A New Functional Encryption for Multidimensional Range Query[⋆]

Jia Xu[1], Ee-Chien Chang[2], and Jianying Zhou[3]

[1] Singapore Telecommunications Limited
jia.xu@singtel.com
[2] National University of Singapore
changec@comp.nus.edu.sg
[3] Singapore University of Technology and Design
jianying_zhou@sutd.edu.sg

Abstract. Functional encryption, which emerges in the community recently, is a generalized concept of traditional encryption (e.g. RSA and AES). In traditional encryption scheme, decrypting a ciphertext with a correct decryption key will output the original plaintext associated to the ciphertext. In contrast, in functional encryption scheme, decrypting a ciphertext with a correct decryption key will output a value that is derived from both the plaintext and the decryption key, and the decryption output would change when different correct decryption key is used to decrypt the same ciphertext. We propose a new functional encryption scheme for multidimensional range query. Given a ciphertext that is the encryption of some secret plaintext under a public attribute (a multidimensional point), and a decryption key corresponding to a query range and a function key. If the public attribute point is within the query range, a user is able to decrypt the ciphertext with the decryption key to obtain a value, which is the output of a pre-defined *one-way* function with the secret plaintext and the function key as input. In comparison, in previous functional encryption for range query, a decryption will simply output the original secret plaintext when the attribute point is within the query range.

Keywords: Functional Encryption, Multidimensional Range Query, Polymorphic Property

1 Introduction

The concept of functional encryption emerges recently, as a generalization of traditional encryption. Informally, in traditional encryption scheme (e.g. public key cipher like RSA and private key cipher like AES), decrypting a ciphertext CT of a secret plaintext Msg with correct decryption key will output the original plaintext Msg. In a functional encryption scheme, decrypting a ciphertext CT with "correct decryption key" SK_k will obtain only a function value $f(k, \mathsf{Msg})$ of the plaintext Msg and the function key k, and nothing more. It will be more interesting when the function f is one-way, such that the original plaintext Msg remains secret after several function values $f(k_j, \mathsf{Msg})$'s for different function keys k_j are revealed.

To the best of our knowledge, almost all previous instances of functional encryption schemes (for example, attribute-based encryption or predicate encryption) implements a functionality F of

[⋆] This work was partly supported by SUTD start-up research grant SRG-ISTD-2017-124.

the following type:

$$F(k, (\mathbf{x}, \mathsf{Msg})) = \begin{cases} \mathsf{Msg} & \text{(if } \textsc{Predicate}(\mathbf{x}, k) = \texttt{True}\text{);} \\ \perp & \text{(otherwise)} \end{cases} \tag{1}$$

where $\textsc{Predicate}$ is pre-defined. In this paper, we are interested in a more general functionality:

$$F(k, (\mathbf{x}, \mathsf{Msg})) = \begin{cases} f(k, \mathsf{Msg}) & \text{(if } \textsc{Predicate}(\mathbf{x}, k) = \texttt{True}\text{);} \\ \perp & \text{(otherwise)} \end{cases} \tag{2}$$

where f is some one-way function. Few works have been devoted to the latter type of functionality (Eq (2)). Very recently, Gorbunov *et al.* [1] proposed a function encryption method for any multi-variable polynomial function, using Secure Multi-party Computation. The supported functionality belongs to the latter style (Eq (2)). In this paper, we will propose a more efficient functional encryption scheme which implements functionality in Equation (2) for a particular one-way function f (defined later) with $\textsc{Predicate}$ replaced by multidimensional range query, using a novel technique.

1.1 Overview of Our Technique

We observe that some (HIBE) encryption scheme ($\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$), e.g. BBG HIBE scheme [2], satisfies a *polymorphic property*: From a pair of keys $(pk, sk) \in \mathsf{KeyGen}(1^\kappa)$, a plaintext M, an identity id, and a random coin r, one can efficiently find multiple tuples $(pk_j, sk_j, M_j, r_j), 1 \leq j \leq n$, such that for any $1 \leq j \leq n$, $(pk_j, sk_j) \in \mathsf{KeyGen}(1^\kappa)$ is a valid key pair and

$$\mathsf{Enc}_{pk}(\mathsf{id}, M; r) = \mathsf{CT} = \mathsf{Enc}_{pk_j}(\mathsf{id}, M_j; r_j).$$

From the opposite point of view, a ciphertext CT can be decrypted into value M_j using the decryption key sk_j, $1 \leq j \leq n$. We can view these decrypted values M_j's as a function of the original plaintext M which is used to produce the ciphertext CT, i.e. decrypting CT using decryption key sk_j will generate the function value $f(j, M) := M_j$ of the plaintext M. Hence, such polymorphic property may lead to a new way to construct functional encryption schemes [3,4,5,6].

1.2 Application

Besides the theoretical merit as an example of a new kind of functional encryption paradigm, our proposed scheme can also be used to authenticate multidimensional range queries. Here we give a brief description in a nutshell: A data owner encrypts his multidimensional data points $(\mathsf{Msg}, \mathbf{x})$ (e.g. netflow, log, or sensor data in a cyber-physical system) using our functional encryption scheme, and outsources all ciphertexts to a cloud. Later, the data owner could choose a multidimensional query range $\mathbf{R}$ and a nonce [4] ρ, and sends a delegation key w.r.t. $(\mathbf{R}, \rho)$ to the cloud. Then the cloud tries to decrypt each ciphertext using this delegation key. If the corresponding point $\mathbf{x}$ is within the query range $\mathbf{R}$, then decryption will succeed and the cloud is able to obtain a one-way function value $f(\rho, \mathsf{Msg})$, which could serve as a proof that $\mathbf{x} \in \mathbf{R}$. The cloud could find and count all ciphertexts of data points within the query range, and sends corresponding proofs to the data owner. This is the basic idea how our proposed scheme can be used to authenticate multidimensional count queries. How to aggregate all individual proofs to reduce total proof size using some homomorphism, and how to prevent miss-counting or double counting, requires other non-trivial techniques. More details are provided in our technical report [7].

[4] Here the nonce ρ is crucial to prevent cloud from abusing delegation keys across different queries.

1.3 Contribution

- We propose a functional encryption scheme, by exploiting a special property (we call it "polymorphic property") of the BBG HIBE scheme [2]. Under this functional encryption scheme, given a secret message Msg and a public identity $\boldsymbol{x}$, which is a d-dimensional point in domain $[1, \mathcal{Z}]^d$ where system parameter $\mathcal{Z}$ is an integer, a ciphertext can be generated using the *private*[5] key. A decryption key w.r.t. a d-dimensional rectangular range $\mathbf{R}$ and a random nonce ρ can also be derived from the private key. With this decryption key and the ciphertext for message Msg under identity $\boldsymbol{x}$, the decryption algorithm will output $\Omega^{\rho \cdot \mathsf{Msg}}$ iff $\boldsymbol{x} \in \mathbf{R}$, where Ω is a part of key of the functional encryption scheme. The size[6] of a public/private key is in $O(1)$, the size of a ciphertext is in $O(d)$, and the size of a decryption key is in $O(d \log^2 \mathcal{Z})$. The application of our functional encryption scheme in authenticating multidimensional range queries [7] is demonstrated in the technical report [7].
- We define $\mathsf{weak\text{-}IND\text{-}sID\text{-}CPA}$ security following the $\mathsf{IND\text{-}sID\text{-}CPA}$ security formulation given by Boneh *et al.* [2]. We prove that the proposed functional encryption scheme is $\mathsf{weak\text{-}IND\text{-}sID\text{-}CPA}$ secure (as defined in Section 3.3), if BBG HIBE scheme [2] is $\mathsf{IND\text{-}sID\text{-}CPA}$ secure (See Theorem 2).

1.4 Organization

The rest of this paper is organized as below. Section 2 reviews related works. Section 3 constructs a new functional encryption scheme for multidimensional range query, and Section 4 presents the security formulation and analyzes the correctness and security of the proposed scheme. At the end, Section 5 concludes this paper.

2 Related Works

Functional encryption [3,4,5,6,8,1,9,10,11] is a new and more general notion to capture all of previous public encryption (e.g. RSA), private encryption (e.g. AES), identity based encryption (e.g. [12]), attribute-based encryption (e.g. [13]), and predicate encryption (e.g. [14]). Some works [3,4,9] aimed to formulate the security of generic functional encryption, some [11,8,1] constructed functional encryption for a somewhat generic class of functionalities, and some [10] analyzed the lower bound of functional encryption scheme.

In particular to functional encryption supporting multidimensional range query, Shi *et al.* [15] proposed a predicate encryption scheme called MRQED (Multi-dimensional Range Query over Encrypted Data). Under their scheme, given a message and an identity, which is a d-dimensional point, a ciphertext can be generated. A short decryption key for a d-dimensional rectangular range can be generated from the master secret key. From this decryption key and the ciphertext, the original message can be decrypted, iff the identity point associated with the ciphertext is within the

[5] Unlike [3,4], our functional encryption scheme is a symmetric key encryption system. However, in the case that dimension $d = 1$, our functional encryption scheme can become a public key encryption scheme.

[6] Since the private key contains $O(d)$ random elements from $\mathbb{Z}_p^*$ and $O(\ell)$ random elements from $\widetilde{\mathbb{G}}$, its size can be reduced from $O(\ell + d)$ to $O(1)$ (precisely, $O(1)$ number of secret seeds, and each seed with length equal to the security parameter κ), using a pseudorandom function.

[7] Note that authenticating multidimensional range queries is very useful when outsourcing multidimensional dataset (e.g. network flow or log data in cyber physical system) to cloud

query range. There is a subtle but crucial difference between MRQED scheme and our implementation of functional encryption scheme: After a successful decryption, MRQED scheme reveals the message, whereas our functional encryption scheme reveals only a function value of the message. Precisely, the functionalities supported by MRQED [15] and this paper are given in Equation (3) and Equation (4), respectively.

$$\text{MRQED:} \qquad F(k=(\mathbf{R}),(\mathbf{x},\mathsf{Msg})) = \begin{cases} \mathsf{Msg} & (\text{if } \mathbf{x}\in\mathbf{R}); \\ \perp & (\text{otherwise}) \end{cases} \tag{3}$$

$$\text{This paper:} \qquad F(k=(\rho,\mathbf{R}),(\mathbf{x},\mathsf{Msg})) = \begin{cases} f(\rho,\mathsf{Msg}) & (\text{if } \mathbf{x}\in\mathbf{R}); \\ \perp & (\text{otherwise}) \end{cases} \tag{4}$$

where f is some one-way function and will be defined later.

On the other hand, MRQED has its own advantages over our proposed functional encryption scheme —MRQED [15] is a public key encryption scheme and has a stronger security model.

Other recent works in functional encryption include [16,17,18,19,20].

3 Construction of A New Functional Encryption Scheme

3.1 Polymorphic Property of BBG HIBE Scheme

We observe that the BBG HIBE scheme [2] satisfies the polymorphic property: An encryption of a message M can be viewed as the encryption of another message $\widehat{M}$ under different key. Precisely, let CT and $\widehat{\mathsf{CT}}$ be defined as follows, we have $\mathsf{CT} = \widehat{\mathsf{CT}}$:

$$\begin{aligned} &\mathsf{CT} = \mathsf{Encrypt}(\mathsf{params},\mathsf{id},M;s) = \left(\Omega^s\cdot M,\ g^s,\ \left(h_1^{I_1}\cdots h_k^{I_k}\cdot g_3\right)^s\right) \\ &\quad \text{under key: } \mathtt{params} = (g,g_1,g_2,g_3,h_1,\ldots,h_\ell,\Omega=e(g_1,g_2)),\ \ \mathtt{master\text{-}key} = g_2^{\alpha} \\ &\widehat{\mathsf{CT}} = \mathsf{Encrypt}(\widehat{\mathsf{params}},\mathsf{id},\widehat{M};sz) = \left(\Omega^{sz}\cdot\widehat{M},\ \widehat{g}^{sz},\ \left(\widehat{h}_1^{I_1}\cdots\widehat{h}_k^{I_k}\cdot\widehat{g}_3\right)^{sz}\right), \\ &\quad \text{under key: } \widehat{\mathtt{params}} = (\widehat{g},g_1,g_2,\widehat{g}_3,\widehat{h}_1,\ldots,\widehat{h}_\ell,\Omega=e(g_1,g_2)),\ \ \widehat{\mathtt{master\text{-}key}} = g_2^{\alpha z} \end{aligned} \tag{5}$$

where ℓ is the maximum depth of the HIBE scheme, $k\le\ell$ is the length of identity id, $\widehat{M} = M\Omega^{s(1-z)}$, $\widehat{g} = g^{z^{-1} \bmod p}$, $\widehat{g}_3 = g_3^{z^{-1} \bmod p}$, $\widehat{h}_i = h_i^{z^{-1} \bmod p}$ for $1\le i\le\ell$ and identity $\mathsf{id} = (I_1,\ldots,I_k)\in\left(\mathbb{Z}_p^*\right)^k$. To be self-contained, the description of this BBG HIBE scheme is given in Appendix A (on page 14). One can verify the above equality easily.

3.2 Define Identities based on Binary Interval Tree

An identity is a sequence of elements from $\mathbb{Z}_p^*$. To apply HIBE scheme, we intend to construct two mappings, named ID and IdSet, to associate identities to integers or integer intervals:

– $\mathsf{ID}(\cdot)$ maps an integer $x\in[\mathcal{Z}]$ into an identity $\mathsf{ID}(x)\in\left(\mathbb{Z}_p^*\right)^\ell$, where $\ell=\lceil\log\mathcal{Z}\rceil$ is the height of identity hierarchy tree of the BBG HIBE scheme.
– $\mathsf{IdSet}(\cdot)$ maps an integer interval $[a,b]\subseteq[\mathcal{Z}]$ into a set of $O(\ell)$ identities, where each identity is a sequence of at most ℓ elements from $\mathbb{Z}_p^*$.

The two mappings ID and IdSet are required to satisfy this property: *For any $x \in [a,b] \subseteq [\mathcal{Z}]$, there is a unique identity id in the set $\mathsf{IdSet}([a,b])$, such that identity id is a prefix of identity $\mathsf{ID}(x)$. If $x \notin [a,b]$, then there is no such identity id in $\mathsf{IdSet}([a,b])$. For each dimension $\iota \in [d]$, we will construct such mappings ID_ι and IdSet_ι using a* binary interval tree *[15]. The resulting mappings are made public.*

Binary Interval Tree. The binary interval tree is constructed as below: First, we build a complete ordered binary tree with 2^ℓ leaf nodes. Next, we associate an integer interval to each tree node in a bottom-up manner: (1) Counting from the leftmost leaf, the j-th leaf is associated with interval $[j,j]$; (2) For any internal node, the associated interval is the union of the two intervals associated to its left and right children respectively. As a result, the interval associated to the root node is $[1, 2^\ell]$. An example of binary interval tree with size 8 is showed in Figure 1.

Constructions of Mappings ID_ι and IdSet_ι for dimension ι. Let $\mathcal{H} : \mathbb{Z}_{2^\ell+1} \times \mathbb{Z}_{2^\ell+1} \times [d] \to \mathbb{Z}_p^*$ be a collision resistant hash function. Let $(\mathtt{v}_1, \mathtt{v}_2, \ldots, \mathtt{v}_m)$ be the unique simple path from the root node $\mathtt{v}_1$ to the node $\mathtt{v}_m$ in the binary interval tree. We associate to node $\mathtt{v}_m$ the identity $(\mathcal{H}(a_1, b_1, \iota), \ldots, \mathcal{H}(a_m, b_m, \iota)) \in \left(\mathbb{Z}_p^*\right)^m$, where $[a_j, b_j]$ is the interval associated to node $\mathtt{v}_j$, $1 \le j \le m$.

For any $x \in [\mathcal{Z}]$, we define $\mathsf{ID}_\iota(x)$ as the identity associated to the x-th leaf node (counting from the left). For any interval $[a,b] \subseteq [\mathcal{Z}]$, we find the minimum covering set $\mathsf{MCS}_{a,b}$, which is a set $\{\mathtt{v}_j : \mathtt{v}_j \text{ is a tree node}, 1 \le j \le n\}$ with minimal size such that the intervals associated to $\mathtt{v}_j$'s form a partition of the interval $[a,b]$, and define $\mathsf{IdSet}_\iota([a,b]) := \{\mathsf{id}_j : \mathsf{id}_j \text{ is the identity associated to node } \mathtt{v}_j,\ \mathtt{v}_j \in \mathsf{MCS}_{a,b}\}$. One can verify that the newly constructed mappings ID_ι and IdSet_ι satisfy the property mentioned in the beginning of Section 3.2. Furthermore, the set $\mathsf{IdSet}_\iota([a,b])$ contains $O(\ell)$ identities and each identity is a sequence of at most ℓ elements from $\mathbb{Z}_p^*$.

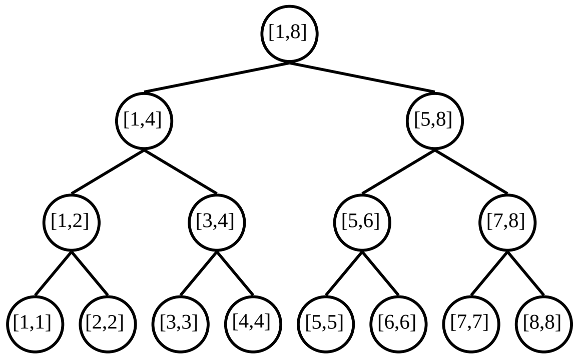

Fig. 1. Binary Interval Tree with 8 leaf nodes.

3.3 Construction of Functional Encryption Scheme

Let (Setup, KeyGen, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$) be the BBG Hierarchical Identity Based Encryption (HIBE) scheme proposed by Boneh, Boyen and Goh [2] (the description of this scheme is in Appendix A on page 14). Based on this HIBE scheme, we construct a functional encryption scheme $\mathsf{FE} = (f\mathsf{Setup}, f\mathsf{Enc}, f\mathsf{KeyGen}, f\mathsf{Dec})$ as below.

$\underline{f\mathsf{Setup}(1^\lambda, d, \mathcal{Z})}$: `security parameter` λ`, dimension` d`, maximum integer` $\mathcal{Z}$`; the domain of points is` $[\mathcal{Z}]^d$

1. Let $\ell = \lceil \log \mathcal{Z} \rceil$. Run algorithm $\mathsf{Setup}(\ell, \lambda)$ to obtain bilinear groups $(p, \mathbb{G}, \widetilde{\mathbb{G}}, e)$, public parameter $\mathtt{params} = (g, g_1, g_2, g_3, h_1, \ldots, h_\ell, \Omega = e(g_1, g_2))$ and master private key $\mathtt{master\text{-}key} = g_2^\alpha$, such that p is a λ bits prime, $\mathbb{G}, \widetilde{\mathbb{G}}$ are cyclic multiplicative groups of order p, $e : \mathbb{G} \times \mathbb{G} \to \widetilde{\mathbb{G}}$ is a bilinear map, g is a generator of $\mathbb{G}$, $\alpha \in \mathbb{Z}_p$, $g_1 = g^\alpha \in \mathbb{G}$, and $g_2, g_3, h_1, \ldots, h_\ell \in \mathbb{G}$.
2. Let ID_ι and IdSet_ι, $\iota \in [d]$, be the mappings as in Section 3.2.
3. Choose d random elements $\tau_1, \ldots, \tau_d$ from $\mathbb{Z}_p^*$ and let $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_d)$.
4. Let $pk = (p, \mathbb{G}, \widetilde{\mathbb{G}}, e, \Omega)$ and $sk = (pk, \mathtt{params}, \mathtt{master\text{-}key}, \boldsymbol{\tau})$. Make ID_ι's and IdSet_ι's public and output (pk, sk).

$\underline{f\mathsf{Enc}(\mathsf{Msg}, \boldsymbol{x}, sk)}$: `message` $\mathsf{Msg} \in \mathbb{Z}_p^*$`,` d`-dimensional point` $\boldsymbol{x}$

1. Treat the d-dimensional point $\boldsymbol{x}$ as $(x_1, \ldots, x_d) \in [\mathcal{Z}]^d$; recall that the private key sk is $(pk, \mathsf{params}, \mathsf{master\text{-}key}, \boldsymbol{\tau})$, where $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_d)$.
2. Choose d random elements $s_1, \ldots, s_d$ from $\mathbb{Z}_p$ with constraint $\mathsf{Msg} = -\sum_{j=1}^d s_j \cdot \tau_j \pmod p$.
3. Choose d random elements $\sigma_1, \ldots, \sigma_d$ from $\widetilde{\mathbb{G}}$ with constraint $\prod_{j=1}^d \sigma_j = \Omega^{-\sum_{j=1}^d s_j}$.
4. For each $j \in [d]$, encrypt σ_j under identity $\mathsf{ID}_j(x_j)$ with random coin s_j to obtain ciphertext $\boldsymbol{c}_j$ as follows
$$\boldsymbol{c}_j \leftarrow \mathsf{Encrypt}(\mathtt{params},\ \mathsf{ID}_j(x_j),\ \sigma_j;\ s_j). \tag{6}$$
5. Output ciphertext $\mathsf{CT} = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_d)$.

$\underline{f\mathsf{KeyGen}(\mathbf{R}, \rho, sk)}$: d`-dimensional rectangular range` $\mathbf{R}$`, function key` $\rho \in \mathbb{Z}_p^*$

1. Treat the d-dimensional rectangular range $\mathbf{R} \subseteq [\mathcal{Z}]^d$ as Cartesian product $\mathbf{A}_1 \times \mathbf{A}_2 \ldots \times \mathbf{A}_d$, where $\mathbf{A}_j \subseteq [\mathcal{Z}]$ for each $j \in [d]$; recall that the private key sk is $(pk, \mathsf{params}, \mathsf{master\text{-}key}, \boldsymbol{\tau})$, where $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_d)$.
2. For each dimension $j \in [d]$, generate a set δ_j in this way:
   (a) For each identity $\mathsf{id} \in \mathsf{IdSet}_j(\mathbf{A}_j)$, generate the private key d_{id}, using algorithm KeyGen and taking the value $\mathsf{master\text{-}key}^{\rho\tau_j}$ as the master key.
   (b) Set $\delta_j \leftarrow \{d_{\mathsf{id}} : \mathsf{id} \in \mathsf{IdSet}_j(\mathbf{A}_j)\}$.
3. Output delegation key $\boldsymbol{\delta} = (\delta_1, \delta_2, \ldots, \delta_d)$.

$\underline{f\mathsf{Dec}(\mathsf{CT}, \boldsymbol{x}, \mathbf{R}, \boldsymbol{\delta}, pk)}$: `ciphertext` CT`,` d`-dimensional point` $\boldsymbol{x}$`,` d`-dimensional rectangular range` $\mathbf{R}$`, delegation key` $\boldsymbol{\delta}$

1. Treat the d-dimensional rectangular range $\mathbf{R} \subseteq [\mathcal{Z}]^d$ as Cartesian product $\mathbf{A}_1 \times \mathbf{A}_2 \ldots \times \mathbf{A}_d$, where $\mathbf{A}_j \subseteq [\mathcal{Z}]$ for each $j \in [d]$. Let us write the ciphertext CT as $(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_d)$, and the d-dimensional point $\boldsymbol{x}$ as $(x_1, \ldots, x_d)$.

2. For each dimension $j \in [d]$, generate $\widetilde{t_j}$ in this way: If $x_j \notin \mathbf{A}_j$, then output $\perp$ and abort. Otherwise, do the followings:
   (a) Find the unique identity $\mathsf{id}^* \in \mathsf{IdSet}_j(\mathbf{A}_j)$ such that id^* is a prefix of identity $\mathsf{ID}_j(x_j)$.
   (b) Parse $\boldsymbol{\delta}$ as $(\delta_1, \ldots, \delta_d)$ and find the private key $d_{\mathsf{id}^*} \in \delta_j = \{d_{\mathsf{id}} : \mathsf{id} \in \mathsf{IdSet}_j(\mathbf{A}_j)\}$ for identity id^*.
   (c) Generate the private key d_j for the identity $\mathsf{ID}_j(x_j)$ from private key d_{id^*}, using algorithm KeyGen.
   (d) Decrypt $\boldsymbol{c}_j$ using algorithm $\mathsf{Decrypt}$ with decryption key d_j, and denote the decrypted message as $\widetilde{t_j}$.
3. Output $\widetilde{t} = \prod_{1 \le j \le d} \widetilde{t_j} \ \in \widetilde{\mathbb{G}}$.

4 The Constructed Functional Encryption Scheme is Correct and Secure

In this section, we analyze the correctness and security of the newly constructed functional encryption scheme.

4.1 Correctness

Let us define a key-ed function family $\{f_\rho : \mathbb{Z}_p^* \to \widetilde{\mathbb{G}}\}_{\rho \in \mathbb{Z}_p^*}$ as below: Let $\Omega \in \widetilde{\mathbb{G}}$ be as in $f\mathsf{Setup}$ of Section 3.3.

$$f_1(\mathsf{Msg}) = \Omega^{\mathsf{Msg}}; \qquad \forall \rho \in \mathbb{Z}_p^*, \ f_\rho(\mathsf{Msg}) = f_1(\mathsf{Msg})^\rho \ \in \widetilde{\mathbb{G}}. \tag{7}$$

Lemma 1 (FE is correct) *The functional encryption scheme* FE *described in Section 3.3 satisfies this property: For any public-private key pair* $(pk, sk) \leftarrow f\mathsf{Setup}(1^\lambda, d, \mathcal{Z})$, *for any message* $\mathsf{Msg} \in \mathbb{Z}_p^*$, *for any point* $\boldsymbol{x} \in [\mathcal{Z}]^d$, *for any rectangular range* $\mathbf{R} \subseteq [\mathcal{Z}]^d$, *for any* $\rho \in \mathbb{Z}_p^*$, *if* $\mathsf{CT} \leftarrow f\mathsf{Enc}(\mathsf{Msg}, \boldsymbol{x}, sk)$ *and* $\boldsymbol{\delta} \leftarrow f\mathsf{KeyGen}(\mathbf{R}, \rho, sk)$, *then*

$$f\mathsf{Dec}(\mathsf{CT},\ \boldsymbol{x},\ \mathbf{R},\ \boldsymbol{\delta},\ pk) = \begin{cases} f_\rho(\mathsf{Msg}) & (\mathit{if}\ \boldsymbol{x} \in \mathbf{R}) \\ \perp & (\mathit{otherwise}) \end{cases} \tag{8}$$

Most of previous functional encryption schemes [5] (e.g. attribute-based encryption [13], and predicate encryption [14]), if not all, allow the decryptor to obtain the original plaintext Msg from a ciphertext of Msg in "good" case (e.g. if the attribute of plaintext and/or the decryption key satisfy the designated predicate), and nothing otherwise. In contrast, our functional encryption scheme FE only allows the decryptor to obtain $f_1(\mathsf{Msg})^\rho$ in "good" case, from a ciphertext of Msg, where f_1 is a one-way function. Unlike [3,4], our functional encryption scheme is a symmetric key system. Our security formulation is weaker than previous works (e.g. [3,4]).

4.2 Proof of Correctness

Proof (of Lemma 1). We observe that the BBG HIBE scheme [2] satisfies the polymorphic property: An encryption of a message M can be viewed as the encryption of another message $\widehat{M}$ under different key. Precisely, let CT and $\widehat{\mathsf{CT}}$ be defined as follows, we have $\mathsf{CT} = \widehat{\mathsf{CT}}$:

$$\begin{aligned}
&\mathsf{CT} = \mathsf{Encrypt}(\mathsf{params}, \mathsf{id}, M; s) = \left(\Omega^s \cdot M,\ g^s,\ \left(h_1^{I_1} \cdots h_k^{I_k} \cdot g_3 \right)^s \right) \\
&\quad \text{under key: } \mathtt{params} = (g, g_1, g_2, g_3, h_1, \ldots, h_\ell, \Omega = e(g_1, g_2)),\ \ \mathtt{master\text{-}key} = g_2^{\alpha} \\
&\widehat{\mathsf{CT}} = \mathsf{Encrypt}(\widehat{\mathsf{params}}, \mathsf{id}, \widehat{M}; sz) = \left(\Omega^{sz} \cdot \widehat{M},\ \widehat{g}^{sz},\ \left(\widehat{h}_1^{I_1} \cdots \widehat{h}_k^{I_k} \cdot \widehat{g}_3 \right)^{sz} \right), \\
&\quad \text{under key: } \widehat{\mathtt{params}} = (\widehat{g}, g_1, g_2, \widehat{g}_3, \widehat{h}_1, \ldots, \widehat{h}_\ell, \Omega = e(g_1, g_2)),\ \ \widehat{\mathtt{master\text{-}key}} = g_2^{\alpha z} \qquad (9)
\end{aligned}$$

where identity $\mathsf{id} = (I_1, \ldots, I_k) \in \left(\mathbb{Z}_p^*\right)^k$, $\widehat{M} = M\Omega^{s(1-z)}$, $\widehat{g} = g^{z^{-1} \bmod p}$, $\widehat{g}_3 = g_3^{z^{-1} \bmod p}$ and $\widehat{h}_i = h_i^{z^{-1} \bmod p}$ for $1 \le i \le \ell$. To be self-contained, the description of this BBG HIBE scheme [2] is given in Appendix A. One can verify the above equality easily.

Let $(pk, sk) \leftarrow f\mathsf{Setup}(1^\lambda)$, message $\mathsf{Msg} \in \mathbb{Z}_p^*$, point $\boldsymbol{x} \in [\mathcal{Z}]^d$, $\mathbf{R}$ be a d-dimensional rectangular range, and $\rho \in \mathbb{Z}_p^*$. Let $\mathsf{CT} \leftarrow f\mathsf{Enc}(\mathsf{Msg}, \boldsymbol{x}, sk)$, $\boldsymbol{\delta} \leftarrow f\mathsf{KeyGen}(\mathbf{R}, \rho, sk)$, and $y \in \widetilde{\mathbb{G}}$.

We consider dimension $j \in [d]$ and apply the polymorphic property of BBG scheme (Equation (9)): Take $M = \sigma_j, s = s_j$ and $z = \rho\tau_j$. Then $\widehat{M} = M\Omega^{s(1-z)} = \sigma_j \Omega^{s_j(1-\tau_j\rho)}$.

In case $\boldsymbol{x} \in \mathbf{R}$. If $\boldsymbol{x} \in \mathbf{R}$, then the HIBE decryption will succeed in the process of $f\mathsf{Dec}$ (Section 3.3). Note that during decryption for dimension j, we use decryption key derived from $\mathsf{master\text{-}key}^{\rho\tau_j}$. Let $\widetilde{t}_j$ be as in Step 2(d) of $f\mathsf{Dec}$ for decrypting ciphertext CT. We have

$$\widetilde{t}_j = \widehat{M} = \sigma_j \Omega^{s_j(1-\tau_j\rho)}, j \in [d]. \qquad (10)$$

Combining all d dimensions, and applying the two equalities (see algorithm $f\mathsf{Enc}$ in Section 3.3) $\mathsf{Msg} = -\sum_{j=1}^d s_j\tau_j \mod p$ and $\prod_{j=1}^d \sigma_j = \Omega^{-\sum_{j=1}^d s_j}$ we have,

$$\begin{aligned}
f\mathsf{Dec}(\mathsf{CT},\ \boldsymbol{x},\ \mathbf{R},\ \boldsymbol{\delta},\ pk) = \widetilde{t} &= \prod_{j=1}^d \widetilde{t}_j = \prod_{j=1}^d \left(\sigma_j \Omega^{s_j(1-\tau_j\rho)} \right) \\
&= \prod_{j=1}^d \sigma_j \ \cdot \ \prod_{j=1}^d \ \Omega^{s_j} \ \cdot \ \left(\prod_{j=1}^d \Omega^{-s_j\tau_j} \right)^{\rho} \\
&= \Omega^{-\sum_{j=1}^d s_j} \cdot \prod_{j=1}^d \ \Omega^{s_j} \cdot \left(\Omega^{\mathsf{Msg}} \right)^{\rho} \\
&= \Omega^{\rho \mathsf{Msg}} \\
&= f_\rho(\mathsf{Msg}).
\end{aligned}$$

In case $\boldsymbol{x} \notin \mathbf{R}$. Let $\mathbf{R} = \mathbf{A}_1 \times \mathbf{A}_2 \ldots \times \mathbf{A}_d$ as in Step 1 of $f\mathsf{Dec}$. If $\boldsymbol{x} \notin \mathbf{R}$, then for some dimension $j \in [d]$, $\boldsymbol{x}[j] \notin \mathbf{A}_j$, and $f\mathsf{Dec}$ will output $\perp$ (Step 2 of $f\mathsf{Dec}$ in Section 3.3).

4.3 Security

We formulize the security requirement of our functional encryption scheme by modifying the $\mathsf{IND\text{-}sID\text{-}CPA}$ security game [2]. The resulting $\mathsf{weak\text{-}IND\text{-}sID\text{-}CPA}$ security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ is defined as below:

Commit: The adversary $\mathcal{A}$ chooses the target point $\boldsymbol{x}^*$ from the space $[\mathcal{Z}]^d$ and sends it to the challenger $\mathcal{C}$.
Setup: The challenger $\mathcal{C}$ runs the setup algorithm $f\mathsf{Setup}$ and gives $\mathcal{A}$ the resulting system parameters pk, keeping the secret key sk to itself.
Challenge: The challenger $\mathcal{C}$ chooses two plaintexts $\mathsf{Msg}_0, \mathsf{Msg}_1$ at random from the message space $\mathbb{Z}_p^*$, and chooses a random bit $b \in \{0,1\}$. $\mathcal{C}$ sets the challenge ciphertext to $\mathsf{CT} = f\mathsf{Enc}(\mathsf{Msg}_b, \boldsymbol{x}^*, sk)$, and sends $(\mathsf{CT}, f_1(\mathsf{Msg}_0), f_1(\mathsf{Msg}_1))$ to the adversary $\mathcal{A}$.

Learning Phase: The adversary $\mathcal{A}$ adaptively issues queries to the challenger $\mathcal{C}$, where each query is one of the following:

– Delegation key query $(\mathbf{R}, \rho)$, where $\boldsymbol{x}^* \notin \mathbf{R}$: In response to this query, $\mathcal{C}$ runs algorithm $f\mathsf{KeyGen}(\mathbf{R}, \rho, sk)$ to generate the delegation key $\boldsymbol{\delta}$, and sends $\boldsymbol{\delta}$ to $\mathcal{A}$.
– Anonymous delegation key query $(\mathbf{R})$: In response to this query, $\mathcal{C}$ chooses ρ at random from the space $\mathbb{Z}_p^*$ and runs algorithm $f\mathsf{KeyGen}(\mathbf{R}, \rho, sk)$ to generate the delegation key $\boldsymbol{\delta}$, and sends $\boldsymbol{\delta}$ to $\mathcal{A}$.
– Encryption query $(\mathsf{Msg}, \boldsymbol{x})$: In response to this query, $\mathcal{C}$ runs $f\mathsf{Enc}(\mathsf{Msg}, \boldsymbol{x}, sk)$ to obtain a ciphertext, and sends the ciphertext to $\mathcal{A}$.

Guess: Finally, the adversary $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$ and wins if $b = b'$.

We refer to the above adversary $\mathcal{A}$ as a weak-IND-sID-CPA adversary. We define the advantage of the adversary $\mathcal{A}$ in attacking the scheme FE as

$$\mathsf{Adv}_{\mathsf{FE},\mathcal{A}}^{\mathsf{weak\text{-}IND\text{-}sID\text{-}CPA}} = \left| \mathsf{Pr}[b = b'] - \frac{1}{2} \right|.$$

We will show in the following theorem that: if the BBG HIBE scheme is IND-sID-CPA secure (as defined in [2]), then the functional encryption scheme FE constructed in Section 3.3 is weak-IND-sID-CPA secure. Note that the above security definition is *weak* in the sense that the two challenged messages Msg_0 and Msg_1 are chosen randomly instead of adversarially. Compared to Gorbunov *et al.* [1] which only allows a pre-defined number of delegation queries (they called "q-Collusions"), the above security definition allows practically unlimited number of delegation key queries.

Theorem 2 *Suppose there exists a weak-IND-sID-CPA adversary $\mathcal{A}_{\mathsf{FE}}$, which runs in time t_{FE} and has non-negligible advantage ϵ against the functional encryption scheme FE (constructed in Section 3.3) with one chosen delegation key query and N_{aq} chosen anonymous delegation key queries and N_{enc} chosen encryption queries. Then there exists an IND-sID-CPA adversary $\mathcal{A}_{\mathsf{BBG}}$, which has advantage $\frac{\epsilon}{2d}$ against the BBG HIBE scheme [2] with $O(d\ell)$ chosen private key queries and zero chosen decryption query, and runs in time $t_{\mathsf{FE}} + O(d\ell \cdot t_{max} \cdot (N_{aq} + N_{enc}))$, where t_{max} is the maximum time for a random sampling (within a space of size at most p), a BBG encryption* Encrypt, *or a BBG key generation* KeyGen.

4.4 Proof of Security

Proof (of Theorem 2).

The proof idea. Let $\mathcal{A}_{\mathsf{FE}}$ be the weak-IND-sID-CPA adversary against the functional encryption scheme FE as in Theorem 2. We try to construct an IND-sID-CPA adversary $\mathcal{A}_{\mathsf{BBG}}$ against BBG based on $\mathcal{A}_{\mathsf{FE}}$: Choose two random messages m_0 and m_1, and send them to the BBG challenger. After receiving the challenge ciphertext CT for message m_b where $b \in \{0,1\}$, guess $b = 0$ and construct a FE challenge $(f_1(\mathsf{Msg}_0), f_1(\mathsf{Msg}_1), \mathsf{CT}_{\mathsf{FE}})$ based on the BBG challenge CT. If the adversary $\mathcal{A}_{\mathsf{FE}}$ wins the weak-IND-sID-CPA game, then output a guess $b' = 0$; otherwise output a guess $b' = 1$.

We argue that if indeed $b = 0$, then the forged FE challenge is valid, and the hypothesis is applicable: $\mathcal{A}_{\mathsf{FE}}$ wins with probability $1/2 + \epsilon$. If $b = 1$, the forged FE challenge is invalid, we cannot apply the hypothesis. However, in this case the forged FE challenge is independent on the value of Msg_a. Hence, in case of $b = 1$, $\mathcal{A}_{\mathsf{FE}}$ wins (in guessing the value $a \in \{0,1\}$) with probability exactly 1/2.

Recall that the BBG HIBE scheme is (Setup, KeyGen, Encrypt, Decrypt) and the functional encryption scheme FE is (fSetup, fEnc, fKeyGen, fDec). Now let us construct the IND-sID-CPA adversary $\mathcal{A}_{\mathsf{BBG}}$ against BBG. $\mathcal{A}_{\mathsf{BBG}}$ will simulate the weak-IND-sID-CPA game where $\mathcal{A}_{\mathsf{BBG}}$ takes the role of challenger and invokes $\mathcal{A}_{\mathsf{FE}}$ in the hypothesis as the adversary.

Construction of IND-sID-CPA adversary $\mathcal{A}_{\mathsf{BBG}}$ against BBG HIBE scheme based on $\mathcal{A}_{\mathsf{FE}}$

BBG Commit :

FE Commit : Adversary $\mathcal{A}_{\mathsf{FE}}$ chooses a random point $\boldsymbol{x}^* = (x_1, \ldots, x_d) \in [\mathcal{Z}]^d$. $\mathcal{A}_{\mathsf{FE}}$ sends $\boldsymbol{x}^*$ to FE challenger $\mathcal{A}_{\mathsf{BBG}}$ as the target identity.

BBG adversary $\mathcal{A}_{\mathsf{BBG}}$ chooses $\xi \in [d]$ at random and sends target identity $\mathsf{id}^* = \mathsf{ID}_\xi(x_\xi) \in (\mathbb{Z}_p^*)^\ell$ to BBG challenger $\mathcal{C}_{\mathsf{BBG}}$.

BBG Setup : BBG challenger $\mathcal{C}_{\mathsf{BBG}}$ runs setup algorithm Setup, and give $\mathcal{A}_{\mathsf{BBG}}$ the resulting system parameter params, keeping the master-key private.

BBG Phase 1 : Adversary $\mathcal{A}_{\mathsf{BBG}}$ does nothing.

BBG Challenge : Adversary $\mathcal{A}_{\mathsf{BBG}}$ chooses m_0, m_1 at random from the plaintext space $\widetilde{\mathbb{G}}$, and sends (m_0, m_1) to the challenger $\mathcal{C}_{\mathsf{BBG}}$. $\mathcal{C}_{\mathsf{BBG}}$ picks a random bit $b \in \{0,1\}$ and sends the challenge ciphertext $\mathsf{CT} = \mathsf{Encrypt}(\mathsf{params}, \mathsf{id}^*, m_b; s)$ to $\mathcal{A}_{\mathsf{BBG}}$.

BBG Phase 2 :

FE Setup : $\mathcal{A}_{\mathsf{BBG}}$ chooses d random elements $\tau_1, \ldots, \tau_d$ from $\mathbb{Z}_p^*$ and let $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_d)$. Let $(p, \mathbb{G}, \widetilde{\mathbb{G}}, e, \Omega)$ be a part of params, where p is a prime, both $\mathbb{G}$ and $\widetilde{\mathbb{G}}$ are cyclic multiplicative group of order p, $e : \mathbb{G} \times \mathbb{G} \to \widetilde{\mathbb{G}}$ is a bilinear map, and $\Omega \in \widetilde{\mathbb{G}}$. Let $pk = (p, \mathbb{G}, \widetilde{\mathbb{G}}, e, \Omega)$ and $sk = (pk, \mathsf{params}, \mathsf{master\text{-}key}, \boldsymbol{\tau})$. $\mathcal{A}_{\mathsf{BBG}}$ sends pk to $\mathcal{A}_{\mathsf{FE}}$.
Note: $\mathcal{A}_{\mathsf{BBG}}$ does not know master-key.

FE Challenge : The FE challenger $\mathcal{A}_{\mathsf{BBG}}$ chooses a random bit $a \in \{0,1\}$ and a random message Msg_{1-a} from the message space $\mathbb{Z}_p^*$. $\mathcal{A}_{\mathsf{BBG}}$ will decide Msg_a and generate the challenge ciphertext $\mathsf{CT}_{\mathsf{FE}}$ in this way:

1. Parse the BBG challenge ciphertext as $\mathsf{CT} = (A, B, C)$, where $A = \Omega^s m_b$.
2. Choose $(d-1)$ random elements $s_1, \ldots, s_{\xi-1}, s_{\xi+1}, \ldots, s_d$ (i.e. excluding s_ξ) from $\mathbb{Z}_p^*$.
3. Choose d random elements $\sigma_1, \ldots, \sigma_d$ from $\widetilde{\mathbb{G}}$ with constraint

$$\prod_{j=1}^{d} \sigma_j = (\Omega^s m_b)^{-1} m_0 \cdot \Omega^{-\sum_{\substack{1 \le j \le d \\ j \ne \xi}} s_j}$$

4. For each $j \in [d]$ and $j \neq \xi$, encrypt σ_j under identity $\mathsf{ID}_j(x_j)$ with random coin s_j to obtain ciphertext $\boldsymbol{c}_j$ as follows
$$\boldsymbol{c}_j \leftarrow \mathsf{Encrypt}(\mathtt{params},\ \mathsf{ID}_j(x_j),\ \sigma_j;\ s_j). \tag{11}$$
5. Define $\boldsymbol{c}_\xi$ based on the BBG challenge ciphertext $\mathsf{CT} = (\varOmega^s m_b, B, C)$:
$$\boldsymbol{c}_\xi = (\varOmega^s m_b \cdot m_0^{-1} \cdot \sigma_\xi,\ B,\ C).$$
6. Define $\mathsf{Msg}_a = -\sum_{j\in[d]} s_j \cdot \tau_j \pmod p$, where unknown $s_\xi \in \mathbb{Z}_p$ is defined by $\varOmega^{s_\xi} = \varOmega^s m_b \cdot m_0^{-1}$. Although the value Msg_a is unknown since s_ξ is unknown, $\mathcal{A}_{\mathsf{BBG}}$ can still compute $f_1(\mathsf{Msg}_a)$:
$$f_1(\mathsf{Msg}_a) = \varOmega^{\mathsf{Msg}_a} = \left((\varOmega^s m_b)^{-1} \cdot m_0\right)^{\tau_\xi} \cdot \varOmega^{-\sum\limits_{\substack{1\le j\le d\\ j\neq\xi}} s_j\cdot\tau_j}$$
$\mathcal{A}_{\mathsf{BBG}}$ computes $f_1(\mathsf{Msg}_{1-a}) = \varOmega^{\mathsf{Msg}_{1-a}}$.
7. Set the challenge ciphertext to $\mathsf{CT}_{\mathsf{FE}} = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_d)$, and send $(\mathsf{CT}_{\mathsf{FE}},\ f_1(\mathsf{Msg}_0),\ f_1(\mathsf{Msg}_1))$ to $\mathcal{A}_{\mathsf{FE}}$.

FE Learning Phase :

1. $\mathcal{A}_{\mathsf{FE}}$ issues a *delegation key query* $(\mathbf{R}, \rho)$, where $\boldsymbol{x}^* \notin \mathbf{R}$ and $\mathbf{R} = \mathbf{A}_1 \times \mathbf{A}_2 \ldots \times \mathbf{A}_d \subseteq [\mathcal{Z}]^d$: If $x_\xi \in \mathbf{A}_\xi$, then $\mathcal{A}_{\mathsf{BBG}}$ aborts and outputs a random bit $b' \in \{0,1\}$ (Denote this event as $\mathbf{E}_1$). Otherwise, simulate the procedure of $f\mathsf{KeyGen}$:
   (a) The private key is $sk = (pk, \mathsf{params}, \mathsf{master\text{-}key}, \boldsymbol{\tau} = (\tau_1, \ldots, \tau_d))$, where $\mathcal{A}_{\mathsf{BBG}}$ has only pk, params and $\boldsymbol{\tau}$, and does not know $\mathsf{master\text{-}key}$ (which is kept securely by the BBG challenger $\mathcal{C}_{\mathsf{BBG}}$).
   (b) For each $j \in [d]$, generate a set δ_j in this way:
   – For each identity $\mathsf{id} \in \mathsf{IdSet}_j(\mathbf{A}_j)$, issue a private key query with identity id to BBG challenger $\mathcal{C}_{\mathsf{BBG}}$ and get reply d_{id}.
   Note: The BBG private key query (id*) is valid, i.e.* $\mathsf{id} \neq \mathsf{id}^*$ *and* id *is not a prefix of* id^**. This is implied by the following two properties satisfied by our constructions of* ID_ι *and* IdSet_ι *in Section 3.2: (1) For any* $i, j \in [d], x, y \in [\mathcal{Z}]$*, if* $\mathsf{ID}_i(x)$ *and* $\mathsf{ID}_j(y)$ *share a non-empty prefix, then* $i = j$*; (2) For any* $\in [a,b] \subseteq [\mathcal{Z}]$*, iff* $x \in [a,b]$*, there exits an identity* id *in the set* $\mathsf{IdSet}_j([a,b])$*, such that* id *is a prefix of identity* $\mathsf{ID}_j(x)$*.*
   – For each identity $\mathsf{id} \in \mathsf{IdSet}_j(\mathbf{A}_j)$, parse the key d_{id} as $(K_0, K_1, \varUpsilon_k, \ldots, \varUpsilon_\ell)$ and set $d'_{\mathsf{id}} = (K_0^{\rho\tau_j}, K_1^{\rho\tau_j}, \varUpsilon_k^{\rho\tau_j}, \ldots, \varUpsilon_\ell^{\rho\tau_j})$.
   – Set $\delta_j \leftarrow \{d'_{\mathsf{id}} : \mathsf{id} \in \mathsf{IdSet}_j(\mathbf{A}_j)\}$.
   (c) Send $\boldsymbol{\delta} = (\delta_1, \delta_2, \ldots, \delta_d)$ to $\mathcal{A}_{\mathsf{FE}}$ as the delegation key w.r.t. $(\mathbf{R}, \rho)$.
   Note: $\mathcal{A}_{\mathsf{FE}}$ *can make at most one delegation key query.*
2. $\mathcal{A}_{\mathsf{FE}}$ issues an *anonymous delegation key query* $(\mathbf{R})$: Choose a random element $Z \in \widetilde{\mathbb{G}}$. For each anonymous delegation key query $(\mathbf{R})$, choose $\rho \in \mathbb{Z}_p^*$ at random, run the algorithm $f\mathsf{KeyGen}(\mathbf{R}, \rho, sk')$, where $sk' = (pk, \mathsf{params}, Z, \boldsymbol{\tau})$ (i.e. taking Z as the master key), and get output $\boldsymbol{\delta}$. Send $\boldsymbol{\delta}$ to $\mathcal{A}_{\mathsf{FE}}$ as the delegation key w.r.t. $\mathbf{R}$.
   Note: (1) $\mathcal{A}_{\mathsf{BBG}}$ *can answer anonymous delegation key query* without *the help of BBG challenger* $\mathcal{C}_{\mathsf{BBG}}$*. (2) There exists an unknown* ω*, such that* $Z = \mathsf{master\text{-}key}^\omega$*. The generated delegation key* $\boldsymbol{\delta}$ *corresponds to range* $\mathbf{R}$ *and (unknown) function key* $\rho\omega$*, where* $\rho\omega$ *is uniformly distributed in* $\mathbb{Z}_p^*$ *as desired.*
3. $\mathcal{A}_{\mathsf{FE}}$ issues an encryption key query $(\mathsf{Msg}, \boldsymbol{x})$: Run the encryption algorithm: $\mathsf{C} \leftarrow f\mathsf{Enc}(\mathsf{Msg}, \boldsymbol{x}, sk)$ and send the resulting ciphertext C to $\mathcal{A}_{\mathsf{FE}}$ as the reply.
   Note: $\mathcal{A}_{\mathsf{BBG}}$ *can run algorithm* $f\mathsf{Enc}$*, since it requires only* $pk, \mathsf{params}, \boldsymbol{\tau}$*.*

FE Guess : Adversary $\mathcal{A}_{\mathsf{FE}}$ outputs a bit $a' \in \{0,1\}$.

BBG Guess : If $a = a'$, adversary $\mathcal{A}_{\mathsf{BBG}}$ outputs $b' = 0$. Otherwise, $\mathcal{A}_{\mathsf{BBG}}$ outputs $b' = 1$.

The constructed BBG adversary $\mathcal{A}_{\mathsf{BBG}}$ made $O(d\ell)$ private key query and zero decryption query to the BBG challenger $\mathcal{C}_{\mathsf{BBG}}$. Let $\mathsf{id}^* = \mathsf{ID}_\xi(x_\xi) = (I_1, \ldots, I_\ell) \in \left(\mathbb{Z}_p^*\right)^\ell$. Recall that the two BBG ciphertexts CT and $\boldsymbol{c}_\xi$ are

$$\mathsf{CT} = (A,\ B,\ C) = \left(\Omega^s \cdot m_b,\ g^s,\ \left(h_1^{I_1} \ldots h_\ell^{I_\ell} \cdot g_3\right)^s\right) \in \widetilde{\mathbb{G}} \times \mathbb{G}^2$$

$$\boldsymbol{c}_\xi = (\Omega^{s_\xi} \cdot \sigma_\xi,\ B,\ C) = \left(\Omega^{s_\xi} \cdot \sigma_\xi,\ g^s,\ \left(h_1^{I_1} \ldots h_\ell^{I_\ell} \cdot g_3\right)^s\right) \in \widetilde{\mathbb{G}} \times \mathbb{G}^2$$

where $\Omega^{s_\xi} = \Omega^s m_b \cdot m_0^{-1}$. If $b = 0$, then $s_\xi = s$ and thus $\boldsymbol{c}_\xi$ is a valid BBG encryption of σ_ξ under identity $\mathsf{ID}_\xi(x_\xi)$ with random coin s_ξ. Consequently, the FE scheme simulated by $\mathcal{A}_{\mathsf{BBG}}$ is *identical* to a real one from the view of $\mathcal{A}_{\mathsf{FE}}$ (even if $\mathcal{A}_{\mathsf{FE}}$ is computationally unbounded). If $b = 1$, then s_ξ is independent on s. As a result, in the FE scheme simulated by $\mathcal{A}_{\mathsf{BBG}}$, the challenging ciphertext $\mathsf{CT}_{\mathsf{FE}}$ is *independent* on the value of Msg_a. Note that adversary $\mathcal{A}_{\mathsf{FE}}$ does not know m_0, m_1.

Let the d-dimensional range $\mathbf{R} = \mathbf{A}_1 \times \ldots \times \mathbf{A}_d$. Define set $S_\#$:

$$S_\# = \{j \in [d] : \boldsymbol{x}^*[j] \notin \mathbf{A}_j\}$$

Since $\boldsymbol{x}^* \notin \mathbf{R}$, $S_\#$ is not empty and $|S_\#| \geq 1$. We have

$$\mathsf{Pr}[\neg \mathbf{E}_1] = \mathsf{Pr}[\xi \in S_\#] = \frac{|S_\#|}{d} \geq \frac{1}{d}.$$

Note that adversary $\mathcal{A}_{\mathsf{BBG}}$ has two terminal cases: (1) If event $\mathbf{E}_1$ occurs, $\mathcal{A}_{\mathsf{BBG}}$ outputs a random bit $b' \in \{0, 1\}$. (2) If event $\mathbf{E}_1$ does not occur, $\mathcal{A}_{\mathsf{BBG}}$ outputs $b' = 0$ iff $\mathcal{A}_{\mathsf{FE}}$ outputs $a' = a$.

In case of $\mathbf{E}_1$*:* Conditional on event $\mathbf{E}_1$, $\mathsf{Pr}[b = b'] = 1/2$.

In case of $\neg\mathbf{E}_1$*:* Suppose event $\mathbf{E}_1$ does not occur. Then $b' = 0 \Leftrightarrow a = a'$ and $b' = 1 \Leftrightarrow a \neq a'$. We have $\mathsf{Pr}[b' = 0 | b = 0] = \mathsf{Pr}[a = a' | b = 0]$ and $\mathsf{Pr}[b' = 1 | b = 1] = \mathsf{Pr}[a \neq a' | b = 1]$.

As a result, conditional on event $\neg\mathbf{E}_1$,

$$\begin{aligned}
\mathsf{Pr}[b = b'] &= \mathsf{Pr}[b = b' = 0] + \mathsf{Pr}[b = b' = 1] \\
&= \mathsf{Pr}[b = 0]\mathsf{Pr}[b' = 0 | b = 0] + \mathsf{Pr}[b = 1]\mathsf{Pr}[b' = 1 | b = 1] \\
&= \mathsf{Pr}[b = 0]\mathsf{Pr}[a = a' | b = 0] + \mathsf{Pr}[b = 1]\mathsf{Pr}[a \neq a' | b = 1] \\
&= \frac{1}{2} \times \left(\frac{1}{2} + \epsilon\right) + \frac{1}{2} \times \frac{1}{2} \\
&= \frac{1}{2} + \frac{1}{2}\epsilon
\end{aligned}$$

Combining the two cases ($\mathbf{E}_1$ and $\neg\mathbf{E}_1$), we obtain the advantage of $\mathcal{A}_{\mathsf{BBG}}$ against BBG scheme in the $\mathsf{IND\text{-}sID\text{-}CPA}$ game:

$$\mathsf{Adv}_{\mathsf{BBG}}^{\mathcal{A}_{\mathsf{BBG}}} + \frac{1}{2} = \mathsf{Pr}[\mathbf{E}_1] \times \frac{1}{2} + \mathsf{Pr}[\neg\mathbf{E}_1] \times \left(\frac{1}{2} + \frac{1}{2}\epsilon\right) = \frac{1}{2} + \frac{\epsilon}{2}\mathsf{Pr}[\neg\mathbf{E}_1] \geq \frac{1}{2} + \frac{\epsilon}{2d}$$

The adversary $\mathcal{A}_{\mathsf{BBG}}$ wins the game with probability at least $1/2 + \epsilon/(2d)$ using $O(d\ell)$ private key queries and running in time $t_{\mathsf{FE}} + O(d\ell \cdot t_{max} \cdot (N_{aq} + N_{enc}))$, where t_{max} is the maximum time for random sampling (within a space of size at most p), BBG encryption $\mathsf{Encrypt}$, or BBG key generation KeyGen, and N_{aq} (N_{enc}, respectively) is the number of anonymous delegation key queries (encryption key queries, respectively) made by $\mathcal{A}_{\mathsf{FE}}$.

5 Conclusion

In this paper, we constructed a new functional encryption scheme for multidimensional range query, using a new technique. The proposed functional encryption scheme allows a user with a valid ciphertext and a correct decryption key to obtain only a one-way function value of the plaintext, where the plaintext remains secure, assuming that the multidimensional point associated to the ciphertext is within the query range associated to the decryption key. Our functional encryption scheme is designed by exploiting the polymorphic property of existing BBG HIBE scheme: encryption of a message can be viewed as encryption of another message under a different key.

References

1. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional Encryption with Bounded Collusions via Multi-party Computation. In: CRYPTO '12: Annual International Cryptology Conference on Advances in Cryptology. 162–179
2. Boneh, D., Boyen, X., Goh, E.J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: EUROCRYPT '05: Annual International Conference on Advances in Cryptology. (2005) 440–456
3. Boneh, D., Sahai, A., Waters, B.: Functional Encryption: Definitions and Challenges. In: (*will appear in*) TCC '11: Theory of Cryptography Conference. (2011) `http://eprint.iacr.org/2010/543`.
4. O'Neill, A.: Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556 (2010) `http://eprint.iacr.org/`.
5. Lewko, A.B., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In: EUROCRYPT '10: Annual International Conference on Advances in Cryptology. (2010) 62–91
6. Okamoto, T., Takashima, K.: Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption. In: CRYPTO '10: Annual International Cryptology Conference on Advances in Cryptology. (2010) 191–208
7. Xu, J., Chang, E.C.: Authenticating aggregate range queries over multidimensional dataset. Cryptology ePrint Archive, Report 2010/050 (2010) `http://eprint.iacr.org/`.
8. Waters, B.: Functional Encryption for Regular Languages. In: CRYPTO '12: Annual International Cryptology Conference on Advances in Cryptology. 218–235
9. Barbosa, M., Farshim, P.: Semantically Secure Functional Encryption, Revisited. Cryptology ePrint Archive, Report 2012/474 (2012) `http://eprint.iacr.org/`.
10. Agrawal, S., Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional Encryption: New Perspectives and Lower Bounds. Cryptology ePrint Archive, Report 2012/468 (2012) `http://eprint.iacr.org/`.
11. Sahai, A., Seyalioglu, H.: Worry-free encryption: functional encryption with public keys. In: CCS '10: ACM conference on Computer and communications security. 463–472
12. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. SIAM J. Comput. **32**(3) (2003) 586–615
13. Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: EUROCRYPT '05: Annual International Conference on Advances in Cryptology. (2005) 457–473
14. Katz, J., Sahai, A., Waters, B.: Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. In: EUROCRYPT '08: Annual International Conference on Advances in Cryptology. (2008) 146–162
15. Shi, E., Bethencourt, J., Chan, T.H.H., Song, D., Perrig, A.: Multi-Dimensional Range Query over Encrypted Data. In: SP '07: IEEE Symposium on Security and Privacy. (2007) 350–364
16. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate Indistinguishability Obfuscation and Functional Encryption for All Circuits. SIAM J. Comput. **45**(3) (June 2016) 882–929

17. Garg, S., Gentry, C., Halevi, S., Zhandry, M.: Functional Encryption Without Obfuscation. In: TCC '16: Theory of Cryptography Conference. 480–511
18. Waters, B.: A Punctured Programming Approach to Adaptively Secure Functional Encryption. In: Advances in Cryptology – CRYPTO 2015. 678–697
19. GoldwasserS, S., Gordon, D., Goyal, V., Jain, A., Katz, J., Liu, F.H., Sahai, A., Shi, E., Zhou, H.S.: Multi-input Functional Encryption. In: Advances in Cryptology EUROCRYPT 2014. 578–602
20. Ben A. Fisch, Dhinakaran Vinayagamurthy, D.B.S.G.: Iron: Functional Encryption using Intel SGX. Cryptology ePrint Archive, Report 2016/1071 (2016) `http://eprint.iacr.org/2016/1071`.

A BBG HIBE

We restate the BBG HIBE scheme proposed by Boneh *et al.* [2], to make this paper self-contained. Let p be a λ bits safe prime, and $e : \mathbb{G} \times \mathbb{G} \to \widetilde{\mathbb{G}}$ be a bilinear map, where the orders of $\mathbb{G}$ and $\widetilde{\mathbb{G}}$ are both p. The HIBE scheme contains four algorithms ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}$), which are described as follows.

$\mathsf{Setup}(\ell)$

To generate system parameters for an HIBE of maximum depth ℓ, select a random generator $g \in \mathbb{G}$, a random $\alpha \in \mathbb{Z}_p$, and set $g_1 = g^\alpha$. Next, pick random elements $g_2, g_3, h_1, \ldots, h_\ell \in \mathbb{G}$. The public parameters and the master key are

$$\mathsf{params} = (g, g_1, g_2, g_3, h_1, \ldots, h_\ell, \Omega = e(g_1, g_2)), \quad \mathsf{master\text{-}key} = g_2^\alpha.$$

$\mathsf{KeyGen}(d_{\mathsf{id}|k-1}, \mathsf{id})$

To generate a private key d_{id} for an identity $\mathsf{id} = (I_1, \ldots, I_k) \in \left(\mathbb{Z}_p^*\right)^k$ of depth $k \le \ell$, using the master secret key $\mathsf{master\text{-}key}$, pick a random $r \in \mathbb{Z}_p$ and output

$$d_{\mathsf{id}} = \left(g_2^\alpha \cdot \left(h_1^{I_1} \ldots h_k^{I_k} \cdot g_3\right)^r,\ g^r,\ h_{k+1}^r, \ldots, h_\ell^r\right) \in \mathbb{G}^{2+\ell-k}$$

The private key for id can be generated incrementally, given a private key for the parent identity $\mathsf{id}_{|k-1} = (I_1, \ldots, I_{k-1}) \in \left(\mathbb{Z}_p^*\right)^{k-1}$. Let

$$d_{\mathsf{id}|k-1} = \left(g_2^\alpha \cdot \left(h_1^{I_1} \ldots h_{k-1}^{I_{k-1}} \cdot g_3\right)^{r'},\ g^{r'},\ h_k^{r'}, \ldots, h_\ell^{r'}\right) = (K_0, K_1, W_k, \ldots, W_\ell)$$

be the private key for $\mathsf{id}_{|k-1}$. To generate d_{id}, pick a random $t \in \mathbb{Z}_p$ and output

$$d_{\mathsf{id}} = \left(K_0 \cdot W_k^{I_k} \cdot \left(h_1^{I_1} \ldots h_k^{I_k} \cdot g_3\right)^t,\ K_1 \cdot g^t,\ W_{k+1} \cdot h_{k+1}^t, \ldots, W_\ell \cdot h_\ell^t\right).$$

This private key is a properly distributed private key for $\mathsf{id} = (I_1, \ldots, I_k)$ for $r = r' + t \in \mathbb{Z}_p$.

$\mathsf{Encrypt}(\mathsf{params}, \mathsf{id}, M; s)$

To encrypt a message $M \in \widetilde{\mathbb{G}}$ under the public key $\mathsf{id} = (I_1, \ldots, I_k) \in \left(\mathbb{Z}_p^*\right)^k$, pick a random $s \in \mathbb{Z}_p$ and output

$$\mathsf{CT} = \left(\Omega^s \cdot M,\ g^s,\ \left(h_1^{I_1} \ldots h_k^{I_k} \cdot g_3\right)^s\right) \in \widetilde{\mathbb{G}} \times \mathbb{G}^2. \tag{12}$$

$\mathsf{Decrypt}(d_{\mathsf{id}}, \mathsf{CT})$

Consider an identity $\mathsf{id} = (I_1, \ldots, I_k)$. To decrypt a given ciphertext $\mathsf{CT} = (A, B, C)$ using the private key $d_{\mathsf{id}} = (K_0, K_1, W_{k+1}, \ldots, W_\ell)$, output

$$A \cdot \frac{e(K_1, C)}{e(B, K_0)}.$$

For a valid ciphertext, we have

$$\frac{e(K_1, C)}{e(B, K_0)} = \frac{e\left(g^r, \left(h_1^{I_1} \ldots h_k^{I_k} \cdot g_3\right)^s\right)}{e\left(g^s, g_2^\alpha \left(h_1^{I_1} \ldots h_k^{I_k} \cdot g_3\right)^r\right)} = \frac{1}{e(g^s, g_2^\alpha)} = \frac{1}{e(g_1, g_2)^s} = \frac{1}{\Omega^s}. \qquad (13)$$