(C,D)/A) = \tilde{C}/(e(h^s, g^{(\alpha+r)/\beta})/e(g,g)^{rs}) = M$

2.6 Security Intuitions

As in prior attribute based encryption schemes the main challenge in designing this scheme was to prevent against attacks from colluding users.The scheme randomizes users private keys such

that they cannot be combined. However the secret sharing must be embedded into the ciphertext instead to the private keys. In order to decrypt an attacker clearly must recover $e(g,g)^s$. In order to do this the attacker must pair C from the ciphertext with the D component from some user's private key. This will result in the desired value $e(g,g)^s$, but blinded by some value $e(g,g)^{rs}$. This value can be blinded out if and only if enough the user has the correct key components to satisfy the secret sharing scheme embedded in the ciphertext. Collusion attacks won't help since the blinding value is randomized to the randomness from a particular user's private key. While we described this scheme to be secure against chosen plaintext attacks, the security of the scheme can efficiently be extended to chosen ciphertext attacks by applying a random oracle technique such as that of the the Fujisaki-Okamoto transformation [13]. Alternatively, delegation mechanism can be leverged and Cannetti, Halevi, and Katz [6] method can be applied for achieving CCA-security.

The generic bilinear group model and the random oracle model are used to argue that no efficient adversary that acts generically on the groups underlying this scheme can break the security of the scheme with any reasonable probability.

The generic bilinear group model. We consider two random encodings ψ_0, ψ_1 of the additive group F_p , that is injective maps $\psi_0, \psi_1 : F_p \to \{0,1\}^m$, where $m > 3log(p)$. For i = 0, 1 we write $G_i = \psi_i$ (x) : x $\in F_p$. We are given oracles to compute the induced group action on $\mathbb{G}_0, \mathbb{G}_1$ and an oracle to compute a non-degenerate bilinear map e : $\mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$. We are also given a random oracle to represent the hash function H. We refer to $\mathbb{G}_0$ as a generic bilinear group.
The following theorem gives a lower bound on the advantage of a generic adversary in breaking our CP- ABE scheme.

Theorem 2.6.1 *Let ψ_0, ψ_1 , $\mathbb{G}_0, \mathbb{G}_1$ be defined as above. For any adversary $\mathcal{A}$, let q be a bound on the total number of group elements it receives from queries it makes to the oracles for the hash function, groups $\mathbb{G}_0$ and $\mathbb{G}_1$, and the bilinear map e, and from its interaction with the CP-ABE security game. Then we have that the advantage of the adversary in the CP-ABE security game is $\mathcal{O}(\frac{q^2}{p})$.*

2.7 Bounded Ciphertext-Policy Attribute-Based Encryption

2.7.1 Access Trees

In this construction, user decryption keys will be identified with a set γ of attributes. A party who wishes to encrypt a message will specify through an access tree structure, a policy that private keys must satisfy in order to decrypt.

Access Tree

Let τ be a tree representing an access structure. Each non-leaf node of the tree represents a threshold gate, described by its children and a threshold value.If num_x is the number of children of a node x and k_x is its threshold value, then $0 < k_x \leq num_x$. Each leaf node x of the tree is described by an attribute and a threshold value $k_x = 1$.

Let Φ_τ denote the set of all the non-leaf nodes in the tree τ. Further, let Ψ_τ be the set of all non-leaf nodes at depth $d-1$, where d is the depth of τ. The parent of a node x in the tree is denoted by $parent(x)$. The access tree τ also defines an ordering between the children of every node, that is, the children of a node x are numbered from 1 to num_x. The function $index(x)$ returns such a number associated with a node x, where the index values are uniquely assigned to nodes in an arbitrary manner for a given access structure. Also for a leaf-node x, index is defined as $index(x) = att(x)$ and $att(x)$ is the attribute associated with it.

2.7.2 Satisfying an Access Tree

Let τ be an access tree with root r. denote by τ_x the sub-tree of τ rooted at the node x. Hence τ is the same as τ_r. If a set of attribute γ satisfies the access tree τ_x, it is denoted as $\tau_x(\gamma) = 1$. $\tau_x(\gamma)$ is computed recursively as follows. If x is a non-leaf node, evaluate $\tau_z(\gamma)$ for all children z of a node x. $\tau_x(\gamma)$ returns 1 if and only if at least k_x children return 1. If x is leaf node, then $\tau_x(\gamma)$ returns 1 iff $att(x) \in \gamma$.

2.7.3 Universal Access Tree

Given a pair of integers values (d, num), defines complete num-ary tree τ of depth d, where each non-leaf has a threshold value of num. The leaf nodes in τ are empty, i.e, no attributes are assigned to the leaf nodes. Next, $num - 1$ new leaf nodes are attached to each non-leaf node x, thus increasing the cardinality of x to $2.num - 1$ while the threshold value num is left intact. Choose an arbitrary assignment of dummy attributes to these newly added leaf nodes for each x. the resultant tree τ is called a (d, num)-universal access tree.

2.7.4 Bounded Access Tree

A tree τ' is a (d, num)-bounded access tree if it has depth $d' \leq d$, and each non-leaf node in τ' exhibits a cardinality at most num.

2.7.5 Normal Form

Consider a (d, num)-bounded access tree τ'. τ' is said to be in (d, num)-normal form if

(a) it has depth $d' = d$ and

(b) all the leaves in τ' are at depth d.

2.7.6 Map between Access Tree

Consider a (d, num)-universal access tree τ and another tree τ' that is in (d, num)-normal form. A map between the nodes of τ' and τ is defined in the following way in a top down manner. First, the root of τ' is mapped to the root node of τ. Now suppose that x' in τ' is mapped to x in τ. Let $z'_1, z'_2, ..., z'_{num_{x'}}$ be the child nodes of x', ordered according to their index values. Then, for each child $z'_i (i \in [1, num_{x'})$ of x' in τ', set the corresponding child z_i (i.e. with index value $index(z'_i)$) of x in τ as the map of z'. This procedure is performed recursively, until each non-leaf node in τ' is mapped to a corresponding node in τ.

2.7.7 Construction

Let $\mathbb{G}_1$ be a bilinear group of order p and let g be a generator of $\mathbb{G}_1$. In addition, let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ denote the bilinear map. A security parameter, k, will determine the size of the groups. The Lagrange coefficient $\Delta_{i,s}$ for $i \in \mathbb{Z}_p$ and a set, S of elements in $\mathbb{Z}_p$ is defined as:

$$\Delta_{i,s}(x) = \prod_{j \in S,\, j \neq i} \frac{x-j}{i-j}$$

Each attribute is associated with a unique element in $\mathbb{Z}_p^*$. The construction follows.

Setup(d, num). The algorithm takes as input two system parameters, namely, (a) the maximum tree depth d, and (b) the maximum node cardinality num. The algorithm proceeds as follows. Define the universe of real attribute $U = \{1, 2, ...n\}$ and (num - 1)-sized universe of dummy attribute $U^* = \{n+1, ..., n+num-1\}$. Next, define a (d, num)-universal access tree τ.

Now, for each real attribute $j \in U$, choose a set of $|\Psi_\tau|$ numbers $\{ t_{j,x}\}_{x \in \Psi_\tau}$ uniformly at random from $\mathbb{Z}_p$. Further for each dummy attribute $j \in U^*$, choose a set of $|\Phi_\tau|$ numbers $\{ t^*_{j,x}\}_{x \in \Phi_\tau}$ uniformly at random from $\mathbb{Z}_p$. Finally, choose y uniformly at random from $\mathbb{Z}_p$. The public parameters $\mathbf{PP}$ are :

$$\mathbf{PP} = Y = e(g,g)^y, \{T_{j,x} = g^{t_{j,x}}\}_{j \in U, x \in \Psi_\tau}, \{T_{j,x}* = g^{t^*_{j,x}}\}_{j \in U^*, x \in \Phi_\tau}$$

The master key $\mathbf{MK}$ is:

$$\mathbf{MK} = y, \{t_{j,x}\}_{j \in U, x \in \Psi}, \{t^*_{j,x}\}_{j \in U^*, x \in \Phi_\tau}$$

KeyGen(γ, $\mathbf{MK}$). Consider a user A with an attribute set γ. The key generation algorithm outputs a private key D that enable A to decrypt a message encrypted under a (d, num)-bounded access tree τ' iff $\tau'(\gamma) = 1$.

The algorithm proceeds as follows. for each user, choose a random polynomial q_x for each non leaf node x in the universal access tree τ. These polynomials are chosen in the following way in a top down manner, starting from the root node r. For each x, set the degree c_x of the polynomial q_x to be one less than the threshold value, i.e., $c_x = num - 1$. Now, for the root

node r, set $q_r(0) = y$ and choose c_r other points of the polynomial q_r randomly to define it completely. For any other non-leaf node x, set $q_x(0) = q_{parent(x)}(index(x))$ and choose c_x other points randomly to completely define q_x. Once the polynomials have been decided, the following secret values are given to the user:
$\{D_{j,x} = g^{\frac{q_x(j)}{t_{j,x}}}\}_{j\in\gamma, x\in\Psi_\tau}, \{D^*_{j,x} = g^{\frac{q_x(j)}{t^*_{j,x}}}\}_{j\in U^*, x\in\Phi_\tau}$
The set of above secret values is the decryption key D.

Encryption(M, **PP**, τ'). To encrypt a message $M \in \mathbb{G}_2$, the encrypter ε first choose a (d, num)-bounded access tree τ'. ε then chooses an assignment of real attributes to the leaf nodes in τ'.
Now, to be able to encrypt the message M with the access tree τ', the encrypter first converts it to the normal form(if required). Next, ε defines a map between the nodes in τ' and the universal access tree τ. Finally, for each non-leaf node x in τ', ε chooses an arbitrary (num - k_x)-sized set w_x of dummy child nodes of map(x) in τ.

Let $f(j,x)$ be a boolean function such that $f(j,x) = 1$ if a real attribute $j \in U$ is associated with a leaf child of node $x \in \Psi_{\tau'}$ and 0 otherwise. Now, choose a random value $s \in \mathbb{Z}_p$ and publish the encryption E as:
$< T', E' = M.Y^s, \{E_{j,x} = T^s_{j,map(x)}\}_{j\in U, x\in\Psi_{\tau'}: f(j,x)=1}, \{E^*_{j,x} = T_{j,map(x)} * s\}_{j=att(x): z\in w_x, x\in\Phi_{\tau'}} >$

Decryption($\mathbf{E}, \mathbf{D}$). They define a recursive algorithm $DecryptNode(\mathbf{E}, \mathbf{D}, \mathbf{x})$ that takes as input the ciphertext E, the private key D, and a node x in τ'. It outputs a group element of $\mathbb{G}_2$ or $\perp$. First, consider the case when x is a leaf node. Let $j = att(x)$ ans w be the parent of x. Then value of the function:

$$DecryptNode(E, D, x) = \begin{cases} e(D_{j,map(w)}, E_{j,w}) = e(g^{\frac{q_{map(w)}(j)}{t_{j,map(w)}}}, g^{s.t_{j,map(w)}}) & \text{if} j \in \gamma \\ \perp & \text{Otherwise} \end{cases}$$

which reduces to $e(g,g)^{s.q_{map(w)}(j)}$ when $j \in \gamma$. Now consider the recursive case when x is a non-leaf node in τ'. The algorithm proceeds as follows: For all nodes z that are children x, it calls $DecryptNode(E, D, z)$ and stores the output as F_z. additionally, for each dummy node $z \in \omega_x$, it invokes a function $DecryptDummy(\mathbf{E}, \mathbf{D}, \mathbf{z})$ that is defined below, and stores the output as F_z. let j be the dummy attribute associate with z. Then value of the function

$DecryptDummy(\mathbf{E}, \mathbf{D}, \mathbf{z})$:

$DecryptDummy(E, D, z) = e(D^*_{j,map(x)}, E^*_{j,x}) = e(g^{\frac{q_{map(x)}(j)}{t^*_{j,map(x)}}}, g^{s.t^*_{j,map(x)}})$

which reduces to $e(g,g)^{s.q_{map(x)}(j)}$. Let Ω_x be an arbitrary k_x-sized set of child nodes z such that $F_z \neq \perp$. Further, let S_x be the union of the sets Ω_x and ω_x. Thus $|S_x| = num$. Let $\hat{g} = e(g,g)$. If no k_x-sized set Ω_x exists, then the node x was not satisfied and the function returns $\perp$. Otherwise, compute:

$$
\begin{aligned}
F_x &= \prod_{z \in S_x} F_z^{\Delta_{i,S'_x}(0)} \\
&= \prod_{z \in \Omega_x} F_z^{\Delta_{i,S'_x}(0)} \prod_{z \in \omega_x} F_z^{\Delta_{i,S'_x}(0)} \\
&= \begin{cases} \prod_{z \in \Omega_x} (\hat{g}^{s.q_{map(x)}(i)})^{\Delta_{i,S'_x}(0)} \prod_{z \in \omega_x} (\hat{g}^{s.q_{map(x)}(i)})^{\Delta_{i,S'_x}(0)} & \text{if } x \in \Psi_{T'} \\ \prod_{z \in \Omega_x} (\hat{g}^{s.q_{map(z)}(0)})^{\Delta_{i,S'_x}(0)} \prod_{z \in \omega_x} (\hat{g}^{s.q_{map(x)}(i)})^{\Delta_{i,S'_x}(0)} & \text{else} \end{cases} \\
&= \begin{cases} \prod_{z \in S_x} (\hat{g}^{s.q_{map(x)}(i)})^{\Delta_{i,S'_x}(0)} & \text{if } x \in \Psi_{\tau'} \\ \prod_{z \in \Omega_x} (\hat{g}^{s.q_{map(parent(z))}index(map(z)))})^{\Delta_{i,S'_x}(0)} \prod_{z \in \omega_x} (\hat{g}^{s.q_{map(x)}(i)})^{\Delta_{i,S'_x}(0)} & \text{else} \end{cases} \\
&= \prod_{z \in S_x} (\hat{g}^{s.q_{map(x)}(i)})^{\Delta_{i,S'_x}(0)} \\
&= \hat{g}^{s.q_{map(x)}(0)} \\
&= e(g,g)^{s.q_{map(x)}(0)} \quad (Using\ polynomial\ interpolation)
\end{aligned}
$$

Where

$$
i = \begin{cases} att(z) & \text{if z is a leaf node} \\ index(map(x)) & \text{otherwise} \end{cases}
$$

and $S'_x = \{i : z \in S_x\}$ and return the result.

Now that $DecryptNode$ has been defined, the decryption algorithm simply involves it on the root r' of τ'. It is observed that $DecryptNode(E, D, r') = e(g,g)^{sy}$ iff $\tau'(\gamma) = 1$ (It is noted that $F_{r'} = e(g,g)^{s.q_{map(r')}(0)} = e(g,g)^{s.q_r(0)} = e(g,g)^{sy}$, where r is the root of the universal tree τ). Since $E' = M.e(g,g)^{sy}$, the decryption algorithm simply divides out $e(g,g)^{sy}$ and recovers M.

2.7.8 Proof of Security

Theorem 2.7.1 *Assume that Decisional BDH assumption holds. Then the Bounded Ciphertext-Policy Attribute Based Encryption is IND-CPA secure.*

2.8 Provably Secure Ciphertext Policy Attribute Based Encryption

This is a CP-ABE schemes proposed in [8] in which access structures are AND gates on positive and negative attributes. This scheme has been proved to be chosen plaintext(CPA) secure under the decisional bilinear Diffie-Hellman(DBDH) assumption.

2.8.1 Basic Construction

For notational simplicity, let the set of attributes be $\mathcal{N}$:=1, . . . , n for some natural number n. We refer to attributes i and their negations $\neg i$ as literals. In this section, we con- sider access structures that consist of a single AND gate whose inputs are literals. This is denoted $\wedge_{i\in\mathcal{I}}\ \underline{i}$, where $\mathcal{I} \subseteq \mathcal{N}$ and every $\underline{i}$ is a literal (i.e., $ior\neg i$).

Setup. This algorithm selects:

• A bilinear group $\mathbb{G}_0$ of prime order p, with bilinear map $\mathbf{e} : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$,
• Random elements $y, t_1, ..., t_{3n}$ in $\mathbb{Z}_p$ and a random generator g of $\mathbb{G}_0$.

Let Y := $e(g, g)^y$ and $_k := g^{t_k}$ for each k $\in$ 1, . . . , 3n.

The public key is

$\mathbf{PK}$:= (e, g, Y, $T_1, ..., T_{3n}$).

The master secret key is

$\mathbf{MK}$:=(y, $t_1, ..., t_{3n}$) .

Intuitively, the public key elements T_i, T_{n+i} and T_{2n+i} correspond to the three types of occurrences of i: positive, negative and don't care.. Because of the technique we use to randomize secret key components, we must provide a don't care element for each attribute i not appearing in the AND gate. This should become clear after we introduce **KeyGen** and **Decrypt**.

Encrypt. Given a message M $\in \mathbb{G}_{\not\vdash}$ and an AND gate W =$\wedge_{i\in I}$ $\underline{i}$ the Encrypt algorithm first selects a random $s \in Z_p$ and sets $\tilde{C}$:= M $\cdot Y^s$ and C := g^s . For each $i \in I$, let C_i be T_is if $\underline{i} = i$ and T_{n+i} if $\underline{i}$ = ¬i. For each $i \in \mathcal{N} \setminus \mathcal{I}$, let C_i be T_is T_{2n+i} .

The ciphertext is **CT** := (W, $\tilde{C}$, $\hat{C}$, $C_i \| i \in N$) .

In total, Encrypt performs n + 1 exponentiations in $\mathbb{G}_0$, one exponentiation in $\mathbb{G}_1$ and one multiplication in $\mathbb{G}_1$. The ciphertext contains n + 1 elements of $\mathbb{G}_0$, one element of $\mathbb{G}_1$ and the description of W .

KeyGen. Let S denote the input attribute set. Every $i \in S$ is implictly considered a negative attribute. First, KeyGen selects random r_i from $\mathbb{Z}_p$ for every $i \in \mathcal{N}$ and sets r=:$\sum_{i=1}^{n} r_i$. Let $\hat{D}$ be g^{y-r}.For each $i \in \mathcal{N}$ let D_i be $g^{\frac{r_i}{t_i}}$ if $i \in$ S;otherwise, let it be $g^{\frac{r_i}{t_{n+i}}}$. Finally, let F_i be $g^{\frac{r_i}{t_{2n+i}}}$. for every $i \in N$.

The secret key is defined as

$\mathbf{SK}$:= ($\hat{D}$, D_i, F_i $\| i \in N$).

Note that we use the equation r=:$\sum_{i=1}^{n} r_i$. to bind together the D_i elements. (Similarly for the F_i elements.) This is a key difference between our scheme and the BSW scheme, and it is crucial in our reduction proof. The F_i elements are provided because every r_i must be recovered in order to decrypt. If i is a don't care for a particular encryption operation (i.e., i does not occur in the AND gate W), then F_i will be used for decryption, instead of D_i .

Decrypt. Suppose the input ciphertext is of the form $\mathbf{CT}$ = (W, $\tilde{C}$, $\hat{C}$, $C_i \| i \in N$) where

W = $\wedge_{i \in I}\ \underline{i}$. Also, let S denote the attribute set used to generate the input

secret key **SK**.

SK = ($\hat{D}$, $\{\ D_i, F_i \| i \in N\}$).

For each $i \in I$, Decrypt computes the pairing e(C_i, D_i).

If $\underline{i} = i$ and $i \in S$, then

e(C_i, D_i) = e($g^{t_i s}, g^{\frac{r_i}{t_i}}$) = $e(g,g)^{r_i s}$.

Similarly, if $i = \neg i$ and $i \notin S$, then

For each $i \in I$, Decrypt computes the pairing e(C_i, D_i) = e($g^{t_{n+i} s}$, $g^{\frac{r_i}{t_{2n+i}}}$) = $e(g,g)^{r_i s}$.

For each $i \notin I$, Decrypt computes the pairing

e(C_i, D_i) = e($g^{t_{2n+i} s}$, $g^{\frac{r_i}{t_{2n+i}}}$) = $e(g,g)^{r_i s}$.

Decrypt finishes as follows: M =$\frac{\tilde{C}}{Y^s}$=$\frac{\tilde{C}}{e(g,g)^{ys}}$, where

$e(g,g)^{ys} = e(g^s, g^{y-r}) e(g,g)^{r\Delta s} = e(\hat{C}, \hat{D}).\prod_{i=1}^{n} e(g,g)^{r_i s}$

2.8.2 Proof of Security

Theorem 2.8.1 *If a probabilistic polynomial-time adversary wins the CP-ABE game with non-negligible advantage, then we can construct a simulator that distinguishes a DBDH tuple from a random tuple with non-negligible advantage.*

Chapter 3

A Brief Survey on Predicate Based Encryption

Predicate encryption scheme(or Functional Encryption Scheme) is an encryption system where secret keys of an user are associated with predicates (i.e., boolean functions) in some class F, and a sender associates a ciphertext with an attribute in a set Σ; a ciphertext associated with the attribute $I \in \Sigma$ can be decrypted by a secret key SK_f corresponding to the predicate $f \in F$ if and only if $f(I) = 1$. The "basic" level of security achieved by such schemes guarantees, informally, that a ciphertext associated with attribute I hides all information about the underlying message unless one is in possession of a secret key giving the explicit ability to decrypt. I.e., if an adversary $\mathcal{A}$ holds keys $\mathbf{SK}_{f_1}, ..., \mathbf{SK}_{f_l}$, then $\mathcal{A}$ learns nothing about the message if $f_1(I) = ... = f_l(I) = 0$. This security notion is called payload hiding. Attribute hiding,where ciphertext hide the message as above,is another strong notion of security but it additionally requires that the ciphertext hide all information about the associated attribute I except that which is explicitly leaked by the keys in one's possession; i.e., an adversary holding secret keys as above learns only $f_1(I), ..., f_l(I)$, and learns nothing else about I.

3.1 Definitions

In this section we define the syntax of predicate encryption and the security properties of predicate encryption system. Σ denotes an arbitrary set of attributes and *F* denotes an arbitrary set of predicates over Σ.

Definition 1: ($PredicateEncryptionScheme$). A predicate encryption scheme for the class of predicates F over the set of attributes Σ consists of four ppt algorithms **Setup, GenKey, Enc, Dec** such that:

• The **Setup** algorithm takes as input the security parameter 1^n and outputs a (master) public parameter PP and a (master) secret key SK.

• The **GenKey** algorithmtakes as input the master secret key SK and a (description of a) predicate $f \in F$. It outputs a key SK_f .

• The **Enc** algorithm takes as input the public key PP, an attribute $I \in \Sigma$, and a message M in some associated message space. It returns a ciphertext C. We write this as $C \leftarrow Enc_{PP}, (I, M)$.

• The **Dec** algorithm takes as input a secret key SK_f and a ciphertext C. It outputs either a message M or the distinguished symbol $\perp$.

For correctness, we require that for all n, all $(\mathbf{PP}, \mathbf{SK})$ generated by Setup(1^n), all $f \in F$, any key $SK_f \leftarrow GenKey_{SK}(f)$, and all $I \in \Sigma$:

• If $f(I) = 1$ then $Dec_{SK_f}(Enc_{PP}(I, M)) = M$.

• If $f(I) = 0$ then $Dec_{SK_f}(Enc_{PP}(I, M)) = \perp$ with all but negligible probability.

Definition 2: A predicate encryption scheme with respect to *F* and Σ is *attribute hiding* if for all *ppt adversaries* $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following experiment is *negligible* in the security parameter n:

1) $\mathcal{A}(1^n)$ outputs $I_0, I_1 \in \Sigma$.

2) Setup(1^n) is run to generate PP and SK, and the adversary is given PP.

3) $\mathcal{A}$ may adaptively request keys for any predicates $f_1, ..., f_l \in F$ subject to the restriction that $f_i(I_0) = f_i(I_1)$ for all i. In response, $\mathcal{A}$ is given the corresponding keys $SK_{f_i} \leftarrow GenKey_{SK}(f_i)$.

4) $\mathcal{A}$ outputs two equal-length messages M_0, M_1 . If there is an i for which $f_i(I_0) = f_i(I_1)$ = 1, then it is required that $M_0 = M_1$. A random bit b is chosen, and $\mathcal{A}$ is given the ciphertext $C \leftarrow Enc_{PP}(I_b, M_b)$.

5) The adversary may continue to request keys for additional predicates, subject to the same restrictions as before.

6) $\mathcal{A}$ outputs a bit b' , and succeeds if b = b'.

The advantage of $\mathcal{A}$ is the absolute value of the difference between its success probability and 1/2.

For predicate-only encryption schemes, attribute hiding is define by simply omitting the messages in the above experiment. Payload hiding, a strictly weaker notion of security, is defined by forcing $I_0 = I_1 = I$ in the above experiment (in which case $\mathcal{A}$ has no possible advantage if it ever holds that $f_i(I) = 1$).

3.2 Brief Background on Pairings and Assumptions

In this section we concentrate mainly on bilinear group of *composite* order first used in cryptographic applications by [3]. Here we use groups whose order N is a product of three distinct primes. Let $\mathcal{G}$ be an algorithm that takes as input a security parameter 1^n and outputs a tuple $(p, q, r, \mathbb{G}, \mathbb{G}_T , \hat{e})$ where p, q, r are distinct primes, $\mathbb{G}$ and $\mathbb{G}_T$ are two cyclic groups of order $N = pqr$, and $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a non-degenerate bilinear map, $\forall u, v \in \mathbb{G}$ and $\forall a, b \in \mathbb{Z}_N$ we have $\hat{e}(u^a, v^b)$ = $\hat{e}(u, v)ab$, and if g generates $\mathbb{G}$ then $\hat{e}(g, g)$ generates $\mathbb{G}_T$. We assume mul-

tiplication in $\mathbb{G}$ and $\mathbb{G}_T$, as well as the bilinear map $\hat{e}$, are all computable in time polynomial in n.

We use the notation $\mathbb{G}_p, \mathbb{G}_q, \mathbb{G}_r$ to denote the subgroups of $\mathbb{G}$ having order $p, q,$ and r, respectively. Clearly $\mathbb{G}$ is isomorphic to $\mathbb{G}_p \times \mathbb{G}_q \times \mathbb{G}_r$. Note also that if g is a generator of $\mathbb{G}$, then the element g^{pq} is a generator of $\mathbb{G}_r$; the element g^{pr} is a generator of $\mathbb{G}_q$; and the element g^{qr} is a generator of $\mathbb{G}_p$. Furthermore, if $h_p \in \mathbb{G}_p$ and $h_q \in \mathbb{G}_q$ then $\hat{e}(h_p, h_q) = \hat{e}((g^{qr})^{\alpha_1}, (g^{pr})^{\alpha_2}) = \hat{e}(g^{\alpha_1}, g^{r\alpha_2})^{pqr} = 1$,

where $\alpha_1 = log_{g^{qr}} h_p$ and $\alpha_2 = log_{g^{pr}} h_q$. A similar rule holds whenever $\hat{e}$ is applied to elements in distinct subgroups.

We now state the assumptions which are used to prove security of the construction

Assumption 1. Let $\mathcal{G}$ be as above. We say that $\mathcal{G}$ satisfies Assumption 1 if the advantage of any ppt algorithm $\mathcal{A}$ in the following experiment is negligible in the security parameter n:

1) $\mathcal{G}(1^n)$ is run to obtain $(p, q, r, \mathbb{G}, \mathbb{G}_T, \hat{e})$. Set $N = pqr$, and let g_p, g_q, g_r be generators of $\mathbb{G}_p, \mathbb{G}_q$, and $\mathbb{G}_r$, respectively.
2) Choose random $Q_1, Q_2, Q_3 \in \mathbb{G}_q$, random $R_1, R_2, R_3 \in \mathbb{G}_r$, random $a, b, s \in \mathbb{Z}_p$, and a random bit b. Give to $\mathcal{A}$ the values $(N, \mathbb{G}, \mathbb{G}_T, \hat{e})$ as well as:

$$g_p, g_r, g_q R_1, g_p^b, g_p^{b^2}, g_p^a g_q, g_p^{ab} Q_1, g_p^s, g_p^{bs} Q_2 R_2$$

If $b = 0$ give $\mathcal{A}$ the value $T = g_p^{b^2 s} R_3$, while if $b = 1$ give $\mathcal{A}$ the value $T = g_p^{b^2 s} Q_3 R_3$.
3) $\mathcal{A}$ outputs a bit b' , and succeeds if $b = b'$.

The advantage of $\mathcal{A}$ is the absolute value of the difference between its success probability and 1/2.

Assumption 2. Let $\mathcal{G}$ be as above. We say that $\mathcal{G}$ satisfies Assumption 2 if the advantage of any ppt algorithm $\mathcal{A}$ in the following experiment is negligible in the security parameter n:

1) $\mathcal{G}(1^n)$ is run to obtain $(p, q, r, \mathbb{G}, \mathbb{G}_T, \hat{e})$. Set $N = pqr$, and let g_p, g_q, g_r be generators of $\mathbb{G}_p, \mathbb{G}_q$, and $\mathbb{G}_r$, respectively.

2) Choose random $h \in \mathbb{G}_p$ and $Q_1, Q_2 \in \mathbb{G}_q$, random $s, \psi \in \mathbb{Z}_q$, and a random bit b. Give to $\mathcal{A}$ the values (N, $\mathbb{G}, \mathbb{G}_T, \hat{e}$) as well as

$g_p, g_q, g_r, h, g_p^s, h^s Q_1, g_p^\gamma Q_2, \hat{e}(g_p, h)^\gamma$.

If $b = 0$ then give $\mathcal{A}$ the value $\hat{e}(g_p, h)^{\gamma s}$, while if $b = 1$ then give $\mathcal{A}$ a random element of $\mathbb{G}_T$.

3) $\mathcal{A}$ outputs a bit $b^{'}$, and succeeds if $b = b^{'}$.

The advantage of $\mathcal{A}$ is the absolute value of the difference between its success probability and 1/2.

Both the above assumptions imply the hardness of finding any non-trivial factor of N .

3.3 Predicate Only Encryption Scheme

We first describe the construction of the scheme in detail then we go to the security proof for the scheme.

3.3.1 Construction of Predicate Only Encryption Scheme

Setup(1^n) : The setup algorithm first runs $\mathcal{G}(1^n)$ to obtain $(p, q, r, \mathbb{G}, \mathbb{G}_T, \hat{e})$ with $\mathbb{G}$ = $\mathbb{G}_p \times \mathbb{G}_q \times \mathbb{G}_r$. Next, it computes g_p , g_q , and g_r as generators of $\mathbb{G}_p$, $\mathbb{G}_q$, and $\mathbb{G}_r$, respectively. It then chooses $R_{1,i}$, $R_{2,i} \in \mathbb{G}_r$ and $h_{1,i}$, $h_{2,i} \in \mathbb{G}_p$ uniformly at random for $i = 1$ to n, and $R_0 \in \mathbb{G}_r$ uniformly at random. The public parameters include ($N = pqr$, $\mathbb{G}$, $\mathbb{G}_T$, $\hat{e}$) along with:

PP = $(g_p, Q = g_q R_0, g_r, \{H_{1,i} = h_{1,i} R_{1,i}, H_{2,i} = h_{2,i} R_{2,i}\}_{i=1}^n)$

The master secret key **SK** is

SK = $(p, q, r, g_q, \{h_{1,i}, h_{2,i}\}_{i=1}^n)$.

Enc$(\mathbf{PP}, \overrightarrow{x})$: Let x = $(x_1, ..., x_n)$ with $x_i \in \mathbb{Z}_N$. This algorithm chooses random s, α, β

$\in \mathbb{Z}_N$ and $R_{3,i}, R_{4,i} \in \mathbb{G}_r$ for $i = 1$ to n. (a random element R $\in \mathbb{G}_r$ can be sampled, even without knowing the factorization of N , by choosing random $\delta \in \mathbb{Z}_N$ and setting $R = g_r^{\delta}$.) It outputs the ciphertext

C = ($C_0 = g_p^s, \{C_{1,i} = H_{1,i}^s Q^{\alpha x_i} R_{3,i}, C_{2,i} = H_{2,i}^s Q^{\beta x_i} R_{4,i}\}_{i=1}^n$).

GenKey(**SK**, $\overrightarrow{v}$): Let $\overrightarrow{v}$ = $(v_1, ..., v_n)$. This algorithm chooses random $r_{1,i}, r_{2,i} \in \mathbb{Z}_p$ for $i = 1$ to n, random $R_5 \in \mathbb{G}_r$, random $f_1, f_2 \in \mathbb{Z}_q$, and random $Q_6 \in \mathbb{G}_q$. It then outputs

$$\mathbf{SK}_{\overrightarrow{v}} = (K = R_5 Q_6 h^{-\gamma} \prod_{i=1}^n h_{1,i}^{r_{1,i}} h_{2,i}^{r_{2,i}}, \{K_{1,i} = g_p^{r_{1,i}} g_q^{f_{1,v_i}}, K_{2,i} = g_p^{r_{2,i}} g_q^{f_{2,v_i}}\}_{i=1}^n)$$

Dec($\mathbf{SK}_{\overrightarrow{v}}, C$): Let **C** = $(C_0, \{C_{1,i}, C_{2,i}\}_{i=1}^n)$ and $\mathbf{SK}_{\overrightarrow{v}}$ = $(K, \{K_{1,i}, K_{2,i}\}_{i=1}^n)$ be as above. The decryption algorithm outputs 1 iff

$$\hat{e}(C_0, K) \prod_{i=1}^n \hat{e}(C_{1,i}, K_{1,i}) \hat{e}(C_{2,i}, K_{2,i}) = 1$$

3.3.2 Correctness of Predicate Only Encryption Scheme

To see that correctness holds, let **C** and $\mathbf{SK}_{\overrightarrow{v}}$ be as above.

$\hat{e}(C_0, K) \prod_{i=1}^n \hat{e}(C_{1,i}, K_{1,i}) \hat{e}(C_{2,i}, K_{2,i})$

= $\hat{e}(g_p^s, R_5 Q_6 \prod_{i=1}^n h_{1,i}^{r_{1,i}} h_{2,i}^{r_{2,i}}) \prod_{i=1}^n \hat{e}(H_{1,i}^s Q^{\alpha x_i} R_{3,i}, g_p^{r_{1,i}} g_q^{f_{1,v_i}}) \hat{e}(H_{2,i}^s Q^{\beta x_i} R_{4,i}, g_p^{r_{2,i}} g_q^{f_{2,v_i}})$

= $\hat{e}(g_p^s, \prod_{i=1}^n h_{1,i}^{r_{1,i}} h_{2,i}^{r_{2,i}}) \prod_{i=1}^n \hat{e}(h_{1,i}^s g_q^{\alpha x_i}, g_p^{r_{1,i}} g_q^{f_{1,v_i}}) \hat{e}(h_{2,i}^s g_q^{\beta x_i}, g_p^{r_{2,i}} g_q^{f_{2,v_i}})$

= $\prod_{i=1}^n \hat{e}(g_q, g_q)^{(\alpha f_1 + \beta f_2) x_i v_i}$

= $\prod_{i=1}^n \hat{e}(g_q, g_q)^{(\alpha f_1 + \beta f_2 \ mod \ q) < \overrightarrow{x}, \overrightarrow{v} >}$

Where α, β chosen randomly from $\mathbb{Z}_N$ and f_1, f_2 are random in $\mathbb{Z}_q$

If $< \overrightarrow{x}, \overrightarrow{v} >$ = 0 then the above value evaluates to 1. If $< \overrightarrow{x}, \overrightarrow{v} > \neq 0$ mod N but $< \overrightarrow{x}, \overrightarrow{v} >$ = 0 mod q, then the above would evaluate to 1 and reveals a non trivial factor of N which is assumed to be hard. So this occurs with negligible probability. If $< \overrightarrow{x}, \overrightarrow{v} > \neq 0$ mod

N and $< \overrightarrow{x}, \overrightarrow{v} > \neq 0$ mod q then with all but negligible probability the above evaluates to an element other than identity.

3.3.3 Security Proof for Predicate Only Encryption Scheme

The scheme has been proved to be IND-CPA Assumption 1 (By proving the theorem below) by constructing a sequence of Games $\mathbf{Game_{Real}}, \mathbf{Game_1}, \mathbf{Game_2}, \mathbf{Game_3}, \mathbf{Game_4}$ played by a Challenger $\mathcal{C}$ with the Adversary $\mathcal{A}$.

$\mathbf{Game_{Real}}, \mathbf{Game_1}, \mathbf{Game_2}, \mathbf{Game_3}, \mathbf{Game_4}$ are defined as follows:

$\mathbf{Game_{Real}}$: This is the Real game where the challenge ciphertext is generated as a proper encryption using $\overrightarrow{x}$. (Clearly $\overrightarrow{x}$, $\overrightarrow{y}$ denote the two vectors output by the adversary.) s, $\alpha, \beta \in \mathbb{Z}_N$ and $\{R_{3,i}, R_{4,i}\} \in \mathbb{G}_r$ are chosen randomly and the ciphertext is computed as

C = ($C_1 = g_p^s, \{C_{1,i} = H_{1,i}^s Q^{\alpha x_i} R_{3,i}, C_{2,i} = H_{2,i}^s Q^{\beta x_i} R_{4,i}\}_{i=1}^n$).

$\mathbf{Game_1}$: In this game the $C_{2,i}$ component is generated as a proper encryption using $\overrightarrow{0}$.That is s, $\alpha, \beta \in \mathbb{Z}_N$ and $\{R_{3,i}, R_{4,i}\} \in \mathbb{G}_r$ are chosen randomly and the ciphertext is computed as

C = ($C_1 = g_p^s, \{C_{1,i} = H_{1,i}^s Q^{\alpha x_i} R_{3,i}, C_{2,i} = H_{2,i}^s R_{4,i}\}_{i=1}^n$).

$\mathbf{Game_2}$: In this game the $C_{2,i}$ component is generated as a proper encryption using $\overrightarrow{y}$.That is s, $\alpha, \beta \in \mathbb{Z}_N$ and $\{R_{3,i}, R_{4,i}\} \in \mathbb{G}_r$ are chosen randomly and the ciphertext is computed as

C = ($C_1 = g_p^s, \{C_{1,i} = H_{1,i}^s Q^{\alpha x_i} R_{3,i}, C_{2,i} = H_{2,i}^s Q^{\beta y_i} R_{4,i}\}_{i=1}^n$).

$\mathbf{Game_3}$: In this game the $C_{1,i}$ component is generated as a proper encryption using $\overrightarrow{0}$.That is s, $\alpha, \beta \in \mathbb{Z}_N$ and $\{R_{3,i}, R_{4,i}\} \in \mathbb{G}_r$ are chosen randomly and the ciphertext is computed as

C = ($C_1 = g_p^s, \{C_{1,i} = H_{1,i}^s R_{3,i}, C_{2,i} = H_{2,i}^s Q^{\beta y_i} R_{4,i}\}_{i=1}^n$).

$\mathbf{Game_4}$: In this game the $C_{1,i}$ component is generated as a proper encryption using $\overrightarrow{y}$.That is s, $\alpha, \beta \in \mathbb{Z}_N$ and $\{R_{3,i}, R_{4,i}\} \in \mathbb{G}_r$ are chosen randomly and the ciphertext is computed as

C = ($C_1 = g_p^s, \{C_{1,i} = H_{1,i}^s Q^{\alpha y_i} R_{3,i}, C_{2,i} = H_{2,i}^s Q^{\beta y_i} R_{4,i}\}_{i=1}^n$).

The proof of the theorem 3.5.1 is done by proving the indistinguishability (Using only Assumption 1.) of ($\mathbf{Game_{Real}}$ and $\mathbf{Game_0}$), ($\mathbf{Game_1}$ and $\mathbf{Game_2}$), ($\mathbf{Game_2}$ and $\mathbf{Game_3}$) and ($\mathbf{Game_3}$ and $\mathbf{Game_4}$) .

Let ($\overrightarrow{a}$, $\overrightarrow{b}$) denote a ciphertext encrypted using vector $\overrightarrow{a}$ in the first sub-system and $\overrightarrow{b}$ in the second sub-system. To prove indistinguishability between the case when the challenge ciphertext is associated with $\overrightarrow{x}$ (which corresponds to ($\overrightarrow{x}$, $\overrightarrow{x}$)) and the case when the challenge ciphertext is associated with $\overrightarrow{y}$ (which corresponds to ($\overrightarrow{y}$, $\overrightarrow{y}$)), Here a sequence of intermediate hybrid games ($\overrightarrow{x}$, $\overrightarrow{0}$), ($\overrightarrow{x}$, $\overrightarrow{y}$), ($\overrightarrow{0}$, $\overrightarrow{y}$), has been used showing indistinguishability in each case. That is, it has been shown

($\overrightarrow{x}$, $\overrightarrow{x}$) $\approx$ ($\overrightarrow{x}$, $\overrightarrow{0}$) $\approx$ ($\overrightarrow{x}$, $\overrightarrow{y}$) $\approx$ ($\overrightarrow{0}$, $\overrightarrow{y}$) $\approx$ ($\overrightarrow{y}$, $\overrightarrow{y}$),

To prove the desired following theorem.

Theorem 3.3.1 *If $\mathcal{G}$ satisfies Assumption 1 (In Section 3.2) then the scheme described (In Section 3.3.1) is an **attribute hiding predicate-only encryption scheme**.*

3.4 A Full-Fledged Predicate Encryption Scheme

Here,the scheme is the extension on Predicate only scheme in the sense of Definition 1 in section 3.1 .

3.4.1 Construction of A Full-Fledged Predicate Encryption Scheme

Setup(1^n) : The setup algorithm first runs G(1n) to obtain $(p, q, r, \mathbb{G}, \mathbb{G}_T, \hat{e})$ with G = $\mathbb{G}_p \times \mathbb{G}_q \times \mathbb{G}_r$. Next, it computes g_p , g_q , and g_r as generators of $\mathbb{G}_p$, $\mathbb{G}_q$, and $\mathbb{G}_r$, respectively. It then chooses $R_{1,i}$, $R_{2,i} \in \mathbb{G}_r$ and $h_{1,i}$, $h_{2,i} \in \mathbb{G}_p$ uniformly at random for $i = 1$ to n, and $R_0 \in \mathbb{G}_r$ uniformly at random. It also chooses random $\gamma \in \mathbb{Z}_p$ and $h \in \mathbb{G}_p$. The public parameters include ($N = pqr$, $\mathbb{G}$, $\mathbb{G}_T$, $\hat{e}$) along with:

PP = $(g_p, Q = g_q R_0, g_r, \underline{P = \hat{e}(g_p, h)^{\gamma}}, \{H_{1,i} = h_{1,i} R_{1,i}, H_{2,i} = h_{2,i} R_{2,i}\}_{i=1}^{n})$

The master secret key **SK** is (p, q, r, $g_q, \underline{h^{-\gamma}}, \{h_{1,i}, h_{2,i}\}_{i=1}^{n}$) .

Enc($\mathbf{PP}$, $\overrightarrow{x}$): Let x = $(x_1, ..., x_n)$ with $x_i \in \mathbb{Z}_N$. This algorithm chooses random s, $\alpha, \beta \in Z_N$ and $R_{3,i}, R_{4,i} \in \mathbb{G}_r$ for $i = 1$ to n. (a random element $R \in \mathbb{G}_r$ can be sampled, even without knowing the factorization of N , by choosing random $\delta \in Z_N$ and setting R = g_r^{δ} .)

It outputs the ciphertext **C** = ($\underline{C' = M.P^s}, C_0 = g_p^s, \{C_{1,i} = H_{1,i}^s Q^{\alpha x_i} R_{3,i}, C_{2,i} = H_{2,i}^s Q^{\beta x_i} R_{4,i}\}_{i=1}^{n}$).

GenKey(**SK**, $\overrightarrow{v}$): Let v = $(v_1, ..., v_n)$. This algorithm chooses random $r_{1,i}, r_{2,i} \in \mathbb{Z}_p$ for $i = 1$ to n, random $R_5 \in \mathbb{G}_r$, random $f_1, f_2 \in \mathbb{Z}_q$, and random $Q_6 \in \mathbb{G}_q$. It then outputs

$\mathbf{SK}_v = (K = R_5 Q_6 h^{-\gamma} \prod_{i=1}^{n} h_{1,i}^{r_{1,i}} h_{2,i}^{r_{2,i}}, \{K_{1,i} = g_p^{r_{1,i}} g_q^{f_{1,v_i}}, K_{2,i} = g_p^{r_{2,i}} g_q^{f_{2,v_i}}\}_{i=1}^{n})$

Dec($\mathbf{SK}_{\overrightarrow{v}}, C$): Let **C** = $(C_0, \{C_{1,i}, C_{2,i}\}_{i=1}^{n})$ and $SK_{\overrightarrow{v}}$ = $(K, \{K_{1,i}, K_{2,i}\}_{i=1}^{n})$ be as above. The decryption algorithm outputs

$\underline{C'} \hat{e}(C_0, K) \prod_{i=1}^{n} \hat{e}(C_{1,i}, K_{1,i}) \hat{e}(C_{2,i}, K_{2,i})$

decryption never returns an error (i.e, 0). In this scheme when $< \overrightarrow{v} . \overrightarrow{x} >$ = 0 then the output is essentially a random element in the order-q subgroup of $\mathbb{G}_T$.

3.4.2 Security Proof for Full-Fledged Predicate Encryption Scheme

The scheme has been proved to be IND-CPA-secure under Assumption 1 and Assumption 2 (By proving the theorem below) by constructing a sequence of Games $\mathbf{Game_{Real}}$, $\mathbf{Game_1}$, $\mathbf{Game_2}$, $\mathbf{Game_3}$, $\mathbf{Game_4}$, $\mathbf{Game_5}$, $\mathbf{Game_6}$ played by a Challenger $\mathcal{C}$ with the Adversary $\mathcal{A}$.

$\mathbf{Game_{Real}}, \mathbf{Game_1}, \mathbf{Game_2}, \mathbf{Game_3}, \mathbf{Game_4}, \mathbf{Game_5}, \mathbf{Game_6}$ are defined as follows:

$\mathbf{Game_{Real}}$: This is the Real IND-CPA security game where the challenge ciphertext is generated as a proper encryption of M_0 using $\overrightarrow{x}$. (Clearly $\overrightarrow{x}, \overrightarrow{y}$ denote the two vectors output by the adversary.) s, $\alpha, \beta \in \mathbb{Z}_N$ and $\{R_{3,i}, R_{4,i}\} \in \mathbb{G}_r$ are chosen randomly and the ciphertext is computed as

C = ($C^{'} = M_0 \dot{P^s}, C_0 = g_p^s, \{C_{1,i} = H_{1,i}^s Q^{\alpha x_i} R_{3,i}, C_{2,i} = H_{2,i}^s Q^{\beta x_i} R_{4,i}\}_{i=1}^n$).

$\mathbf{Game_1}$: The challenge ciphertext is generated as a proper encryption of a random element of $\mathbb{G}_T$, but still using $\overrightarrow{x}$. I.e., the ciphertext is formed as above except that C is chosen uniformly from $\mathbb{G}_T$.

$\mathbf{Game_2}$: In this game the $C_{2,i}$ component is generated as a proper encryption using $\overrightarrow{0}$ .That is $s, \alpha, \beta \in \mathbb{Z}_N$ and $\{R_{3,i}, R_{4,i}\} \in \mathbb{G}_r$ are chosen randomly and the ciphertext is computed as

C = ($C^{'}, C_0 = g_p^s, \{C_{1,i} = H_{1,i}^s Q^{\alpha x_i} R_{3,i}, C_{2,i} = H_{2,i}^s R_{4,i}\}_{i=1}^n$).

$\mathbf{Game_3}$: In this game the $C_{2,i}$ component is generated as a proper encryption using $\overrightarrow{y}$.That is s, $\alpha, \beta \in \mathbb{Z}_N$ and $\{R_{3,i}, R_{4,i}\} \in \mathbb{G}_r$ are chosen randomly and the ciphertext is computed as

C = ($C^{'}, C_0 = g_p^s, \{C_{1,i} = H_{1,i}^s Q^{\alpha x_i} R_{3,i}, C_{2,i} = H_{2,i}^s Q^{\beta y_i} R_{4,i}\}_{i=1}^n$).

$\mathbf{Game_4}$ and $\mathbf{Game_5}$: These games are defined symmetrically to $\mathbf{Game_2}$ and $\mathbf{Game_3}$, as discussed in the Security definition of the previous scheme. We continue to let $C^{'}$ be a random element of $\mathbb{G}_T$. Note that $\mathbf{Game_5}$ corresponds to a proper encryption of a random element of $\mathbb{G}_T$ using $\overrightarrow{y}$.

$\mathbf{Game_6}$: The challenge ciphertext is generated as a proper encryption of M_1 using $\overrightarrow{y}$.

The proof of the theorem 3.4.1 is done by proving the indistinguishability (Using both Assumption 1. and Assumption 2.) of ($\mathbf{Game_{Real}}$ and $\mathbf{Game_1}$), ($\mathbf{Game_1}$ and $\mathbf{Game_2}$), ($\mathbf{Game_2}$ and $\mathbf{Game_3}$), ($\mathbf{Game_3}$ and $\mathbf{Game_4}$), ($\mathbf{Game_4}$ and $\mathbf{Game_5}$) and ($\mathbf{Game_5}$ and $\mathbf{Game_6}$)

To prove the desired following theorem.

Theorem 3.4.1 *If $\mathcal{G}$ satisfies Assumptions 1 and 2(In Section 3.2) then the scheme described (In Section 3.4.1) is an **attribute-hiding predicate encryption scheme**.*

3.5 Hidden vection Encryption(HVE)

Let $\overrightarrow{x}$ be vector of length over the alphabet 0, 1 and $\overrightarrow{y}$ vector of the same length over the alphabet $\{0, 1, *\}$. Define the predicate Match($\overrightarrow{x}$, $\overrightarrow{y}$) = TRUE if and only if for any $i \in [l]$, it holds that $x_i = y_i$ or $y_i = *$. That is, the two vectors must match only in the positions j where yj $\neq$ * . This predicate is called Hidden Vector Encryption (henceforth, abbreviated in HVE)

A Hidden Vector Encryption scheme is a tuple of four efficient probabilistic algorithms **(Setup, Encrypt, KeyGen, Test)** with the following semantics.

Setup$(1^\lambda, 1^l)$: Outputs the public parameters **Pp** and the master secret key **Msk**.

KeyGen$(Msk, \overrightarrow{y})$: Takes as input the master secret key **Msk** and a vector $\overrightarrow{y} \in \{0, 1, *\}$, and outputs a secret key $\mathbf{Sk}_{\overrightarrow{y}}$.

Encrypt$(Pp, \overrightarrow{x})$: Takes as input the public parameters **Pp** and a vector $\overrightarrow{x} \in \{0, 1\}^l$ and outputs a ciphertext **Ct**.

Test$(Pp, Ct, Sk_{\overrightarrow{y}})$: takes as input the public parameters **Pp**, a ciphertext **Ct** encrypting $\overrightarrow{x}$ and a secret key $Sk_{\overrightarrow{y}}$ and outputs Match($\overrightarrow{x}$, $\overrightarrow{y}$).

Correctness of HVE. For correctness we require that for all pairs **(Pp, Msk)** output by **Setup**$(1^\cdot 1^l)$, it holds that for vectors $\overrightarrow{x} \in \{0, 1\}^l$ and $\overrightarrow{y} \in \{0, 1, *\}^l$, we have that

Test$(Pp, Encrypt(Pp, x), KeyGen(Msk, y))$ = **Match**(x, y) except with negligible in λ probability.

3.5.1 Security Definition for Hidden Vector Encryption Scheme

We give two security notions depending on the type of queries $\mathcal{A}$ is allowed to ask. We formalize the two notions by means of security games $\mathbf{Game_{Real}}(\epsilon)$, with $\epsilon \in \{0, 1\}$, between an Adversary $\mathcal{A}$ and a Challenger $\mathcal{C}$. $\mathbf{Game_{Real}}$ is defined as follows:

Setup. $\mathcal{C}$ runs the Setup algorithm on input ($1^{\lambda}, 1^{l}$) (given in unary) to generate public parameters Pp and master secret key Msk. $\mathcal{C}$ starts the interaction with $\mathcal{A}$ on input Pp.

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

Challenge Construction. Upon receiving a pair $x_0, x_1 \in \{0, 1\}^{l}$, $\mathcal{C}$ randomly picks $\eta \in \{0, 1\}$ and return $\mathbf{Encrypt}(\mathbf{Pp}, \overrightarrow{x_{\eta}})$.

At the end of the game, $\mathcal{A}$ outputs a guess $\eta^{'}$ for η. We say that $\mathcal{A}$ wins if $\eta^{'} = \eta$ and for all $\overrightarrow{y}$ for which $\mathcal{A}$ has seen a secret key, it holds that **Match**$(x_0, \overrightarrow{y})$ = **Match**$(x_1, \overrightarrow{y}) = \epsilon$.
The advantage $Adv^{\mathcal{A}}_{HHVE}(\lambda)$ of $\mathcal{A}$ is defined to be the prob[$\mathcal{A}$ wins the game] - 1/2 . We are now ready for the following definition.

Definition 3. A Hidden Vector Encryption scheme is $\epsilon-$ secure if for all probabilistic polynomial time ϵ adversaries $\mathcal{A}$, we have that $Adv^{\mathcal{A}}_{HVE}(\lambda)$ is a negligible function of λ.

3.5.2 Complexity Assumptions

The third assumption that we state is a subgroup-decision type assumption for bilinear settings with groups of order product of four primes. 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. For a generator $\mathcal{G}$ returning bilinear settings of order product of four primes, we define the following distribution. First pick a random bilinear setting
$\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, e)$ by running $\mathcal{G}(1^\lambda)$ and then pick $A_3 \leftarrow \mathbb{G}_{p_3}, A_{13} \leftarrow \mathbb{G}_{p_1p_3}, A_{12} \leftarrow \mathbb{G}_{p_1p_2}, A_4 \leftarrow \mathbb{G}_{p_4}, T_1 \leftarrow \mathbb{G}_{p_1p_3}, T_2 \leftarrow \mathbb{G}_{p_2p_3}$. 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 $Adv_3^{\mathcal{A}}(\lambda)$ = $|Prob[\mathcal{A}(D, T_1) = 1] - Prob[\mathcal{A}(D, T_2) = 1]|$

Assumption 3: We say that Assumption 1 holds for generator $\mathcal{G}$ if for all probabilistic polynomial-time algorithms $\mathcal{A}$, $Adv_3^{\mathcal{A}}(\lambda)$ is a negligible function of λ.

Our fourth assumption can be seen as the Decision Diffie-Hellman Assumption for composite order groups. Specifically, Assumption 4 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. For a generator $\mathcal{G}$ returning bilinear settings of order product of four primes, we define the following distribution. First pick a random bilinear setting $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, e)$ by running $\mathcal{G}(1^\lambda)$ and then pick $A_1 \leftarrow \mathbb{G}_{p_1}, A_2 \leftarrow \mathbb{G}_{p_2}, A_3 \leftarrow \mathbb{G}_{p_3}, A_4, B_4, C_4, D_4 \leftarrow \mathbb{G}_{p_4}, \alpha, \beta \leftarrow \mathbb{Z}_{p_1}, T_2 \leftarrow G_{p_1p_4}$ and set $T_1 = A_1^{\alpha\beta}D_4$ and $D = (\mathcal{I}, A_1, A_2, A_3, A_4, A_4^B, A_4^C)$.
We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 4 to be $Adv_4^{\mathcal{A}}(\lambda)$ = $|Prob[\mathcal{A}(D, T_1) = 1] - Prob[\mathcal{A}(D, T_2) = 1]|$

Assumption 4: We say that Assumption 3 holds for generator $\mathcal{G}$ if for all probabilistic polynomial-time algorithms $\mathcal{A}$, $Adv_4^{\mathcal{A}}(\lambda)$ is a negligible function of λ.

Assumption 5 is a generalization of Assumption 4 in the sense it posits the difficult of deciding if two triplets sharing an element are both Diffie-Hellman (looking at the formal definition below, the two triplets are the one composed of elements whose $\mathbb{G}_{p_1}$ parts are respectively $(A_1^\alpha, A_1^\beta, A_1^{\alpha\beta}$) and $(A_1^\gamma, A_1^{\alpha\beta}, A_1^{\alpha\beta\gamma}))$ given a third related Diffie-Hellman triplets (composed

of elements whose G_{p_1} parts are respectively $(A_1^{\alpha\gamma}, A_1^{\beta}, A_1^{\alpha\beta\gamma}$)). More formally, we have the following definition.

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 $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, e)$ by running $\mathcal{G}(1^\lambda)$ and then pick

$A_1 \leftarrow \mathbb{G}_{p_1}, A_2 \leftarrow \mathbb{G}_{p_2}, A_3 \leftarrow \mathbb{G}_{p_3}, A_4, B_4, C_4, D_4, E_4, F_4, G_4 \leftarrow \mathbb{G}_{p_4}, \alpha, \beta, \gamma \leftarrow \mathbb{Z}_{p_1}$,
$T_2 \leftarrow \mathbb{G}_{p_1p_4}$ and set $T_1 = A_1^{\alpha\beta}G_4$ and $D = (\mathcal{I}, A_1, A_2, A_3, A_4, A_1^{\alpha}B_4, A_1^{\beta}C_4, A_1^{\gamma}D_4, A_1^{\alpha\gamma}E_4, A_1^{\alpha\beta\gamma}F_4)$.
We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 5 to be $Adv_5^{\mathcal{A}}(\lambda) = |Prob[\mathcal{A}(D, T_1) = 1] - Prob[\mathcal{A}(D, T_2) = 1]|$

Assumption 5: We say that Assumption 3 holds for generator $\mathcal{G}$ if for all probabilistic polynomial-time algorithms $\mathcal{A}$, $Adv_5^{\mathcal{A}}(\lambda)$ is a negligible function of λ.

Our final assumption is, like Assumption 3, a subgroup-decision type of assumption. More formally, for a generator $\mathcal{G}$ returning bilinear settings of order N product of five primes, we define the following distribution. First pick a random bilinear setting $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, e)$ by running $\mathcal{G}(1^\lambda)$ and then pick :

$A_2 \leftarrow \mathbb{G}_{p_2}, A_3 \leftarrow \mathbb{G}_{p_3}, A_4, B_4 \leftarrow \mathbb{G}_{p_4}, A_{14}, B_{14} \leftarrow G_{p_1p_4}$
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 6 to be $Adv_6^{\mathcal{A}}(\lambda) = |Prob[\mathcal{A}(D, T_1) = 1] - Prob[\mathcal{A}(D, T_2) = 1]|$

Assumption 6: We say that Assumption 3 holds for generator $\mathcal{G}$ if for all probabilistic polynomial-time algorithms $\mathcal{A}$, $Adv_6^{\mathcal{A}}(\lambda)$ is a negligible function of λ.

3.5.3 0-Secure HVE

Construction of 0-Secure HVE Scheme

we describe the construction for a 0-secure (also called match revealing) HVE scheme below. We assume without loss of generality that the vectors $\overrightarrow{y}$ of the keys have at least two indices i, j such that $y_i, y_j \neq *$.

Setup$(1^\lambda, 1^l)$: The setup algorithm chooses a description of a bilinear group $\mathcal{I}$ = ($N = p_1p_2p_3p_4$, $\mathbb{G}, \mathbb{G}_T, e$) with known factorization by running a generator algorithm $\mathcal{G}$ on input 1^λ . The setup algorithm chooses random $g_1 \in \mathbb{G}_{p_1},\ g_2 \in \mathbb{G}_{p_2},\ g_3 \in \mathbb{G}_{p_3},\ g_4 \in \mathbb{G}_{p_4}$. For i $\in [l]$ and b $\in \{0, 1\}$, the algorithm chooses random $t_{i,b} \in \mathbb{Z}_N$ and random $R_{i,b} \in \mathbb{G}_{p_3}$ and sets $T_{i,b} = g_1^{t_{i,b}} R_{i,b}$.

The public parameters are $\mathbf{Pp}$ = $[N, g_3, (T_{i,b})_{i\in[l],b\in\{0,1\}}]$ and
The master secret key is $\mathbf{Msk}$ = $[g_{12}, g_4, (t_{i,b})_{i\in[l],b\in\{0,1\}}]$, where $g_{12} = g_1g_2$

KeyGen$(\mathbf{Msk}, \overrightarrow{y})$: Let $S_{\overrightarrow{y}}$ be the set of indices i such that $y_i \neq *$ The key generation algorithm chooses random $a_i \in \mathbb{Z}_N$ for i $\in S_{\overrightarrow{y}}$ under the constraint that $\Sigma_{i\in S_{\overrightarrow{y}}}\Sigma_{i\in[l]}a_i$ = 0. For i$\in S_{\overrightarrow{y}}$, the algorithm chooses random $W_i \in \mathbb{G}_{p_4}$ (the W_i are chosen by raising g_4 to a random power) and sets $Y_i = g_{12}^{\frac{a_i}{t_{i,y_i}}} W_i$. The algorithm returns the tuple $(Y_i)_{i\in S_{\overrightarrow{y}}}$.

Notice that here we used the fact that $S_{\overrightarrow{y}}$ has size at least 2.

Encrypt$(\mathbf{Pp}, \overrightarrow{x})$: The encryption algorithm chooses random $s \in \mathbb{Z}_N$. For $i \in [l]$, the algorithm chooses random $Z_i \in \mathbb{G}_{p_3}$ (the Z_i are chosen by raising g_3 to a random power) and sets $X_i = T_{i,x_i}^s Z_i$, and returns the tuple $(X_i)_{i\in[l]}$.

Test$(\mathbf{Ct}, \mathbf{Sk}_{\overrightarrow{y}})$: The test algorithm computes T =$\prod_{i\in S_{\overrightarrow{y}}} e(X_i, Y_i)$ and returns **TRUE** iff $T = 1$.

Security of 0-Secure HVE Scheme

The security proof of 0-Secure HVE is based on Assumption 3 and 4. For a probabilistic polynomial-time 0-adversary $\mathcal{A}$ which makes q queries for $\mathbf{KeyGen}$, the proof of security will contain a sequence of $(q+2)$ games $\mathbf{Game_{Real}}, \mathbf{Game_0}, \ldots, \mathbf{Game_q}$ between a probabilistic polynomial time 0-Adversary $\mathcal{A}$ and a Challenger $\mathcal{C}$.

$\mathbf{Game}_{Real}$ is the real security game defined in the security definition of HVE(Section 3.5.1)

$\mathbf{Game}_0$ is almost same as $\mathbf{Game}_{Real}$ except that $\mathcal{C}$ uses g_2 instead of g_1 to construct the public parameters $\mathbf{Pp}$ given to A. Specifically $\mathcal{C}$ constructs,

$\mathbf{Pp}$= $[N, g_3, (T_{i,b})_{i\in[l],b\in\{0,1\}}]$, $\mathbf{Pp}'$ = $[N, g_3, (T'_{i,b})_{i\in[l],b\in\{0,1\}}]$ and $\mathbf{Msk}$ = $[g_{12}, g_4, (t_{i,b})_{i\in[l],b\in\{0,1\}}]$

where $\mathcal{C}$ 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 $g_{12} = g_1 g_2$. For each i $\in$ [l] and b $\in \{0,1\}$, $\mathcal{C}$ chooses random $t_{i,b} \in \mathbb{Z}_N$ and $R_{i,b} \in \mathbb{G}_{p_3}$ and sets $T_{i,b} = g_2^{t_{i,b}} R_{i,b}$ and $T'_{i,b} = g_1^{t_{i,b}} R_{i,b}$.

$\mathbf{Game}_k$ for $1 \leq k \leq q$ is almost equivalent to $\mathbf{Game}_0$ 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 $\mathbf{Game_0}$.

Specifically for the first k -queries $\mathcal{C}$ chooses random $a_i, c_i \in \mathbb{Z}_N$ for i $\in S_{\overrightarrow{y}}$ (where $\overrightarrow{y}$ is the input vector), under the constraint that $\Sigma_{i\in S_{\overrightarrow{y}}} \Sigma_{i\in[l]}\ a_i$ = 0. For i$\in S_{\overrightarrow{y}}$, the algorithm chooses random $W_i \in \mathbb{G}_{p_4}$ and sets

$Y_i = g_1^{c_i}\ g_2^{\frac{a_i}{t_{i,y_i}}} W_i$. The algorithm returns the tuple $(Y_i)_{i\in S_{\overrightarrow{y}}}$.

The remaining $(q-k)$ queries are answered like in $\mathbf{Game_0}$.

The security Proof is based on the follwing two Lemmas.

Lemma 3.1. Suppose there exists a PPT algorithm $\mathcal{A}$ such that $Adv^{\mathcal{A}}_{Game_{Real}} - Adv^{\mathcal{A}}_{Game_0}$ = ϵ , for $1 \leq k \leq q$. Then, there exists a PPT algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 3.

Lemma 3.2. Suppose there exists a PPT algorithm $\mathcal{A}$ such that $Adv^{\mathcal{A}}_{Game_{k-1}} - Adv^{\mathcal{A}}_{Game_k}$ = ϵ , for $1 \leq k \leq q$. Then, there exists a PPT algorithm $\mathcal{B}$ with advantage atleast $\frac{\epsilon}{2l}$ in breaking Assumption 4.

Following the Lemmas it has been proved that the $\mathbf{Game_q}$ has no advantage i.e, $Adv^{\mathcal{A}}_{Game_q}$ = 0 which proves the following Theorem.

Theorem 3.5.1 *If Assumptions 3 and 4 hold for generator $\mathcal{G}$, then the HVE scheme presented is 0-secure .*

3.5.4 1-Secure HVE

Construction of 1-Secure HVE Scheme

Setup$(1^\lambda, 1^l)$: The setup algorithm chooses a description of a bilinear group $\mathcal{I}$ = ($N = p_1p_2p_3p_4$, $\mathbb{G}, \mathbb{G}_T, e$) with known factorization by running a generator algorithm $\mathcal{G}$ on input 1^λ . The setup algorithm chooses random $g_1 \in \mathbb{G}_{p_1}, g_2 \in \mathbb{G}_{p_2}, g_3 \in \mathbb{G}_{p_3}, g_4 \in \mathbb{G}_{p_4}$. For i ∈ [l] and b ∈ $\{0, 1\}$, the algorithm chooses random $t_{i,b} \in \mathbb{Z}_N$ and random $R_{i,b} \in \mathbb{G}_{p_3}$ and sets $T_{i,b} = g_1^{t_{i,b}} g_4^{v_i} R_{i,b}$.

The public parameters are $\mathbf{Pp}$ = $[N, g_3, (T_{i,b})_{i\in[l],b\in\{0,1\}}]$ and
The master secret key is $\mathbf{Msk}$ = $[g_{12}, g_4, (t_{i,b})_{i\in[l],b\in\{0,1\}}, (v_i)_{i\in[l]}]$, where $g_{12} = g_1g_2$

KeyGen$(Msk, \overrightarrow{y})$: Let $S_{\overrightarrow{y}}$ be the set of indices i such that $y_i \neq *$ The key generation algorithm chooses random $a_i \in \mathbb{Z}_N$ for $i \in S_{\overrightarrow{y}}$ under the constraint that $\sum_{i\in S_{\overrightarrow{y}}} a_i$ = 0. For i∈$S_{\overrightarrow{y}}$

, the algorithm sets $Y_i = g_{12}^{\frac{a_i}{t_{i,y_i}}} g_4^{\frac{a_i}{v_i}}$.
The algorithm returns the tuple $(Y_i)_{i \in S_{\overrightarrow{y}}}$.

Notice that here we used the fact that $S_{\overrightarrow{y}}$ has size at least 2.

Encrypt$(Pp, \overrightarrow{x})$: The encryption algorithm chooses random $s \in \mathbb{Z}_N$. For $i \in [l]$, the algorithm chooses random $Z_i \in \mathbb{G}_{p_3}$ (the Z_i are chosen by raising g_3 to a random power) and sets $X_i = T_{i,x_i}^s Z_i$, and returns the tuple $(X_i)_{i \in [l]}$.

Test$(Ct, Sk_{\overrightarrow{y}})$: The test algorithm computes T =$\prod_{i \in S_{\overrightarrow{y}}} e(X_i, Y_i)$ and returns **TRUE** iff $T = 1$.

Security of 1-Secure HVE Scheme

The security proof of 1-Secure HVE is based on Assumption 3 and 6. The proof of security contains a sequence of three games between a polynomial-time 1-Adversary $\mathcal{A}$ and a Challenger $\mathcal{C}$. The first game, $\mathbf{Game_{Real}}$ is the real HVE security game. The remaining games are $\mathbf{Game_0}$ and $\mathbf{Game_1}$

Game$_{Real}$ is the real security game defined in the security definition of HVE(Section 3.5.1)

Game$_0$ is almost same as **Game**$_{Real}$ except that $\mathcal{C}$ uses g_2 instead of g_1 to construct the public parameters $\mathbf{Pp}$ given to A. Specifically $\mathcal{C}$ constructs,

$\mathbf{Pp}$ = $[N, g_3, (T_{i,b})_{i \in [l], b \in \{0,1\}}]$, $\mathbf{Pp}'$ = $[N, g_3, (T'_{i,b})_{i \in [l], b \in \{0,1\}}]$ and $\mathbf{Msk}$ = $[g_{12}, g_4, (t_{i,b})_{i \in [l], b \in \{0,1\}}]$

where $\mathcal{C}$ 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 $g_{12} = g_1 g_2$. For each i $\in [l]$ and b $\in \{0, 1\}$, $\mathcal{C}$ chooses random $t_{i,b} \in \mathbb{Z}_N$, $v_i \in \mathbb{Z}_N$ and $R_{i,b} \in \mathbb{G}_{p_3}$ and sets $T_{i,b} = g_2^{t_{i,b}} g_4^{v_i} R_{i,b}$ and $T'_{i,b} = g_1^{t_{i,b}} g_4^{v_i} R_{i,b}$.

Game$_1$ is almost like **Game**$_0$ except in the query phase $\mathcal{C}$ answers the queries in the following

way.

On input vector $\overrightarrow{y}$, for $i \in S_{\overrightarrow{y}}$, $\mathcal{C}$ chooses random $a_i, b_i \in \mathbb{Z}_N$ under the constraint that i$\in S_{\overrightarrow{y}}$ $\Sigma_{i \in S_{\overrightarrow{y}}} a_i = \Sigma_{i \in S_{\overrightarrow{y}}} b_i = 0$. $\mathcal{C}$ sets, for $i \in S_{\overrightarrow{y}}$,

$$Y_i = g_2^{\frac{a_i}{t_{i,y_i}}} \dot{g}_4^{\frac{b_i}{v_i}} \dot{W}_i$$

The security Proof here also is based on the follwing two lemmas.

Lemma 3.3. Suppose there exists a PPT algorithm $\mathcal{A}$ such that $Adv^{\mathcal{A}}_{Game_{Real}} - Adv^{\mathcal{A}}_{Game_0}$ = ϵ , for $1 \leq k \leq q$. Then, there exists a PPT algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 3.

Lemma 3.4. Suppose there exists a PPT algorithm $\mathcal{A}$ such that $Adv^{\mathcal{A}}_{Game_0} - Adv^{\mathcal{A}}_{Game_1}$ = ϵ , for $1 \leq k \leq q$. Then, there exists a PPT algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 6.

Following the Lemmas it has been proved that the $\mathbf{Game_1}$ has no advantage i.e, $Adv^{\mathcal{A}}_{Game_1}$ = 0 which proves the following Theorem.

Theorem 3.5.2 *If Assumptions 3 and 6 hold for generator $\mathcal{G}$, then the HVE scheme presented is 1-secure.*

3.5.5 Hierarchical HVE

A **Hierarchical HVE scheme (HHVE)** consists of five efficient algorithms ($\mathbf{Setup}$, $\mathbf{Encrypt}$, $\mathbf{KeyGen}$, $\mathbf{Test}$, $\mathbf{Delegate}$).

Setup$(1^\lambda, 1^l)$: The setup algorithm chooses a description of a bilinear group $\mathcal{I}$ = ($N = p_1p_2p_3p_4$, $\mathbb{G}, \mathbb{G}_T, e$) with known factorization by running a generator algorithm $\mathcal{G}$ on input 1^λ . The setup algorithm chooses random $g_1 \in \mathbb{G}_{p_1}, g_2 \in \mathbb{G}_{p_2}, g_3, R \in \mathbb{G}_{p_3}, g_4 \in \mathbb{G}_{p_4}$. For $i \in [l]$

and b $\in \{0,1\}$, the algorithm chooses random $t_{i,b} \in \mathbb{Z}_N$ and random $R_{i,b} \in \mathbb{G}_{p_3}$ and sets $T_{i,b} = g_1^{t_{i,b}} g_4^{v_i} R_{i,b}$.

The public parameters are **Pp** = $[N, g_3, g_4, g_1.R, (T_{i,b})_{i\in[l],b\in\{0,1\}}]$ and
The master secret key is **Msk** = $[g_{12}, (t_{i,b})_{i\in[l],b\in 0,1}]$, where $g_{12} = g_1 g_2$

KeyGen$(Msk, \overrightarrow{y})$: Let $S_{\overrightarrow{y}}$ be the set of indices i such that $y_i \neq *$ The key generation algorithm chooses random $a_i \in \mathbb{Z}_N$ for i $\in S_{\overrightarrow{y}}$ under the constraint that $\sum_{i\in[l]} a_i$ = 0.

$$Y_i = \begin{cases} g_{12}^{\frac{a_i}{t_{i,y_i}}} R_i & for \ \ i \in S_{\overrightarrow{y}} \\ g_{12}^{a_i} R_i & for \ \ i \notin S_{\overrightarrow{y}} \end{cases}$$

The algorithm also sets for $i \notin S_{\overrightarrow{y}}$ and $b \in \{0,1\}$ $\quad D_{i,b} = g_{12}^{\frac{a_i}{t_{i,b}}} R_{i,b}$

The algorithm returns the tuple $\mathbf{Sk}_{\overrightarrow{y}}$= $[(Y_i)_{i\in[l]}, (D_{i,b})_{i\notin S_{\overrightarrow{y}},b\in\{0,1\}}]$. Notice that here we used the fact that $S_{\overrightarrow{y}}$ has size at least 2.

Encrypt$(Pp, \overrightarrow{x})$: The encryption algorithm chooses random $s \in \mathbb{Z}_N$ and $Z \in \mathbb{G}_{p_3}$. For $i \in [l]$, the algorithm chooses random $Z_i \in \mathbb{G}_{p_3}$ (the Z_i are chosen by raising g_3 to a random power) and sets $X_0 = (g_1 R)^s Z$ and $X_i = T_{i,x_i}^s Z_i$ for each $i \in [l]$, and Returns the Tuple

Ct= $[X_0, (X_i)_{i\in[l]}]$.

Test$(Ct, Sk_{\overrightarrow{y}})$: The test algorithm computes T =$e(X_0, \prod_{i\notin S_{\overrightarrow{y}}} Y_i) \prod_{i\in S_{\overrightarrow{y}}} e(X_i, Y_i)$ and returns **TRUE** iff T = 1 and **FALSE** otherwise.

Delegate$(Pp, Sk_{\overrightarrow{y}}, \overrightarrow{y}, \overrightarrow{w})$: On a input of $\mathbf{Sk}_{\overrightarrow{y}}$= $[(Y_i^{'})_{i\in[l]}, (D_{i,b})'_{i\notin S_{\overrightarrow{y}},b\in\{0,1\}}]$ for vector $\overrightarrow{y}$ the delegation algorithm chooses random $z \in \mathbb{Z}_N$. For $i \in S_{\overrightarrow{w}}$, the algorithm chooses random $R_i \in \mathbb{G}_{p_4}$ and, for $i \notin S_{\overrightarrow{w}}$ and b $\in \{0,1\}$, random $R_{i,b} \in \mathbb{G}_{p_4}$.

The Delegation algorithm computes Y_i for $i \in S_{\overrightarrow{w}}$ as

$$Y_i = \begin{cases} {Y_i'}^z R_i & for \ \ i \in S_{\overrightarrow{y}} \\ {D'_{i,w_i}}^z R_i & for \ \ i \notin S_{\overrightarrow{y}} \end{cases}$$

and

$D_{i,b}$ = ${D'_{i,b}}^z R_{i,b}$ for $i \notin S_{\overrightarrow{w}}$ and $b \in \{0,1\}$

and returns the Secret Key as $\mathbf{Sk}_{\overrightarrow{w}} = [(\mathbf{Y_i})_{\mathbf{i}\in[\mathbf{l}]}, (\mathbf{D_{i,b}})_{\mathbf{i}\notin\mathbf{S}_{\overrightarrow{y}},\mathbf{b}\in\{\mathbf{0},\mathbf{1}\}}]$

Remark 1. Let $\mathbf{Pp}$ =$[N, g_3, g_4, g_1\dot{R}, (T_{i,b})_{i\in[l],b\in\{0,1\}}]$ and $\mathbf{Msk}$ = $[g_1g_2, (t_{i,b})_{i\in[l],b\in\{0,1\}}$] be a pair public parameter and master secret key output by *Setup* algorithm and consider $\mathbf{Pp}^{'}$ = $[N, g_3, g_4, \hat{g}_1R, (T'_{i,b})_{i\in[l],b\in\{0,1\}}]$ and $\mathbf{Msk}^{'}$ = $[\hat{g}_1g_2, (t_{i,b})_{i\in[l],b\in\{0,1\}}$] with $T'_{i,b} = \hat{g}_1^{t_{i,b}} Z_{i,b}$ where $\hat{g}_1 \in \mathbb{G}_{p_1} and Z_{i,b} \in \mathbb{G}_{p_3}$.

Then we have the following easy observation.

1. For every $\mathbf{y} \in \{0,1,*\}^l$, the distributions *Genkey*(Msk, **y**) and *Genkey*($Msk^{'}$, **y**) are identical.

2. Similarly, for every $\mathbf{x} \in \{0,1\}^l$, the distributions *Encrypt*(Pp, **x**) and *Encrypt*($Pp^{'}$, **x**) are identical.

Security for Hierarchical HVE

The security proof of Hierarchical HVE is based on Assumption 3 and 5. For a probabilistic polynomial-time adversary $\mathcal{A}$ which makes q $\mathbf{Reveal}$ queries for , the proof of security will contain a sequence of (q + 2) games $\mathbf{Game_{Real}}, \mathbf{Game_0}, \ldots, \mathbf{Game_q}$ between a probabilistic polynomial time Adversary $\mathcal{A}$ and a Challenger $\mathcal{C}$.

$\mathbf{Game}_{Real}$ consists of the $\mathbf{Setup}$ Phase followed by a $\mathbf{Query}$ Phase. The $\mathbf{Query}$ Phase consists of several Key Queries and one Challenge Construction Query. We stress that the Challenge Construction Query is not necessarily the last query of the Query Phase. More precisely, we have the following game.

Setup. $\mathcal{C}$ runs the Setup algorithm on input ($1^\lambda, 1^l$) (given in unary) to generate public pa-

rameters Pp and master secret key Msk. $\mathcal{C}$ starts the interaction with $\mathcal{A}$ on input Pp.

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**. To make a Create query, A specifies a vector $\overrightarrow{y} \in \{0,1,*\}^l$. In response, $\mathcal{C}$ creates a key for $\overrightarrow{y}$ by running the KeyGen algorithm on input Msk and $\overrightarrow{y}$. $\mathcal{C}$ adds this key to the set S and gives $\mathcal{A}$ only a reference to it, not the actual key.

• **Delegate**. To make a Delegate query, $\mathcal{A}$ specifies a reference to a key $Sk_{\overrightarrow{y}}$ in the set S and a vector $\overrightarrow{w} \in \{0,1,\}^l$ such that $\overrightarrow{w} \prec \overrightarrow{y}$. In response, $\mathcal{C}$ makes a key for $\overrightarrow{w}$ by executing the Delegate algorithm on input Pp, $Sk_{\overrightarrow{y}}$, $\overrightarrow{y}$ and $\overrightarrow{w}$. $\mathcal{C}$ adds this key to the set S and again gives $\mathcal{A}$ only a reference to it, not the actual key.

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

Challenge Construction. To make a Challenge Construction query, $\mathcal{C}$ specifies a pair $x_0, x_1 \in \{0,1\}^l$. $\mathcal{C}$ answers by picking random $\eta \in \{0,1\}$ and returning Encrypt(Pp, $\overrightarrow{x_\eta}$).

At the end of the game, $\mathcal{A}$ outputs a guess $\eta^{'}$ for η. We say that $\mathcal{A}$ wins if $\eta^{'}$= η and for all $\overrightarrow{y}$ for which $\mathcal{A}$ has seen a secret key, it holds that **Match**$(x_0, \overrightarrow{y})$ = **Match**$(x_1, \overrightarrow{y})$ = 0. The advantage $Adv^{\mathcal{A}}_{HHVE}(\lambda)$ of $\mathcal{A}$ is defined to be the probability that $\mathcal{A}$ wins the game minus 1/2. We are now ready for the following definition.

Definition 4. A Hierarchical Hidden Vector Encryption scheme is secure if for all probabilistic polynomial time adversaries $\mathcal{A}$, we have that $Adv^{\mathcal{A}}_{HHVE}(\lambda)$ is a negligible function of λ.

$\mathbf{Game}_0$ is almost same as $\mathbf{Game}_{Real}$ except that $\mathcal{C}$ uses g_2 instead of g_1 to construct the public parameters $\mathbf{Pp}$ given to $\mathcal{A}$. Specifically $\mathcal{C}$ constructs,

$\mathbf{Pp}$= $[N, g_3, g_4, g_1\dot{R}, (T_{i,b})_{i\in[l],b\in 0,1}]$, $\mathbf{Pp}^{'}$= $[N, g_3, g_4, g_1\dot{R}(T^{'}_{i,b})_{i\in[l],b\in 0,1}]$ and

$\mathbf{Msk}$ = $[g_{12}, g_4, (t_{i,b})_{i\in[l],b\in\{0,1\}}]$

where $\mathcal{C}$ 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 $g_{12} = g_1g_2$. For each i $\in [l]$ and b $\in \{0, 1\}$, $\mathcal{C}$ chooses random $t_{i,b} \in \mathbb{Z}_N$ and $R_{i,b} \in \mathbb{G}_{p_3}$ and sets $T_{i,b} = g_2^{t_{i,b}}R_{i,b}$ and $T^{'}_{i,b} = g_1^{t_{i,b}}R_{i,b}$.

$\mathbf{Game}_k$ for $1 \leq k \leq q$ is almost equivalent to $\mathbf{Game}_0$ except among the The first k Reveal queries issued by $\mathcal{A}$ are instead answered by $\mathcal{C}$ by returning keys whose $\mathbb{G}_{\shortmid\nvDash}$ parts are random. All remaining Reveal queries are answered like in $\mathbf{Game_0}$. We stress that the $\mathbb{G}_{\shortmid\nvDash}$ parts of all answers are like in $\mathbf{Game_0}$. More precisely, the Key Query 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 R_v equal to 0.

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

• $\mathbf{Delegate}(v^{'}, \overrightarrow{w})$: For Delegate key query on vector w, C increments v and adds the tuple $(v, \overrightarrow{w}, v^{'})$ to the set S. $\mathcal{C}$ returns v to $\mathcal{A}$.

• $\mathbf{Reveal}(v^{'})$: Suppose entry $v^{'}$ in S refers to key $Sk_{\overrightarrow{w}}$ which is the the result of a delegation path $\overrightarrow{w} = \overrightarrow{w_0} \prec \overrightarrow{w_1}... \prec \overrightarrow{w_n} = \overrightarrow{y}$ of length $n \geq 0$ starting from key $Sk_{\overrightarrow{y}}$ created as result of the $v^{''}$ -th Create key query.

C chooses random $z \in \mathbb{Z}_N$ and, for each $i \in [l]$, random $c_i \in \mathbb{Z}_N$ and $R_i \in \mathbb{G}_{p_4}$. Moreover for each $i \notin S_{\overrightarrow{w}}$ and b $\in \{0, 1\}$, $\mathcal{C}$ chooses random $R_{i,b} \in \mathbb{G}_{p_4}$.

$\mathcal{C}$ increments R_v. If $R_v \leq k$ then for i $\in [l]$ $\mathcal{C}$ sets

$$Y_i = \begin{cases} g_1^{c_i} g_2^{\frac{za_{v'',i}}{t_{i,w_i}}} R_i & for\ \ i \in S_{\overrightarrow{w}} \\ g_1^{c_i} g_2^{za_{v'',i}} R_i & for\ \ i \notin S_{\overrightarrow{w}} \end{cases}$$

and for each i $i \notin S_{\overrightarrow{w}}$ and for b $\in \{0,1\}$ $\mathcal{C}$ sets

$D_{i,b} = g_1^{c_i} g_2^{\frac{za_{v''_{,i}} t_{i,w_i}}{R}}{}_{i,b}$

If instead $R_v > k$ then for each $i \in [l]$ $\mathcal{C}$ sets

$$Y_i = \begin{cases} g_{12}^{\frac{za_{v''_{,i}}}{t_{i,w_i}}} R_i & for\ \ i \in S_{\overrightarrow{w}} \\ g_{12}^{za_{v''_{,i}}} R_i & for\ \ i \notin S_{\overrightarrow{w}} \end{cases}$$

and for each i $i \notin S_{\overrightarrow{w}}$ and for b $\in \{0,1\}$ $\mathcal{C}$ sets

$D_{i,b} = g_{12}^{\frac{za_{v''_{,i}} t_{i,w_i}}{R}}{}_{i,b}$

Finally $\mathcal{C}$ returns the key $Sk_{\overrightarrow{w}}$ consisting of the Y_i's and the $D_{i,b}$'s

The security Proof is based on the follwing two lemmas.

Lemma 3.5. Suppose there exists a PPT algorithm $\mathcal{A}$ such that $Adv^{\mathcal{A}}_{Game_{Real}} - Adv^{\mathcal{A}}_{Game_0}$ = ϵ , for $1 \leq k \leq q$. Then, there exists a PPT algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 3.

Lemma 3.6. Suppose there exists a PPT algorithm $\mathcal{A}$ such that $Adv^{\mathcal{A}}_{Game_{k-1}} - Adv^{\mathcal{A}}_{Game_k}$ = ϵ , for $1 \leq k \leq q$. Then, there exists a PPT algorithm $\mathcal{B}$ with advantage atleast $\frac{\epsilon}{2l}$ in breaking Assumption 5.

Following the Lemmas it has been proved that the $\mathbf{Game_q}$ has no advantage i.e, $Adv^{\mathcal{A}}_{Game_q}$ = 0 which proves the following Theorem.

Theorem 3.5.3 *If Assumptions 3 and 5 hold, then the HHVE scheme is secure.*

Chapter 4

An Improved Security Proof for Hierarchical HVE Scheme

The previous definitions for security of Hierarchical IBE (HIBE) are incomplete which makes bulding delegation into encryption systems difficult. In the previous definitions of HIBE security, the attacker plays a game where he receives all his private key queries directly from the HIBE authority; Hence, this does not accurately model an adversary's view in a real system because an adversary might get the private key **ISI:MTech:Avik** directly from an authority or might choose to get it from a user with the key **ISI:Mtech** in a real system. Beside this, private keys received directly from the authority and delegated private keys may have different distribution or forms. For example, in the Gentry and Silverberg [10] and Boneh and Boyen HIBE [4] schemes if a HIBE private key of depth l is received directly from an authority, the authority will create l newly random elements of $\mathbb{Z}_p^*$ in creating the key; however, if the key is generated by another user, only one new degree of randomness will be added and the rest will be in common with the previous key. Hence in the security game, delegated privte key and private keys from authority may have different distribution and we should not assume that delegated keys have the same distribution as keys directly computed by the authority.

Alison Lewko et al. [15] first pointed the difficulty in and created a general framework and definitions for delegation in predicate encryption systems. To do this a general definition has been proposed that accounts for how predicate capabilities are created. In particular, the definition allows for the adversary to make queries both for capabilities that are created by an

authority and for capabilities delegated by users. The adversary may then ask for some subset of these capabilities to be revealed to him.

In the security for delegation in predicate encryption systems a query security game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ has been described. This game formally captures the notion that the tokens reveal no unintended information about the plaintext. $\mathcal{A}$ asks the challenger for a number of tokens. For each queried token, $\mathcal{A}$ gets to specify its path of derivation: whether the token is directly generated by the root authority, or delegated from another token. If the token is delegated, the adversary also gets to specify from which token it is delegated. The game proceeds as follows:

Setup. $\mathcal{C}$ runs the Setup algorithm and gives the adversary the public key PP.

Query1. $\mathcal{A}$ adaptively makes a polynomial number of queries of the following types:

• **Create token**. $\mathcal{A}$ asks the challenger to create a token for a set of functions $\mathcal{G} \subseteq \mathcal{F}$. $\mathcal{C}$ creates a token for $\mathcal{G}$ without giving it to $\mathcal{A}$.

• **Create delegated token**. $\mathcal{A}$ specifies a token for function family $\mathcal{G}$ that has already been created, and asks the challenger to perform a delegation operation to create a child token for $\mathcal{G}'$ $\subseteq \mathcal{G}$. $\mathcal{C}$ computes the child token without releasing it to $\mathcal{A}$.

• **Reveal token**. $\mathcal{A}$ asks the challenger to reveal an already-created token for function family $\mathcal{G}$.

Challenge. $\mathcal{A}$ outputs two strings $X_0^*, X_1^* \in \{0,1\}^l$ subject to the following constraint: For any token revealed to the adversary in the **Query1** stage, let $\mathcal{G}$ denote the function family corresponding to this token. For all $f \in \mathcal{G}$, f (X_0^*) = f (X_1^*).

Next, $\mathcal{C}$ flips a random coin b and encrypts X_b^* . It returns the ciphertext to $\mathcal{A}$.

Query2. Repeat the **Query1** stage. All tokens revealed in this stage must satisfy the same condition as above.

Guess. $\mathcal{A}$ outputs a guess b' of b. The advantage of $\mathcal{A}$ in the above game is defined to be

$Adv_{\mathcal{A}} = |Pr[b = b'] - 1/2|$.

Definition 4 : We say that a delegatable predicate encryption system is secure if for all polynomial time adversaries $\mathcal{A}$ attacking the system, its advantage $Adv_{\mathcal{A}}$ is a negligible function of λ.

Clearly this security definition is complete in the sense that in the query phase, $\mathcal{A}$ gets to specify, for each queried token, its path of derivation: whether the token is generated by the root authority, or from whom the token has been delegated. In prior work on delegation in identity-based encryption systems (e.g., Hierarchical Identity-Based Encryption (HIBE) [9], Anonymous Hierarchical Identity-Based Encryption (AHIBE) [10] [5]), the security game was under-specified. In these definitions, the adversary did not get to specify from whom each queried token is delegated.One way to deal with this is to create systems where all tokens are generated from the same probability distribution. But under this security definition, the delegated token need not be picked from the same probability distribution as the nondelegated tokens.

Now in the HHVE scheme given in [7] we observed that actual key generated by KeyGen algorithm(Run by PKG) are in the same distribution with the key generatedby running a Delegate(Run by user) algorithm. But Angelo De Caro et al. in [7] has proved the security of their HHVE scheme by using Delegate(Run by user) algorithm in the key query phase which is redundant. Hence we have given a new security proof for the HHVE scheme which removes the flaw of the previous security proof given in [7] by removing the importance of Delegate algorithm in the Key Query phase and the Delegate algorithm just acts like Create algorithm of the Key Query phase. Since in the Lemma 4.1(Section 4.1.1) it has been proved that actual key generated by KeyGen algorithm(Run by PKG) are in the same distribution with the key generated by running a Delegate(Run by user) algorithm, so the adversary $\mathcal{A}$ can not distinguish between the keys(Generated by delegate and create respectively). Hence uselessness of Delegate algorithm in HHVE scheme is verified. Hence our new proof becomes much more simpler.

Angelo de caro [7] has proved the IND-CPA security of the HHVE scheme under the assumption 3 and 5(Section 3.5.2). But we have proved the security of the scheme under the assumptions 3 and 4(Section 3.5.2) where assumption 4 is weaker than assumption 5.

4.1 Improved Security Proof for HHVE Scheme

We prove the security of Hierarchical HVE scheme by introducing a sequence of games $\mathbf{Game}_{Real}$, $\mathbf{Game}_{Real'}$, $\mathbf{Game}_0$, $\mathbf{Game}_1$, . . . $\mathbf{Game}_q$ i.e, we are constructing a sequencce of (q+3) games.

Our security definition requires that no PPT adversary $\mathcal{A}$ has non-negligible advantage over $\frac{1}{2}$ in game $\mathbf{Game}_{Real}$ against a challenger $\mathcal{C}$. $\mathbf{Game}_{Real}$ consists of the $\mathbf{Setup}$ Phase followed by a $\mathbf{Query}$ Phase. The $\mathbf{Query}$ Phase consists of several Key Queries and one Challenge Construction Query. We stress that the Challenge Construction Query is not necessarily the last query of the Query Phase. More precisely, we have the following game.

Setup. $\mathcal{C}$ runs the Setup algorithm on input ($1^\lambda, 1^l$) (given in unary) to generate public parameters Pp and master secret key Msk. $\mathcal{C}$ starts the interaction with $\mathcal{A}$ on input Pp.

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 $\emptyset$.

• $\mathbf{Create}$. To make a Create query, A specifies a vector $\overrightarrow{y} \in \{0, 1, *\}^l$. In response, $\mathcal{C}$ creates a key for $\overrightarrow{y}$ by running the KeyGen algorithm on input Msk and $\overrightarrow{y}$. $\mathcal{C}$ adds this key to the set S and gives $\mathcal{A}$ only a reference to it, not the actual key.

• $\mathbf{Delegate}$. To make a Delegate query, $\mathcal{A}$ specifies a reference to a key $Sk_{\overrightarrow{y}}$ in the set S and a vector $\overrightarrow{w} \in \{0, 1, \}^l$ such that $\overrightarrow{w} \prec \overrightarrow{y}$. In response, $\mathcal{C}$ makes a key for $\overrightarrow{w}$ by executing the Delegate algorithm on input Pp, $Sk_{\overrightarrow{y}}$, $\overrightarrow{y}$ and $\overrightarrow{w}$. $\mathcal{C}$ adds this key to the set S and again gives $\mathcal{A}$ only a reference to it, not the actual key.

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

Challenge Construction. To make a Challenge Construction query, $\mathcal{C}$ specifies a pair $x_0, x_1 \in$

$\{0,1\}^l$. $\mathcal{C}$ answers by picking random $\eta \in \{0,1\}$ and returning Encrypt(Pp, $\overrightarrow{x_\eta}$).

At the end of the game, $\mathcal{A}$ outputs a guess $\eta^{'}$ for η. We say that $\mathcal{A}$ wins if $\eta^{'}$= η and for all $\overrightarrow{y}$ for which $\mathcal{A}$ has seen a secret key, it holds that **Match**$(x_0, \overrightarrow{y})$ = **Match**$(x_1, \overrightarrow{y})$ = 0. The advantage $Adv^{\mathcal{A}}_{HHVE}(\lambda)$ of $\mathcal{A}$ is defined to be the probability that $\mathcal{A}$ wins the game minus 1/2. We are now ready for the following definition.

Definition 1. A Hierarchical Hidden Vector Encryption scheme is secure if for all probabilistic polynomial time adversaries $\mathcal{A}$, we have that $Adv^{\mathcal{A}}_{HHVE}(\lambda)$ is a negligible function of λ.

$\mathbf{Game}_{Real'}$ is almost same as $\mathbf{Game}_{Real}$ except the $\mathbf{Reveal}$ algorithm in the $\mathbf{Query}$ phase acts differently . In stead of giving Delegated Key in some specific query by $\mathcal{A}$ for some $\overrightarrow{w}$, where $\overrightarrow{w} \prec \overrightarrow{y}$ for some existing $Sk_{\overrightarrow{y}}$ stored in S, the Algorithm directly gives Secret key $Sk_{\overrightarrow{w}}$ by using $\mathbf{KeyGen}$ algorithm.

$\mathbf{Game}_0$ is almost same as $\mathbf{Game}_{Real'}$ except in $\mathbf{Game_0}$ the challenger $\mathcal{C}$ gives $\mathcal{A}$ the Public Parameters as

$\mathbf{Pp} = [\mathbf{N}, \mathbf{g_3}, \mathbf{g_4}, \mathbf{g_2}.\mathbf{R}, (\mathbf{T_{i,b}})_{\mathbf{i}\in[\mathbf{l}],\mathbf{b}\in\{\mathbf{0},\mathbf{1}\}}]$ where $T_{i,b} = g_2^{t_{i,b}} Z_{i,b}$.

$\mathcal{C}$ also constructs $\mathbf{Pk}^{'}$ and $\mathbf{Msk}$ where

$\mathbf{Pp}^{'} = [\mathbf{N}, \mathbf{g_3}, \mathbf{g_4}, \mathbf{g_1}.\mathbf{R}, (\mathbf{T'_{i,b}})_{\mathbf{i}\in[\mathbf{l}],\mathbf{b}\in\{\mathbf{0},\mathbf{1}\}}]$ where $T'_{i,b} = g_1^{t_{i,b}} Z_{i,b}$.

$\mathbf{Msk} = [\mathbf{g_{12}}, (\mathbf{t'_{i,b}})_{\mathbf{i}\in[\mathbf{l}],\mathbf{b}\in\{\mathbf{0},\mathbf{1}\}}]$

$\mathbf{Game}_k$ for 1 $\leq$ k $\leq$ q is almost equivalent to $\mathbf{Game}_0$ except among the q key queries by $\mathcal{A}$, for the first k queries $\mathcal{C}$ sets the secret key components

$$Y_i = \begin{cases} g_1^{c_i} g_2^{\frac{a_i}{t_{i,y_i}}} R_i & for \ \ i \in S_{\overrightarrow{y}} \\ g_1^{c_i} g_2^{a_i} R_i & for \ \ i \notin S_{\overrightarrow{y}} \end{cases}$$

and $D_{i,b} = g_1^{c_i} g_2^{\frac{a_i}{t_{i,b}}} R_{i,b}$ for $i \notin S_{\overrightarrow{y}}$ and $b \in \{0,1\}$

and gives $Sk_{\overrightarrow{y}} = [(Y_i)_{i\in[l]}, (D_{i,b})_{i\notin S_{\overrightarrow{y}}, b\in\{0,1\}}]$ to $\mathcal{A}$

4.1.1 Proof of Indistinguishability of $\mathbf{Game_{Real}}$ and $\mathbf{Game_{Real'}}$

Lemma4.1 $\mathbf{Game_{Real}}$ and $\mathbf{Game_{Real'}}$ are indistinguishable

Proof: Let $\mathcal{C}$ be a challenger and $\mathcal{A}$ be an adversery.

Let us consider two vectors such that $\overrightarrow{y}$ and $\overrightarrow{w}$ such that $\overrightarrow{w} \prec \overrightarrow{y}$

Clearly we have to show that $Sk_{\overrightarrow{w}}$= Delegate(Pk, $Sk_{\overrightarrow{y}}$, $\overrightarrow{y}$, $\overrightarrow{w}$) in $\mathbf{Game_{Real}}$ and $Sk_{\overrightarrow{w}}$= Keygen(Msk, $\overrightarrow{w}$) in $\mathbf{Game_{Real'}}$ are from same distribution.

Clearly during Query phase of $\mathbf{Game_{Real}}$ $\mathcal{C}$ runs $\mathbf{Keygen}$ algorithm (If $\mathbf{Create}$ query is called) and produce

$$Y_{i'} = \begin{cases} g_{12}^{\frac{a_i}{t_{i,y_i}}} R_i & for\ \ i \in S_{\overrightarrow{y}} \\ g_{12}^{a_i} R_i & for\ \ i \notin S_{\overrightarrow{y}} \end{cases}$$

and

$D'_{i,b} = g_{12}^{\frac{a_i}{t_{i,b}}} R_{i,b}$ for $i \notin S_{\overrightarrow{y}}$ and $b \in \{0,1\}$

$\mathcal{C}$ runs $\mathbf{Delegation}$ algorithm (If $\mathbf{Delegate}$ query is called) and produce

$$Y_i = \begin{cases} {Y'_i}^z R_i & for\ \ i \in S_{\overrightarrow{y}} \\ {D'_{i,w_i}}^z R_i & for\ \ i \notin S_{\overrightarrow{y}} \end{cases}$$

and

$D'_{i,b}$ = ${D'_{i,b}}^z R_{i,b}$ for $i \notin S_{\overrightarrow{y}}$ and $b \in \{0,1\}$

the individual components of the keys are described in section 3.5.4
Clearly during Query phase of $\mathbf{Game_{Real'}}$ $\mathcal{C}$ runs $\mathbf{Keygen}$ algorithm and produce

$$Y_i = \begin{cases} g_{12}^{\frac{a_i}{t_{i,y_i}}} R_i & for\ \ i \in S_{\overrightarrow{y}} \\ g_{12}^{a_i} R_i & for\ \ i \notin S_{\overrightarrow{y}} \end{cases}$$

and

$D_{i,b} = g_{12}^{\frac{a_i}{t_{i,b}}} R_{i,b}$ for $i \notin S_{\overrightarrow{y}}$ and $b \in \{0,1\}$

This shows that $Sk_{\overrightarrow{w}}$= Delegate(Pp, $Sk_{\overrightarrow{y}}$, $\overrightarrow{y}$, $\overrightarrow{w}$) in $\mathbf{Game_{Real}}$ and $Sk_{\overrightarrow{w}}$= Keygen(Msk, $\overrightarrow{w}$) in $\mathbf{Game_{Real'}}$ are from same distribution.(Y_i 's are from $\mathbb{G}_{p_1p_2p_4}$ and $D_{i,b}$ for $i \notin S_{\overrightarrow{y}}$ and $b \in \{0,1\}$ are from $\mathbb{G}_{p_1p_2p_3}$

4.1.2 Proof of Indistinguishability of $\mathbf{Game_{Real'}}$ and $\mathbf{Game_0}$

Lemma 4.2. Suppose there exists a PPT algorithm $\mathcal{A}$ such that $Adv^{\mathcal{A}}_{Game_{Real'}} - Adv^{\mathcal{A}}_{Game_0}$ = ϵ . Then, there exists a PPT algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 3(section 3.5.2).

Proof: The proof is constructive. We prove the lemma by showing a PPT algorithm $\mathcal{B}$ which receives ($\mathcal{I}$, A_3, A_4, A_{13}, A_{12}, T) and, depending on the distribution of T , simulates $Game_{Real'}$ or $Game_0$ with $\mathcal{A}$.

Setup. $\mathcal{B}$ starts by constructing public parameters $\mathbf{Pp}$ and $\mathbf{Pp}^{'}$ in the following way. $\mathcal{B}$ sets $g_3 = A_3$, $g_{12} = A_{12}$, $g_4 = A_4$ and, for $i \in [l]$ and $b \in \{0,1\}$, $\mathcal{B}$ chooses random $t_{i,b} \in Z_N$ and sets $T_{i,b} = T^{t_{i,b}}$ and $T'_{i,b} = A_{13}^{t_{i,b}}$. Then $\mathcal{B}$ sets

$\mathbf{Pp}$ = $[N, g_3, g_4, T, (T'_{i,b})_{i\in[l],b\in\{0,1\}}]$, and

$\mathbf{Msk}$ = $[g_{12}, (t_{i,b})_{i\in[l],b\in\{0,1\}}]$

$\mathcal{B}$ constructs $\mathbf{Pp'}$ = $[N, g_3, g_4, A_{13}, (T'_{i,b})_{i\in[l],b\in\{0,1\}}$], and keeps it to itself for the Challenge phase and initiallize S = $\emptyset$ and starts the interaction with $\mathcal{A}$ on input $\mathbf{Pp}$.

$\mathbf{KeyQueryAnswering}$. $\mathcal{B}$ handles this phase in the following way

• **Create**. To make a Create query, $\mathcal{A}$ specifies a vector $\overrightarrow{y} \in \{0, 1, *\}^l$. In response, $\mathcal{B}$ creates a key for $\overrightarrow{y}$ by running the KeyGen algorithm on input Msk and $\overrightarrow{y}$. $\mathcal{B}$ adds this key to the set S and gives $\mathcal{A}$ only a reference to it, not the actual key.

•**Delegate**. In this phase $\mathcal{B}$ handles $\mathcal{A}$'s query just like in **Create** query phase. Clearly $\mathcal{A}$ specifies a reference to a key $Sk_{\overrightarrow{y}}$ in the set S and a vector $\overrightarrow{w} \in \{0, 1, \}^l$ such that $\overrightarrow{w} \prec \overrightarrow{y}$. $\mathcal{B}$ runs the KeyGen algorithm on input Msk and $\overrightarrow{w}$. $\mathcal{B}$ adds this key to the set S and gives $\mathcal{A}$ only a reference to it, not the actual key.

• **Reveal**. To make a Reveal query, $\mathcal{A}$ specifies an element of the set S. $\mathcal{B}$ gives the corresponding 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 the Delegate algorithm on the revealed key by itself.

$\mathbf{ChallengeConstruction}$. The challenge is created by $\mathcal{B}$ by picking one of the two vectors provided by $\mathcal{A}$, let us call it $\overrightarrow{x}$, and by encrypting it by running the Encrypt algorithm on input $\overrightarrow{x}$ and Pp' .

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

Now suppose that T $\in \mathbb{G}_{p_1p_3}$. Then T can be written as T = h_1h_3 for some $h_1 \in \mathbb{G}_{p_1}$ and

$h_3 \in \mathbb{G}_{p_3}$. So the public parameter $\mathbf{Pp}$ received by $\mathcal{A}$ in the interaction with $\mathcal{B}$ has the same distribution in $\mathbf{Game_{Real'}}$. Moreover, by writing A_{13} as $A_{13} = \hat{h}_1\hat{h}_3$ for $\hat{h}_1 \in \mathbb{G}_{p_1}$ and $\hat{h}_3 \in \mathbb{G}_{p_3}$, we notice that $\mathbf{Pp}$ and $\mathbf{Pp}'$ are as in the hypothesis of Remark, where 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 $\mathbf{Game_{Real'}}$. Therefore if T $\in \mathbb{G}_{p_1p_3}$, $\mathcal{B}$ simulates $\mathbf{Game_{Real'}}$ for A Similarly if T $\in \mathbb{G}_{p_2p_3}$, $\mathcal{B}$ simulates $\mathbf{Game_0}$ for $\mathcal{A}$. This concludes the proof of the lemma.

4.1.3 Proof of Indistinguishability of $\mathbf{Game_k}$ and $\mathbf{Game_{k-1}}$

Lemma 4.3. Suppose there exists a PPT algorithm $\mathcal{A}$ such that $Adv^{\mathcal{A}}_{Game_{k-1}} - Adv^{\mathcal{A}}_{Game_k}$ = ϵ , for $1 \leq k \leq q$. Then, there exists a PPT algorithm $\mathcal{B}$ with advantage atleast $\frac{\epsilon}{2l}$ in breaking Assumption 4(section 3.5.2).

Proof: The proof is constructive. We prove the lemma by showing a PPT algorithm $\mathcal{B}$ which receives ($\mathcal{I}, A_1, A_2, A_3, A_4, A_1^{\alpha}B_4, A_1^{\beta}C_4, T$) and, depending on the distribution of T , simulates $\mathbf{Game_{k-1}}$ or $\mathbf{Game_k}$ with $\mathcal{A}$.

$\mathcal{B}$ guesses the index j such that the j-th bit $w_j^{(k)}$ of the k-th *Reveal* query $\overrightarrow{w}(k)$ is different from * and different from x_j of the challenge vectors given by $\mathcal{A}$ that is used by $\mathcal{B}$ to construct the challenge ciphertext. Clearly such index j always exists and that the probability that $\mathcal{B}$ correctly guesses j and $w_j^{(k)}$ for which $x_j = 1 - w_j^{(k)}$ is at least $\frac{1}{(2l)}$. Notice that, if during the simulation this is not the case, then $\mathcal{B}$ aborts the simulation and fails. So we assume that $\mathcal{B}$'s initial guess is correct to prove the correctness of the simulation.

Setup. $\mathcal{B}$ sets $g_1 = A_1, g_2 = A_2, g_3 = A_3, g_4 = A_4$ and $g_{12} = A_1A_2$. $\mathcal{B}$ chooses random R $\in G_{p_3}$ and, for $i \in [l] \setminus \{j\}$ and b $\in \{0, 1\}$, $\mathcal{B}$ chooses random $t_{i,b} \in Z_N$ and $Z_{i,b} \in G_{p_3}$. Then $\mathcal{B}$ sets $T_{i,b} = g_2^{t_{i,b}} Z_{i,b}$.
Moreover, $\mathcal{B}$ chooses random $t_{j,x_j} \in Z_N$, $Z_{i,x_j} \in G_{p_3}, r_{j,y_j^{(k)}} \in Z_N, Z_{j,y_j^{(k)}} \in G_{p_3}$. $\mathcal{B}$ then sets $T_{j,x_j} = g_2^{t_{j,x_j}} Z_{j,x_j}$ and $T_{j,y_j^{(k)}} = g_2^{r_{j,y_j^{(k)}}} Z_{j,y_j^{(k)}}$

$\mathcal{B}$ sets Public Parameters as $\mathbf{Pp}$ = $[N, g_3, g_4, g_2R, (T_{i,b})_{i\in[l],b\in\{0,1\}}]$ and give it to $\mathcal{A}$

$\mathcal{B}$ also set master Secret Key $\mathbf{Msk}$ = $[g_{12}, (t_{i,b})_{i\in[l],b\in\{0,1\}}, t_{i,x_j}, r_{j,y_j^{(k)}}]$

In addition for each $i \in [l] \setminus \{j\}, b \in \{0,1\}$ $\mathcal{B}$ chooses $Z'_{i,b} \in G_{p_3}$.

Then $\mathcal{B}$ sets $T'_{i,b} = g_1^{t_{i,b}} Z'_{i,b}$ and $T_{j,x_j} = g_1^{t_{j,x_j}} Z_{j,x_j}$ for $Z_{i,x_j} \in G_{p_3}$.

In answering Key Queries $\mathcal{B}$ will implicitly set $T_{j,y_j^{(k)}} = g_1^{\frac{1}{\beta}} Z'_{j,y_j^{(k)}}$ for $Z'_{j,y_j^{(k)}} \in G_{p_3}$
and constructs $\mathbf{Pp}'$ = $[N, g_3, g_4, g_1R, (T'_{i,b})_{i\in[l],b\in\{0,1\}}]$, and keeps it to itself for the Challenge phase and initiallize S = $\emptyset$ to start the interaction with $\mathcal{A}$ on input $\mathbf{Pp}$.

Key Query Answering. $\mathcal{B}$ handles the first (k-1) queries as follows.

Clearly as in the definition of $\mathbf{Game_k}$ the $\mathbf{Delegate}$ algorithm in the key query phase is same with the $\mathbf{Create}$ algorithm in the key query phase.

Suppose $\mathcal{A}$ requests the key for $\overrightarrow{y}$ (corresponding input vector).

For $i \in [l]$, $\mathcal{B}$ chooses $c_i \in Z_N$, $R_i \in G_{p_4}$, $a_i \in Z_N$, such that $\sum_{i\in[l]} a_i$ = 0 and for $i \notin S_{\overrightarrow{y}}$ and $b \in \{0,1\}$ $\mathcal{B}$ chooses $R_{i,b} \in G_{p_3}$.

For $i \in [l] \setminus \{j\}$ $\mathcal{B}$ sets

$$Y_i = \begin{cases} g_1^{c_i} g_2^{\frac{a_i}{t_{i,y_i}}} R_i & for\ \ i \in S_{\overrightarrow{y}} \\ g_1^{c_i} g_2^{a_i} R_i & for\ \ i \notin S_{\overrightarrow{y}} \end{cases}$$

and $D_{i,b} = g_1^{c_i} g_2^{\frac{a_i}{t_{i,b}}} R_{i,b}$ for $i \neq j$, $i \notin S_{\overrightarrow{y}}$ and $b \in \{0,1\}$

Moreover if $j \in S_{\overrightarrow{y}}$ then $\mathcal{B}$ sets

$$Y_j = \begin{cases} g_1^{c_j} g_2^{\frac{a_j}{r_{j,y_j^{(k)}}}} R_j & for \;\; y_j = y_j^{(k)} \\ g_1^{c_j} g_2^{\frac{a_j}{t_{j,x_j}}} R_i & for \;\; y_j = x_j \end{cases}$$

If $j \notin S_{\overrightarrow{y}}$ then $\mathcal{B}$ sets

$Y_j = g_1^{c_j} g_2^{a_j} R_j, D_{j,y_j^{(k)}} = g_1^{c_j} g_2^{\frac{a_j}{r_{j,y_j^{(k)}}}} R_{j,y_j^{(k)}}$ and $D_{j,x_j} = g_1^{c_j} g_2^{\frac{a_j}{t_{j,x_j}}} R_{j,x_j}$

The answer to the first $(k-1)$ queries has the same distribution in $\mathbf{Game_{k-1}}$ or $\mathbf{Game_k}$.

$\mathcal{B}$ handles the k^{th} query as follows.

Suppose $\mathcal{A}$ requests the key for $\overrightarrow{y}$ (Correspondong input vector).

Let h be an index such that $h \neq j$ and $y_h^{(k)} \neq *$

$\mathcal{B}$ chooses for $i \in [l] \setminus \{j, h\}$, $a_i \in Z_n$ randomly
$\mathcal{B}$ chooses for $i \in [l]$, $R_i \in \mathbb{G}_{p_4}$ randomly
$\mathcal{B}$ also chooses $R_{i,b} \in \mathbb{G}_{p_4}$for $i \neq j, h_i \in S_{\overrightarrow{y}^{(k)}}$, b $\in \{0,1\}$ randomly

For each $i \in [l] \setminus \{j, h\}$, $\mathcal{B}$ sets

$$Y_i = \begin{cases} g_{12}^{\frac{a_i}{t_{i,y_i^{(k)}}}} R_i & for \;\; i \in S_{\overrightarrow{y}^{(k)}} \\ g_{12}^{a_i} R_i & for \;\; i \notin S_{\overrightarrow{y}^{(k)}} \end{cases}$$

Moreover for $i \notin S_{\overrightarrow{y}^{(k)}}$ and b $\in \{0,1\}$, $\mathcal{B}$ sets

$D_{i,b} = g_{12}^{\frac{a_i}{t_{i,b}}} R_{i,b}$.

Let $s = \sum_{i \in [l] \setminus \{j,h\}} a_i$

$\mathcal{B}$ also sets

$Y_j = Tg_2^{\frac{a_j^{'}}{r_{j,y_j^{(k)}}}} R_j$ for some $a_j^{'} \in Z_N$, and

$Y_h = (A_1^{\alpha} B_4)^{-\frac{1}{t_{h,y_h^{(k)}}}} . g_1^{-\frac{s}{t_{h,y_h^{(k)}}}} . g_2^{-\frac{s+a_j^{'}}{t_{h,y_h^{(k)}}}} . R_h$

This terminates the description of how $\mathcal{B}$ handles the k-th $\mathbf{Reveal}$ query.

Suppose now that $T = A^{\alpha\beta} D_4$ and thus by our settings, we have that

$Y_j = g_1^{\alpha\beta} g_2^{\frac{a_j^{'}}{r_{j,y_j^{(k)}}}} (D_4 R_j)$

By the Chinese Remainder Theorem, there exists $a_j \in Z_N$ such that

$$a_j = \begin{cases} \alpha \; mod \; p_1 \\ a_j^{'} \; mod \; p_2 \end{cases}$$

and there exists $t_{j,y_j^{(k)}} \in Z_N$ such that,

$$t_{j,y_j^{(k)}} = \begin{cases} \frac{1}{\beta} \; mod \; p_1 \\ r_{j,y_j^{(k)}} \; mod \; p_2 \end{cases}$$

Clearly we can write Y_j and Y_h as

$Y_j = g_{12}^{\frac{a_j}{t_{j,y_j^{(k)}}}} R_j$, and

$Y_h = g_1^{\frac{-(\alpha+s)}{t_{h,y_h^{(k)}}}} . g_2^{\frac{-(s+a_j^{'})}{t_{h,y_h^{(k)}}}} . R_h$

$= g_{12}^{\frac{-(s+a_j^{'})}{t_{h,y_h^{(k)}}}} . R_h$

$= g_{12}^{\frac{+a_h}{t_{h,y_h^{(k)}}}} . R_h$ where, $a_h = -(a_j + s)$

Therefore the answer to the k-th query is distributed as $\mathbf{Game_{k-1}}$

If T is random in $\mathbb{G}_{p_1 p_4}$ then $\mathbb{G}_{p_1}$ part of the Y_i's are random and independent thus answer to the k-th query is distributed as $\mathbf{Game_k}$

Now $\mathcal{B}$ handles the last $(q-k)$ queries as follows.

Suppose $\mathcal{A}$ requests the key for $\overrightarrow{y}$

Now $\mathcal{B}$ chooses for $i \in [l]$, $R_i \in \mathbb{G}_{p_4}$ randomly

For each $i \in [l] \setminus \{j\}, b \in \{0,1\}$ $\mathcal{B}$ chooses $R_{i,b} \in G_{p_4}$

For $i \in [l]$, $\mathcal{B}$ chooses $a_i \in Z_N$, such that $\sum_{i \in [l]} a_i$ = 0

For $i \in [l] \setminus \{j\}$ $\mathcal{B}$ sets

$$Y_i = \begin{cases} g_{12}^{\frac{a_i}{t_{i,y_i}}} R_i & for \ \ i \in S_{\overrightarrow{y}} \\ g_{12}^{a_i} R_i & for \ \ i \notin S_{\overrightarrow{y}} \end{cases}$$

and

$D_{i,b} = g_{12}^{\frac{a_i}{t_{i,b}}} R_{i,b}$ for $i \neq j$ $i \notin S_{\overrightarrow{y}}$ and $b \in \{0,1\}$

For index j, $\mathcal{B}$ sets

$$Y_j = \begin{cases} (A_1^{\beta} C_4)^{a_j} g_2^{\frac{a_j}{r_{j,y_j^{(k)}}}} R_j & for \ \ y_j = y_j^{(k)} \\ g_{12}^{\frac{a_j}{t_{j,x_j}}} R_j & for \ \ y_j = x_j \\ g_{12}^{aj} R_j & for \ \ y_j \neq * \end{cases}$$

Finally if $y_j \neq *$

$$D_{j,y_j^{(k)}} = (A_1^{\beta} C_4)^{a_j} g_2^{\frac{a_j}{r_{j,y_j^{(k)}}}} R_{j,y_j^{(k)}}$$

$$D_{j,x_j} = g_{12}^{\frac{a_j}{t_{j,x_j}}} R_{j,x_j}$$

By the Chinese Remainder Theorem, there exists $t'_{j,y_j^{(k)}} \in Z_N$ such that

$$t'_{j,y_j^{(k)}} = \begin{cases} \frac{1}{\beta} \; mod \; p_1 \\ r_{j,y_j^{(k)}} \; mod \; p_2 \end{cases}$$

So Y_i can be written in the following format

$$Y_i = \begin{cases} g_{12}^{\frac{a_i}{t_{i,y_i}}} R'_i & for \;\; i \in S_{\overrightarrow{y}} \\ g_{12}^{a_i} R'_i & for \;\; i \notin S_{\overrightarrow{y}} \end{cases}$$

For some random $R'_i \in \mathbb{G}_{p_4}$.

we can therefore conclude that the answer provided by $\mathcal{B}$ has the same distribution as in $\mathbf{Game_k}$ and $\mathbf{Game_{k-1}}$.

Challenge construction. $\mathcal{B}$ creates the challenge ciphertext by running algorithm $\mathbf{Encrypt}$ on input one randomly chosen challenge vector $\overrightarrow{x}$ provide by $\mathcal{A}$ and public parameters $\mathbf{Pp}'$. Notice that under the assumption that $\mathcal{B}$ has correctly guessed $w_j^{(k)}$ we have that $x_j = w_j^{(k)}$, and this $\mathbf{Pp}'$ contains all the values needed for computing an encryption of $\overrightarrow{x}$. Therefore the challenge ciphertext is distributed exactly like in $\mathbf{Game_{k-1}}$ and $\mathbf{Game_k}$.

We observed that in $\mathbf{Game_q}$ the $\mathbb{G}_{p_1}$ part of the challenge ciphertext is the only one depending on η. In addition notice that the $g_1 R_3$ is the only component of the public parameters which contains a $\mathbb{G}_{p_1}$ part but it is independent from η. Thus, it gives no advantage to the adversary and moreover the answer to the key queries have random and independent $\mathbb{G}_{p_1}$ part. Therefore we can conclude that for all adversaries $\mathcal{A}$, $Adv_{\mathbf{game_q}^{\mathcal{A}}} = 0$. We have thus proved.

Theorem 4.1.1 *If Assumptions 3 and 4 holds then the HHVE scheme is secure.*

Chapter 5

Conclusions

In this work we have done a survey on Attribute Based Encryption and predicate Based Encryption and found a redundancy of the security proof in the HHVE scheme proposed in [7]. Hence we improved the security proof of the scheme and proposed a new security proof by removing the redundancy in the previous proof for the HHVE scheme . The previous security proof uses Delegate algorithm in the key query phase which is redundant because actual key generated by KeyGen algorithm(Run by PKG) are in the same distribution with the key generated by running a Delegate(Run by user) algorithm in the key query phase. Our proof removes the importance of Delegate algorithm in the Key Query phase thus makes the proof simpler. Moreover we uses a assumption 3 and 4 instead of assumption 3 and 5, where assumption 4 is weaker version of assumption 5. Hence we proved that the HHVE scheme is secure in much weaker assumption.

Bibliography

[1] Michel Abdalla:Searchable Encryption Revisited. Crypto 2005

[2] Dan Boneh and Brent Waters: Conjunctive, subset, and range queries on encrypted data. .In *Salil Vadhan, editor, Theory of Cryptography Conference (TCC) 2007 ,LNCS, volume 4392* pages 535–554. Springer, 2007

[3] D. Boneh, E.-J. Goh, and K. Nissim : Evaluating 2-DNF Formulas on Ciphertexts. In *Advances in Cryptology — Eurocrypt 2005* Springer LNCS 3378 (2005),pages 325–341.

[4] Dan Boneh and Xavier Boyen: Efficient selective-ID secure Identity Based encryptions without random oracles in *Eurocrypt, 2004*.

[5] Xavier Boyen and Brent Waters : Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles) in *Crypto, 2006*.

[6] Ran Canetti, Shai Halevi, Jonathan Katz: Chosen-Ciphertext Security from Identity-Based Encryption. Eurocrypt 2004.

[7] Angelo De Caro, Vincenzo Iovino and Giuseppe Persiano : Efficient Fully Secure (Hierarchical) Predicate Encryption for Conjunctions, Disjunctions and k-CNF/DNF formulae Cryptology ePrint Archive 2010.

[8] Ling Cheung and Calvin Newport: Provably Secure Ciphertext Policy ABE in *ACM Conference on Computer and Communications Security , pages 456-465, 2007*.

[9] Ling Cheung and Craig Gentry and Alice Silverberg : Hierarchical ID-Based Cryptography in *Eurocrypt '05, LNCS 3493, pages. 440-456, 2005* .

[10] Craig Gentry and Alice Silverberg: Hierarchical id-based cryptography in *Asiacrypt, 2002*

[11] Vipul Goyal, Omkant Pandey, Amit Sahai and Brent Waters: Attribute-Based Encryption for Fine Grained Access Con- trol of Encrypted Data . Advances in *CCS '06 Proceedings of the 13th ACM conference on Computer and communications security*

[12] Vipul Goyal, Abhishek Jain, Omkant Pandey and Amit Sahai: Bounded Ciphertext Policy Attribute Based Encryption. ICALP (2) 2008

[13] Fujisaki E, Okamoto T : A Chosen-Cipher Secure Encryption Scheme Tightly as Secure as Factoring . in *IEICE Transactions on Fundamentals E84- A(1):* Pages 179-187

[14] Jonathan Katz, Amit Sahai and Brent Waters: Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. Eurocrypt 2008

[15] Allison Lewko, Amit Sahai, Tatsuaki Okamoto, Katsuyuki Takashima and Brent Waters: Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. Simulatable Adaptive Oblivious Transfer. in *Eurocrypt, 2010*.

[16] Amit Sahai and Brent Waters: Fuzzy Identity-Based Encryption Advances in *Eurocrypt'2005, Volume 3494 of Lecture Notes in Computer Science,* pages 457-473. Springer-Verlag, 2005.

[17] Amit Sahai, John Bethencourt and Brent Waters: Ciphertext Policy Attribute Based Encryption. Eurocrypt 2007.

[18] Adi Shamir: Identity-Based Cryptosystems and Signature Schemes. *Crypto 1984, LNCS 7,* pages 47-53. Springer-Verlag, 1984.

.