Authenticating Aggregate Range Queries over Multidimensional Dataset

Jia Xu, Ee-Chien Chang

National University of Singapore
Department of Computer Science
{xujia,changec}@comp.nus.edu.sg

Abstract. We are interested in the integrity of the query results from an outsourced database service provider. Alice passes a set $\mathbf{D}$ of d-dimensional points, together with some authentication tag $\mathbf{T}$, to an untrusted service provider Bob. Later, Alice issues some query over $\mathbf{D}$ to Bob, and Bob should produce the query result and a proof based on $\mathbf{D}$ and $\mathbf{T}$. Alice wants to verify the integrity of the query result with the help of the proof, using only the private key. In this paper, we consider aggregate query conditional on multidimensional range selection. In its basic form, a query asks for the total number of data points within a d-dimensional range. We are concerned about the number of communication bits required and the size of the tag $\mathbf{T}$. We give a scheme that requires $O(d^2 \log^2 N)$ communication bits to authenticate an aggregate count query conditional on d-dimensional range selection, where N is the number of points in the dataset. The security of our scheme relies on Generalized Knowledge of Exponent Assumption proposed by Wu and Stinson [1]. The low communication bandwidth is achieved due to a new functional encryption scheme, which exploits a special property of BBG HIBE scheme [2]. Besides counting, our scheme can be extended to support summing, finding of the minimum and usual (non-aggregate) range selection with similar complexity, and the proposed approach potentially can be applied to other queries by using suitable functional encryption schemes.

Keywords: Authentication, Multidimensional Aggregate Query, Secure Outsourced Database, Generalized Knowledge of Exponent Assumption, Functional Encryption

1 Introduction

Alice has a set $\mathbf{D}$ of d-dimensional points. She preprocesses the dataset $\mathbf{D}$ using her private key to generate some authentication tag $\mathbf{T}$. She sends (outsources) $\mathbf{D}$ and $\mathbf{T}$ to an untrusted service provider Bob. Then Alice deletes the original copy of dataset $\mathbf{D}$ and tag $\mathbf{T}$ from her local storage. Later Alice may issue a query over $\mathbf{D}$ to Bob, for example, an aggregate query conditional on a multidimensional range selection, and Bob should produce the query result and a proof based on $\mathbf{D}$ and $\mathbf{T}$. Alice wants to authenticate the query result, using only her private key. In this paper, we focus on count query conditional on a multidimensional range selection, that is, counting the number of points within a multidimensional rectangular range. Our solution can be extended to support other types of aggregate queries, like summing attributes of all points within the query range, and non-aggregate range selection, which asks for all points within the query range.

The problem we study in this paper fits in the framework of the outsourced database applications [3, 4], which emerged in early 2000s as an example of "software-as-a-service" (SaaS). By outsourcing database management, backup services and other IT needs to a professional service provider, companies can reduce expensive cost in purchase of equipments and even more expensive cost in hiring or training qualified IT specialists to maintain the IT services [5].

We are concerned about the communication cost and the storage overhead on Alice/Bob's side. Such requirements exclude the following straightforward approaches: (1) Bob sends back the whole dataset $\mathbf{D}$ with its tag $\mathbf{T}$; (2) Alice keeps a local copy of the dataset; (3) During preprocessing, Alice generates and signs answers to all possible queries. Recently, Gennaro *et al.* [6] and Chung *et al.* [7] showed that, in general any outsourced/delegated polynomial time function can be verified efficiently using a private verification key in the outsourced computation model, by using fully homomorphic encryption (e.g. Gentry [8]). Nevertheless, it is still meaningful to devise more efficient scheme for small class of functions, without using fully homomorphic encryption, as mentioned by Gennaro *et al.* [6].

There are many works on authenticating non-aggregate range selection query (e.g. [9, 10, 5, 11, 12, 13, 14, 15, 16]). Although there are efficient solutions for 1D and 2D range selection (e.g. Atallah *et al.* [13] for 2D grid dataset), solutions for higher dimension typically rely on geometric partitions which suffer from the "curse of dimensionality", leading to exponential communication overhead. Aggregate range query is arguably more challenging, and only a few works (e.g. [17, 18, 19]) are devoted to the authentication of aggregate query, which require high communication overhead for high dimensional dataset. Table 1 gives a comparison between our result and several previous works.

The design of our scheme consists of a few techniques. The first technique exploits the Generalized Knowledge of Exponent Assumption (**GKEA**) proposed by Wu and Stinson [1], to verify that the result is computed only from

Table 1: Worst case performance of different authentication schemes for aggregate range query or range selection query. This table consists of two parts: the first three rows are for aggregate query; the rest four rows are for range selection query.
Note: (1) The symbol "-" indicates that the authors do not provide such information in their paper. (2) We do not include Pang et al. *[17] and Cheng* et al. *[11] in this table, since no concise asymptotic bound are provided. Nevertheless, their performances are limited by their data structure, i.e. KD-tree [17] and R-Tree [11], which require exponential (in dimension) communication overhead in the worst case. (3) Our scheme supports private key verification, while the other works in this table support public key verification. (4) Our scheme can be extended to provide similar privacy protection like PDAS [18] (the extension is in our technique report [20]).*

Scheme	Dimension d	Communication overhead (bits)	Storage overhead	Computation (Verifier Alice)	Computation (Prover Bob)	Query	Techniques
PDAS [18]	$d = 1$	$O(\|S\| \log N)$	$O(N)$	$O(\|S\| \log N)$	$O(\|S\| + K^2)$	SUM,COUNT	Aggregated commitment + Shamir's Secret-Sharing Scheme
Li *et al.* [19]	$d \geq 1$	$O(dN + 2^d)$	$\Omega(dN)$	$O(dN + 2^d)$	$\Omega(N^{1-\frac{1}{d}})$	SUM or COUNT or MIN or MAX (*One authentication data structure per query type*)	MHT-like authentication structure for B-Tree/R-Tree
This paper and extension in our tech. report [20]	$d \geq 1$	$O(d^2 \log^2 \mathcal{Z})$	$O(dN)$	$O(d^2 \log^2 \mathcal{Z})$†	$O(dN \log \mathcal{Z})$‡	SUM,COUNT,MIN,MAX, MEDIAN	(customer designed) functional encryption + **GKEA** based homomorphic tag
Atallah *et al.* [13]	$d = 1, 2$	$O(1)$	$O(N)$	$O(\|S\|)$	$O(1)$	Range Selection	Precomputed prefix sum + **BLS** signature
Martel *et al.* [9]	$d \geq 1$	$O(\log^{d-1} N + \|S\|)$	-	-	-	Range Selection	Authentication Data Structure + Geometry Partition
Chen *et al.* [21]	$d \geq 1$	$O(\log^d \mathcal{Z})$	$O(N \log^d \mathcal{Z})$	$O(\log^d \mathcal{Z})$	$O(\log^d \mathcal{Z})$	Range Selection	Authentication Tree Structure + Access Control
This paper (Section 8.2)	$d \geq 1$	$O(d^2 \log^2 \mathcal{Z})$	$O(dN)$	$O(d^2 \log^2 \mathcal{Z} + \|S\|)$†	$O(dN \log \mathcal{Z} + \|S\|)$‡	Range Selection	(customer designed) functional encryption + **GKEA** based homomorphic tag

N: The number of tuples in the dataset.
K: The number of servers in PDAS [18].
†: $O(d^2 \log^2 \mathcal{Z})$ group multiplications.
S: The set of tuples satisfying the query condition.
$\mathcal{Z}$: The domain size of attributes/points in one dimension.
‡: $O(dN \log \mathcal{Z})$ bilinear map operations.

data points within the query range. This is achieved by first associating a secret number with each location in the space. Next, a homomorphic tag is computed by Alice from the secret number for each data point and kept in Bob's storage. To authenticate a query, Alice generates and sends to Bob *another* homomorphic tag for each location within the query range, based on the associated secret number and a random nonce. Bob aggregates these two types of tags for all data points within the query range, and sends the resulting aggregated values together with the query result to Alice. The aggregated values can be verified due to homomorphism, and it is difficult to forge the aggregated values using data points outside the query range, under Computational Diffie Hellman assumption and **GKEA**.

However, there are two main drawbacks if the above mentioned technique is employed by itself. Firstly, the validity of the aggregated tags do not rule out "over-counting" (where a data point is used more than once by Bob) and "under-counting" (where a data point is omitted by Bob). To prevent over-counting and under-counting, we further query for data points outside the original query range, and check for consistency between proofs and results of the queries.

The second drawback is the high communication overhead required—Alice has to send a tag for *each* location within the query range. In order to lower the communication complexity, we design a functional encryption scheme by exploiting a special property of BBG HIBE scheme [2]. Using this functional encryption scheme, Alice encrypts secret numbers associated with each data point, and sends the resulting ciphertexts to Bob during the setup. For each query, from the query range and the random nonce, Alice can generate a *short* decryption key. From the decryption key, Bob can decrypt those ciphertexts to obtain the tags for data points within the query range as the decrypted values, and learn nothing for data points outside the range, due to the property of our functional encryption scheme.

Contribution Our main contributions can be summarized as below:

1. We propose a functional encryption scheme in Section 5, by exploiting a special property (we call it "polymorphic property") of the BBG HIBE scheme [2]. Under this functional encryption scheme, given a message Msg and an identity $\boldsymbol{x}$, which is a d-dimensional point in domain $[1, \mathcal{Z}]^d$ where $\mathcal{Z}$ is an integer, a ciphertext can be generated

using the *private*[1] 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[2] of a private key is in $O(1)$, the size of a ciphertext is in $O(d)$, and the size of a decryption key is in $O(d \log^2 \mathcal{Z})$.

2. We prove that the proposed functional encryption scheme is weak-IND-sID-CPA secure (as defined in Section 5.3), if BBG HIBE scheme [2] is IND-sID-CPA secure (See Theorem 2).
3. We propose a scheme for aggregate count query in Section 6, by incorporating the functional encryption scheme into the preliminary scheme presented in Section 3. The resulting scheme is efficient. For a dataset $\mathbf{D}$ with N points in $[1, \mathcal{Z}]^d$ and a d-dimensional rectangular query range, round complexity is one per query, communication overhead is $O(d^2 \log^2 \mathcal{Z})$ bits per query, and the storage overheads on Alice/Bob's side are $O(1)$ and $O(dN)$, respectively. If the dataset $\mathbf{D}$ is *normalized*[3] [24], then $\mathcal{Z} = N$ and $O(d^2 \log^2 \mathcal{Z})$ is sublinear in N and polynomial in d. To the best of our knowledge, this is the first solution with worst case communication overhead sublinear in the number of points in the dataset and with polynomial storage overhead on server side, without using fully homomorphic encryption scheme [8,25]. We compare our result with several previous works in Table 1.
4. We prove that the proposed scheme is *correct* and *sound* (Theorem 3) under reasonable assumptions (Computational Diffie Hellman assumption, **GKEA** and ℓ-wBDHI assumption [2]). We describe our proof strategy in Section 7 and illustrate it by proving that the preliminary scheme in Section 3 is *correct* and *sound*. The full proof is in appendix.
5. Our scheme for count query leads to an efficient solution that authenticates multidimensional non-aggregate range selection query, with communication overhead $O(d^2 \log^2 \mathcal{Z})$. The method is described in Section 8.2 and performance is listed in Table 1.

For the clarity in presentation, although we focus on counting in this paper, our scheme can be extended to support other types of aggregate range query with similar complexity, including summing, and finding of minimum, maximum or median. Furthermore, our scheme can also be augmented to support dynamic dataset, provide privacy protection on data points, and prevent Alice from framing Bob. All of these extensions are reported in our technique report [20].

Organization The rest of this paper is organized as follows: Section 2 reviews related works. Section 3 gives a more detailed overview of our scheme. The problem formulation and security definition are presented in Section 4. We propose the functional encryption scheme in Section 5 and our authentication scheme in Section 6. We give the main theorem on the security of our authentication scheme and its proof outline in Section 7. After that, we analyze the performance of our authentication scheme in Section 8, and conclude the paper with Section 9.

2 Related work

Researches in secure outsourced database focus on two major aspects: (1) privacy (i.e. protect the data confidentiality against both the service provider and any third party) e.g. [4,26,27,28], and (2) integrity (i.e. authenticate the soundness and completeness of query results returned by the service provider) e.g. [3,9,29,10,5,17,30,11,12,31,13,32,14,15,16,18,19]. In the latter aspect, a lot of works are done for "identity query" [30], i.e. the query result is a subset of the database. Aggregate range query is arguably more challenging and only a few works (e.g. [17,18,19]) are devoted to the authentication of aggregate query.

There are roughly four categories of approaches for outsourced database authentication in the literature [3,9,29,10,5,17,30,11,12,31,13,32,14,15,16]. (1) Cryptographic primitives, like collision-resistant hash, (homomorphic and/or aggregatable) digital signature/commitment [5,33,18]. (2) Geometry partition and authenticated data structure [9,11,13,14,12,19]. For example, Merkle Hash Tree (typically for 1D case) and variants, KD-tree with chained signature [17], R-Tree with chained signature [11], and MHT-like authenticated B-Tree/R-Tree [19]. (3) Authenticated precomputed

[1] Unlike [22,23], our functional encryption scheme is a symmetric key encryption system.

[2] 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.

[3] For any dataset with size N, one can normalized [24] it by sorting the dataset along each dimension, so that the normalized dataset is a subset of $[1, N]^d$. We remark that such normalization will not loss generality: queries over the original dataset can be translated into queries over normalized dataset online by Bob and Alice can verify this translation by checking some authentication tags.

partial result, e.g. authenticated prefix sum [13, 19] (the static case solution in [19]) and authenticated partial sum hierarchy [17]. (4) Inserting and auditing fake tuples [31].

To the best of our knowledge, the existing few works (e.g. [17, 18, 19]) on authentication of aggregate query either only deal with 1D case, or have communication overhead linear (or even superlinear) w.r.t. the number of data points in the query range, and are suffering from the "curse of dimensionality". Even for multidimensional (non-aggregate) range selection query, the communication overhead is still in $O(\log^{d-1} N + |S|)$ (Martel *et al.* [9], Chen *et al.* [21]), where S is the set of data points within the query range, N is the number of data points in the dataset, and d is the dimension.

Recently, Gennaro *et al.* [6] and Chung *et al.* [7] proposed methods to authenticate *any* outsourced (or delegated) polynomial time function, based on fully homomorphic encryption [8, 25, 34]. They [6, 7] also gave a good discussion on why previous techniques (e.g. interactive proofs, probabilistic checkable proof (PCP), and interactive arguments) are insufficient for authenticating outsourced function from the performance point of view. If a function has input size Γ_1 and output size Γ_2, then both Gennaro *et al.* [6] and Chung *et al.* [7] have communication overhead in $\Omega(\Gamma_1 + \Gamma_2)$ to authenticate this function, where the hidden constant behind the big-Ω notation could be huge. The difference between their solutions and our work may become more clear when authenticating non-aggregate range selection query: Both Gennaro *et al.* [6] and Chung *et al.* [7] will require linear communication overhead, while our solution (the extension in Section 8.2) still requires $O(d^2 \log^2 \mathcal{Z})$ communication overhead.

Shi *et al.* [35] 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 range. There is a subtle but crucial difference between MRQED scheme and our implementation of functional encryption scheme [22, 23, 36, 37]: After a successful decryption, MRQED scheme reveals the message, whereas our functional encryption scheme reveals only a function value of the message and a nonce. As the nonce plays a crucial role in preventing replay attack, it is not suitable to adopt MRQED for our problem. On the other hand, MRQED has its own advantages over our functional encryption scheme, including that MRQED is a public key encryption scheme and has a stronger security model.

Several works [38, 39, 40, 41, 42, 43] in verification of integrity of data stored in remote storage server also adopted some homomorphic and/or aggregatable verification tags to achieve efficient communication cost.

3 Overview of Our Scheme

We illustrate our main ideas in two parts: (1) The first part presents a preliminary scheme that authenticates count query. This preliminary scheme is secure (Theorem 4) under Computational Diffie Hellman assumption and **GKEA**. However, it requires high communication and computation cost. (2) The second part describes the technique that reduces the communication and computation cost. In particular, the reduction is achieved using a functional encryption scheme which is constructed by exploiting a special property of BBG HIBE scheme [2].

3.1 Preliminary Scheme

Let $\mathbf{D} \subset [1, \mathcal{Z}]^d$ be a set of d-dimensional points. Let G be a cyclic multiplicative group of prime order p and $\{\mathsf{F}_s : [1, \mathcal{Z}]^d \to G\}_{s \in \{0,1\}^\kappa}$ be a pseudorandom function. During setup, Alice chooses at random $\beta \in \mathbb{Z}_p^*$, $\theta \in G$, and $s \in \{0, 1\}^\kappa$ as the private key. From the private key, Alice generates a tag value $t_x = (t_{x,1}, t_{x,2}) = (\theta \mathsf{F}_s(x), \mathsf{F}_s(x)^\beta)$ for each data point $x \in \mathbf{D}$. Alice also computes a value $\Delta = \prod_{x \in \mathbf{D}} t_{x,2}$. Next, Alice sends dataset $\mathbf{D}$ and tag values $\mathbf{T} = \{t_x : x \in \mathbf{D}\}$ to Bob and deletes everything except Δ and the private key (β, θ, s) from her storage.

Consider a count query conditional on a range $\mathbf{R} \subset [1, \mathcal{Z}]^d$, which asks for the size of $\mathbf{D} \cap \mathbf{R}$. Bob is expected to send to Alice a number X as the query result, and a proof to show that indeed $X = |\mathbf{D} \cap \mathbf{R}| \pmod p$.

To authenticate this query, Alice chooses two random nonces[4] ρ and $\hat{\rho}$, computes and sends auxiliary messages (called as *challenge-message*) $\Phi = \{\mathsf{F}_s(x)^\rho : x \in \mathbf{R}\}$ and $\hat{\Phi} = \{\mathsf{F}_s(x)^{\hat{\rho}} : x \in \mathbf{R}^\complement\}$ to Bob, where $\mathbf{R}^\complement = \{x \in [1, \mathcal{Z}]^d : x \notin \mathbf{R}\}$ is the complement set of range $\mathbf{R}$. Bob is expected to compute $X = |\mathbf{D} \cap \mathbf{R}|$ and $\hat{X} = |\mathbf{D} \cap \mathbf{R}^\complement|$, and generate the proof $(\Psi_1, \Psi_2, \Psi_3, \hat{\Psi}_1, \hat{\Psi}_2, \hat{\Psi}_3)$ as below:

Step B1: Bob multiplies all tags t_x for point $x \in \mathbf{D} \cap \mathbf{R}$ to obtain Ψ_1, Ψ_2

$$\Psi_1 \leftarrow \prod_{x \in \mathbf{D} \cap \mathbf{R}} t_{x,1} = \theta^X \prod_{x \in \mathbf{D} \cap \mathbf{R}} \mathsf{F}_s(x); \qquad \Psi_2 \leftarrow \prod_{x \in \mathbf{D} \cap \mathbf{R}} t_{x,2} = \prod_{x \in \mathbf{D} \cap \mathbf{R}} \mathsf{F}_s(x)^\beta.$$

[4] Here the secret random nonces prevent Bob from abusing this challenge-message for other queries.

Step B2: Bob multiplies all values $\mathsf{F}_s(x)^\rho$ from the challenge-message Φ for point $x \in \mathbf{D} \cap \mathbf{R}$ to obtain Ψ_3:

$$\Psi_3 \leftarrow \prod_{x \in \mathbf{D} \cap \mathbf{R}} \mathsf{F}_s(x)^\rho.$$

Step B3: Bob repeats Step B1 and Step B2 for data points $x \in \mathbf{D} \cap \mathbf{R}^{\mathsf{C}}$ using challenge-message $\hat{\Phi}$ to obtain $\hat{\Psi}_1, \hat{\Psi}_2, \hat{\Psi}_3$ correspondingly.

Bob sends back $(X, \Psi_1, \Psi_2, \Psi_3; \hat{X}, \hat{\Psi}_1, \hat{\Psi}_2, \hat{\Psi}_3)$ to Alice, and Alice verifies the returned message using the private key (β, θ, s), the secret random nonces $\rho, \hat{\rho}$, and value Δ in this way:

Step A1: Is (Ψ_1, Ψ_2) indeed an aggregated multiplication of valid tags?

$$\left(\frac{\Psi_1}{\theta^X}\right)^\beta \stackrel{?}{=} \Psi_2.$$

Step A2: Is Ψ_2 computed using *only* points inside $\mathbf{D} \cap \mathbf{R}$?

$$\Psi_2^\rho \stackrel{?}{=} \Psi_3^\beta.$$

Step A3: Repeat Step A1 and Step A2 to verify $(\hat{X}, \hat{\Psi}_1, \hat{\Psi}_2, \hat{\Psi}_3)$ using private key and secret random nonce $\hat{\rho}$.
Step A4: Is every point counted for *exactly* once?

$$\Psi_2 \cdot \hat{\Psi}_2 \stackrel{?}{=} \Delta. \quad (1)$$

If all of above verifications succeed, Alice accepts that X is the correct query result.

Remark.

– In the computations of Ψ_1, Ψ_2 and Ψ_3, an adversary (playing the role of Bob) may try to multiply tags for some points within $\mathbf{D} \cap \mathbf{R}$ multiple times, and/or ignore some points within $\mathbf{D} \cap \mathbf{R}$. That is, the adversary tries to find integers μ_x's for each point $x \in \mathbf{D}$, treat $(t_{x,1}^{\mu_x},\ t_{x,2}^{\mu_x})$ as the tag of point $x \in \mathbf{D}$ in the computations of Ψ_1 and Ψ_2, treat $\{\mathsf{F}_s(x)^{\rho \cdot \mu_x} : x \in \mathbf{D} \cap \mathbf{R}\}$ as the challenge-message in the computation of Ψ_3, and compute the query result $X = \sum_{x \in \mathbf{D} \cap \mathbf{R}} \mu_x$. Here $\mu_x > 1$ indicates that the point x is *over-counted*, $\mu_x < 1$ indicates that the point x is *under-counted*, and μ_x might take negative integer value. By doing so, the adversary can pass the verifications in Step A1, A2 and A3. However, if such adversary succeeds in passing Step A4, i.e. he can find integers μ_x's, for each $x \in \mathbf{D}$, such that $\prod_{x \in \mathbf{D}} t_{x,2}^{\mu_x} = \Delta = \prod_{x \in \mathbf{D}} t_{x,2}$, then he can solve **DLP** (Discrete Log Problem).
– The above attack is "restrictive". A stronger adversary may performance something else to pass the verifications. Fortunately, under **GKEA**, we can show that it is not restrictive: (informally) for any efficient adversary, under-counting and/or over-counting are his only ways to pass verifications in Step A1, A2 and A3.
– In the preliminary scheme, the size of challenge-message is linear w.r.t. $|\mathbf{R}|$, which can be very large, leading to large computation and communication cost.
– The second component $t_{x,2} = \mathsf{F}_s(x)^\beta$ in a tag t_x is required to deal with adaptive adversary: An adversary does not gain additional knowledge from adaptive learning, since it can generate challenge-message by itself from $\{\mathsf{F}_s(x)^\beta : x \in \mathbf{D}\}$, and the forged challenge-message is identically distributed as the challenge-message generated by Alice.

3.2 Deliver challenge-message efficiently and securely

To reduce complexity, Alice needs a way to deliver the information $\Phi = \{\mathsf{F}_s(x)^\rho : x \in \mathbf{R}\}$ (actually the subset $\{\mathsf{F}_s(x)^\rho : x \in \mathbf{D} \cap \mathbf{R}\}$ is sufficient to serve the purpose) to Bob by sending some auxiliary data of much smaller size, and Bob must not know the value of $\mathsf{F}_s(x)^\rho$ for point $x \notin \mathbf{R}$. We design such delivery method by exploiting a special property of existing HIBE scheme.

Polymorphic Property. 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 $(pk_j, sk_j, M_j, r_j), 1 \le j \le n$, such that for any $1 \le j \le n$, $(pk_j, sk_j) \in \mathsf{KeyGen}(1^\kappa)$ 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 different values M_j's using different decryption keys. 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. Hence, such polymorphic property may lead to a new way to construct functional encryption schemes [22, 23, 36, 37].

Overview of the Delivery Method. Alice can deliver the challenge-message $\{\mathsf{F}_s(x)^\rho : x \in \mathbf{D} \cap \mathbf{R}\}$ in this way: For simplicity, assume the dataset $\mathbf{D} \subset [1, \mathcal{Z}]$ consists of N 1D data points. Each point x in the domain is associated with an identity $\mathsf{ID}(x)$, which corresponds to a leaf node in the identity hierarchy tree. In the setup phase, for each data point $x_i \in \mathbf{D}$, Alice computes a ciphertext $\boldsymbol{c}_i$, which can be considered as encryption of $M_{i,j}$ under key $(pk_j, sk_j), j = 1, 2, 3, \ldots$ Alice sends these N ciphertexts $\{\boldsymbol{c}_i : 1 \leq i \leq N\}$ to Bob at the end of setup phase. Later, for a query range $\mathbf{R}$, Alice chooses a random nonce ρ and derives the delegation key $\boldsymbol{\delta}$ w.r.t. the set $S = \{\mathsf{ID}(x) : x \in \mathbf{R}\}$ of identities from the key pair (pk_ρ, sk_ρ), and sends $\boldsymbol{\delta}$ as challenge-message to Bob. With this delegation key, Bob is able to decrypt ciphertext $\boldsymbol{c}_i$ to obtain $M_{i,\rho}$ if $x_i \in \mathbf{D} \cap \mathbf{R}$. By carefully choosing parameters, we may have $M_{i,\rho} = \mathsf{F}_s(x_i)^\rho$ as desired.

The tree structure of the identity hierarchy of HIBE scheme facilitates short description of the delegation key, and thus the challenge-message, which originally contains $\mathcal{Z}$ subkeys in the worst case, can now be expressed with only $O(\log \mathcal{Z})$ subkeys.

For high dimensional cases, we perform the above procedure for each dimension, and combines the challenge-messages for all dimensions. The security of this method can be reduced to the $\mathsf{IND\text{-}sID\text{-}CPA}$ security of the underlying HIBE scheme.

4 Formulation

In this section, we formalize the problem and security model, and describe the security assumptions formally.

4.1 Dataset and Query

The dataset $\mathbf{D}$ is a set of N d-dimensional points $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N$ from the domain $[1, \mathcal{Z}]^d$ where $\mathcal{Z}$ can be a large integer (e.g. 64 bits integer). Let $\mathbf{R} = [a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_d, b_d] \subseteq [1, \mathcal{Z}]^d$ be a d-dimensional rectangular range. In this paper, we focus on queries that count the number of points in $\mathbf{D} \cap \mathbf{R}$. Let us define function F :

$$F(\mathbf{D}, \mathbf{R}) \quad \overset{\text{def}}{=} \quad |\mathbf{D} \cap \mathbf{R}| \; = \; |\mathbf{D} \cap \mathbf{R}| \mod p,$$

where p is a large prime. Note that p is exponential in the security parameter κ and N is polynomial in κ.

We write an integer interval $[1, n] = \{1, 2, 3, \ldots, n\}$ as $[n]$ for abbreviation, where n is some positive integer.

4.2 Security Model

We formulize our problem, as a variant of *Verifiable Computation* proposed by Gennaro *et al.* [6]. Let us view a query on a dataset as the function $F : \mathbb{D} \times \mathbb{R} \to \{0,1\}^*$, where $\mathbb{D}$ is the domain of datasets, $\mathbb{R}$ is the set of all possible queries, and the output of F is represented by a binary string. We define a *remote computing* protocol as follows:

Definition 1 ($\mathcal{RC}$) *A* Remote Computing *($\mathcal{RC}$) protocol for a function $F : \mathbb{D} \times \mathbb{R} \to \{0,1\}^*$, between Alice and Bob, consists of a setup phase and a query phase. The setup phase consists of a key generating algorithm* KGen *and data encoding algorithm* DEnc*; the query phase consists of a pair of interactive algorithms, namely the evaluator* Eval *and the extractor* Ext. *These four algorithms* $(\mathsf{KGen}, \mathsf{DEnc}, \langle \mathsf{Eval}, \mathsf{Ext} \rangle)$ *run in the following way:*

Setup Phase *:*

1. *Given security parameter κ, Alice generates a key K: $K \leftarrow \mathsf{KGen}(1^\kappa)$.*
2. *Alice encodes dataset $\mathbf{D} \in \mathbb{D}$: $(\mathbf{D}_{\mathtt{B}}, \mathbf{D}_{\mathtt{A}}) \leftarrow \mathsf{DEnc}(\mathbf{D}, K)$, then sends $\mathbf{D}_{\mathtt{B}}$ to Bob and keeps $\mathbf{D}_{\mathtt{A}}$.*

Query Phase *: The query phase consists of multiple query sessions. In each query session, Alice and Bob interact as below.*

1. *Alice selects a query $\mathbf{R} \in \mathbb{R}$.*
2. *Algorithm $\mathsf{Ext}(\mathbf{D}_{\mathtt{A}}, \mathbf{R}, K)$ on Alice's side, interacts with algorithm $\mathsf{Eval}(\mathbf{D}_{\mathtt{B}})$ on Bob's side to compute $(\zeta, X, \boldsymbol{\Psi}) \leftarrow \langle \mathsf{Eval}(\mathbf{D}_{\mathtt{B}}), \mathsf{Ext}(\mathbf{D}_{\mathtt{A}}, \mathbf{R}, K) \rangle$, where $\zeta \in \{\mathtt{accept}, \mathtt{reject}\}$ and $\boldsymbol{\Psi}$ is the proof of result X. If $\zeta = \mathtt{reject}$, then Alice rejects. Otherwise, Alice accepts that X is equal to $F(\mathbf{D}, \mathbf{R})$.*

We are interested in efficient $\mathcal{RC}$ protocol where the sizes of $K, \mathbf{D}_{\mathtt{A}}, \mathbf{D}_{\mathtt{B}}$, and the communication overhead are all small.

One of the main differences between $\mathcal{RC}$ model and Verifiable Computation model [6] is that: $\mathcal{RC}$ allows multiple rounds of communication between Alice and Bob to compute a function value in a query session; in contrast, Verifiable Computation model [6] allows only one round of communication. Although in this paper, both the scheme for count query in Section 6 and the extension for range selection query in Section 8.2 requires only one round of communication due to parallelism, our extension for min/max query in our technique report requires at least two rounds of communication, since dependencies between different rounds prevent parallelism.

We say a $\mathcal{RC}$ protocol is *verifiable*, if the following conditions hold: (1) Alice always accepts, when Bob follows the protocol honestly; (2) Alice rejects with o.h.p. (overwhelming high probability), when Bob returns a wrong result. Here we consider adversaries, i.e. malicious Bob, who are allowed to interact with Alice and learn for polynomial number of query sessions, before launching the attack. During the learning, the adversary may store whatever it has seen or learnt in a state variable.

Definition 2 **($\mathcal{VRC}$)** *Let κ be the security parameter. We call a $\mathcal{RC}$ protocol $\mathcal{E} = (\mathsf{KGen}, \mathsf{DEnc}, \langle \mathsf{Eval}, \mathsf{Ext}\rangle)$ w.r.t. function $F : \mathbb{D} \times \mathbb{R} \to \{0,1\}^*$ a* Verifiable Remote Computing *($\mathcal{VRC}$) protocol, if the following two conditions hold:*

- *correctness: for any $\mathbf{D} \in \mathbb{D}$, any $K \leftarrow \mathsf{KGen}(1^\kappa)$ and any $\mathbf{R} \in \mathbb{R}$, it holds that $\langle \mathsf{Eval}(\mathbf{D}_{\mathtt{B}}), \mathsf{Ext}(\mathbf{D}_{\mathtt{A}}, \mathbf{R}, K)\rangle = (\mathtt{accept},\ F(\mathbf{D}, \mathbf{R}),\ \boldsymbol{\Psi})$ for some $\boldsymbol{\Psi}$, where $(\mathbf{D}_{\mathtt{B}}, \mathbf{D}_{\mathtt{A}}) \leftarrow \mathsf{DEnc}(\mathbf{D}, K)$.*
- *soundness: for any adaptive PPT adversary $\mathcal{A}$, the advantage $\mathsf{Adv}_{\mathcal{E},\mathcal{A}}(1^\kappa) \le negl(\kappa)$,*

where $\mathsf{Adv}_{\mathcal{E},\mathcal{A}}(1^\kappa)$ is defined as

$$\mathsf{Adv}_{\mathcal{E},\mathcal{A}}(1^\kappa) \overset{\text{def}}{=} \mathsf{Pr}\left[\begin{matrix}(\zeta, X, \boldsymbol{\Psi}, \mathsf{view}_{\mathcal{A}}^{\mathcal{E}}, \mathbf{D}, \mathbf{R}) \leftarrow \mathsf{Exp}_{\mathcal{A}}^{\mathcal{E}}(1^\kappa) : \\ \zeta = \mathtt{accept} \ \wedge \ X \neq F(\mathbf{D}, \mathbf{R})\end{matrix}\right];$$

Experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathcal{E}}(1^\kappa)$

```
$\mathbf{D} \leftarrow \mathcal{A}(\mathsf{view}_{\mathcal{A}}^{\mathcal{E}})$;
$K \leftarrow \mathsf{KGen}(1^\kappa)$;
$(\mathbf{D}_{\mathtt{B}}, \mathbf{D}_{\mathtt{A}}) \leftarrow \mathsf{DEnc}(\mathbf{D}, K)$;
loop until $\mathcal{A}(\mathsf{view}_{\mathcal{A}}^{\mathcal{E}})$ decides to stop
    $\mathbf{R}_i \leftarrow \mathcal{A}(\mathbf{D}_{\mathtt{B}}, \mathsf{view}_{\mathcal{A}}^{\mathcal{E}})$;
    $(\zeta_i, X_i, \boldsymbol{\Psi}_i) \leftarrow \langle \mathcal{A}(\mathbf{D}_{\mathtt{B}}, \mathsf{view}_{\mathcal{A}}^{\mathcal{E}}), \mathsf{Ext}(\mathbf{D}_{\mathtt{A}}, \mathbf{R}_i, K)\rangle$;
$\mathbf{R} \leftarrow \mathcal{A}(\mathbf{D}_{\mathtt{B}}, \mathsf{view}_{\mathcal{A}}^{\mathcal{E}})$;
$(\zeta, X, \boldsymbol{\Psi}) \leftarrow \langle \mathcal{A}(\mathbf{D}_{\mathtt{B}}, \mathsf{view}_{\mathcal{A}}^{\mathcal{E}}), \mathsf{Ext}(\mathbf{D}_{\mathtt{A}}, \mathbf{R}, K)\rangle$;
Output $(\zeta, X, \boldsymbol{\Psi}, \mathsf{view}_{\mathcal{A}}^{\mathcal{E}}, \mathbf{D}, \mathbf{R})$.
```

The probability is taken over all random coins used by related algorithms, $negl(\cdot)$ is some negligible function, and $\mathsf{view}_{\mathcal{A}}^{\mathcal{E}}$ is a state variable[5] *describing all random coins chosen by $\mathcal{A}$ and all messages $\mathcal{A}$ can access during previous interactions with $\mathcal{E}$.*

4.3 Assumptions

Throughout the whole paper, let p be a κ bits safe prime, and $e : \mathbb{G} \times \mathbb{G} \to \widetilde{\mathbb{G}}$ be a bilinear map, where $\mathbb{G}$ and $\widetilde{\mathbb{G}}$ are two cyclic multiplicative groups of order p.

Assumption 1 (Computational Diffie Hellman Assumption) *For any PPT algorithm $\mathcal{A}$, it holds that*

$$\mathsf{Pr}\left[\mathcal{A}(g, g^a, g^b) = g^{ab}\right] \le \nu_1(\kappa),$$

where g is chosen at random from $\widetilde{\mathbb{G}}$, a and b are chosen at random from $\mathbb{Z}_p^*$, and $\nu_1(\cdot)$ is some negligible function.

The assumption **GKEA** is an extension of **KEA1** [44, 45, 46, 47, 48] and **KEA3** [49], and proposed by Wu and Stinson [1]. Roughly, **GKEA** assumption can be described as below:

> *For any adversary $\mathcal{A}$ that takes input $\{(u_i, u_i^\beta) : 1 \le i \le m\}$ and returns (U_1, U_2) with $U_1^\beta = U_2$, there exists an "extractor" $\bar{\mathcal{A}}$, which given the same inputs as $\mathcal{A}$ returns $\{\mu_i : 1 \le i \le m\}$, such that $\prod_{i=1}^m u_i^{\mu_i} = U_1$.*

Assumption 2 (Generalized KEA [44, 49, 1, 50]) *Let $\mathcal{A}$ and $\bar{\mathcal{A}}$ be two algorithms. We define the* **GKEA**-*advantage of $\mathcal{A}$ against $\bar{\mathcal{A}}$ as*

$$\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\mathbf{GKEA}}(\kappa) \overset{\text{def}}{=} \mathsf{Pr}\left[\begin{array}{l} W_m = \{(u_i, u_i^\beta) : i \in [m], u_i \xleftarrow{\$} \widetilde{\mathbb{G}}, \beta \xleftarrow{\$} \mathbb{Z}_p^*\} \\ (U_1, U_2) \leftarrow \mathcal{A}(W_m; r); \\ (\mu_1, \mu_2, \ldots, \mu_m) \leftarrow \bar{\mathcal{A}}(W_m; r, \bar{r}) : \\ \qquad U_1^\beta = U_2 \ \wedge \ U_1 \neq \prod_{j=1}^m u_j^{\mu_j} \end{array}\right], \tag{2}$$

[5] The adaptive adversary $\mathcal{A}$ may keep updating this state variable.

where the probability is taken over all random coins used and with m fixed (Here the notation $\xleftarrow{\$}$ denotes uniformly randomly sampling from a set. E.g. the expression $x \xleftarrow{\$} S$ means that x is uniformly randomly chosen from the set S). For any PPT algorithm $\mathcal{A}$ (called as adversary), there exists PPT algorithm $\bar{\mathcal{A}}$ (called as extractor), such that the **GKEA***-advantage of $\mathcal{A}$ against $\bar{\mathcal{A}}$ is upper bounded by some negligible function $\nu_2(\kappa)$, i.e. $\mathsf{Adv}^{\mathbf{GKEA}}_{\mathcal{A},\bar{\mathcal{A}}}(\kappa) \le \nu_2(\kappa)$, where m is polynomial in κ.*

Remark.

– **GKEA** is a natural extension of **KEA1** and **KEA3**, in the sense that **GKEA** $\Rightarrow$ **KEA3** $\Rightarrow$ **KEA1**. M. Abe and S. Fehr [51] proved **KEA1** and **KEA3** in *generic group model*. Following their techniques, **GKEA** can be proved in generic group model.
– If $u_i = g^{x^i}$ for each i with some random g and x, then Assumption 2 will become the q-**PKE** assumption proposed by J. Groth [50].

Furthermore, the (Decision) ℓ-wBDHI Assumption [2] is required for the $\mathsf{IND\text{-}sID\text{-}CPA}$ security of the underlying BBG HIBE scheme.

5 Functional Encryption Scheme

We construct a functional encryption [22,23] scheme by exploiting the polymorphic property of BBG HIBE scheme [2], following the overview given in Section 3.

5.1 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 $\mathsf{CT} = \widehat{\mathsf{CT}}$:

$$\begin{aligned}
&\mathsf{CT} = \mathsf{Encrypt}(\mathsf{params}, \mathsf{id}, M; s) = \left(\Omega^s \cdot M,\ g^s,\ \left(h_1^{I_1} \cdots h_k^{I_k} \cdot g_3\right)^s\right) \\
&\quad \text{under key: } \mathtt{params} = (g, g_1, g_2, g_3, h_1, \ldots, h_\ell, \Omega = e(g_1, g_2)),\ \ \mathtt{master\text{-}key} = g_2^{\alpha} \\
&\widehat{\mathsf{CT}} = \mathsf{Encrypt}(\widehat{\mathsf{params}}, \mathsf{id}, \widehat{M}; sz) = \left(\Omega^{sz} \cdot \widehat{M},\ \widehat{g}^{sz},\ \left(\widehat{h}_1^{I_1} \cdots \widehat{h}_k^{I_k} \cdot \widehat{g}_3\right)^{sz}\right), \\
&\quad \text{under key: } \widehat{\mathtt{params}} = (\widehat{g}, g_1, g_2, \widehat{g}_3, \widehat{h}_1, \ldots, \widehat{h}_\ell, \Omega = e(g_1, g_2)),\ \ \widehat{\mathtt{master\text{-}key}} = g_2^{\alpha z}
\end{aligned} \tag{3}$$

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. One can verify the above equality easily.

5.2 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 to associate identities to integers or integer intervals: (1) $\mathsf{ID}(\cdot)$ 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. (2) $\mathsf{IdSet}(\cdot)$ 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 the 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* [35]. 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 $[1, 2^\ell]$.

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 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 set $\{\mathtt{v}_j : \mathtt{v}_j \text{ is a tree node}, 1 \le j \le n\}$ such that the intervals associated to $\mathtt{v}_j$'s form a partition of $[a, b]$, and define $\mathsf{IdSet}_\iota([a, b])$ as the set $\{\mathsf{id}_j : \mathsf{id}_j \text{ is the identity associated to node } \mathtt{v}_j, 1 \le j \le n\}$. One can verify that the newly constructed mappings ID_ι and IdSet_ι satisfy the property mentioned in the beginning of Section 5.2. Furthermore, the set $\mathsf{IdSet}_\iota([a, b])$ contains $O(\ell)$ identities and each identity is a sequence of at most ℓ elements from $\mathbb{Z}_p^*$.

5.3 Construction of Functional Encryption Scheme based on HIBE

Let (Setup, KeyGen, Encrypt, Decrypt) be the BBG Hierarchical Identity Based Encryption (HIBE) scheme proposed by Boneh *et al.* [2] (the description of this scheme is in Appendix A). Based on this HIBE scheme, we construct a functional encryption scheme $\mathsf{FE} = (f\mathsf{Setup}, f\mathsf{Enc}, f\mathsf{KeyGen}, f\mathsf{Dec}, \mathsf{Mult})$ as in Figure 1.

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 Figure 1.

$$f_1(\mathsf{Msg}) = \Omega^{\mathsf{Msg}}; \qquad \forall \rho \in \mathbb{Z}_p^*,\ f_\rho(\mathsf{Msg}) = f_1(\mathsf{Msg})^\rho. \tag{4}$$

Lemma 1 *The functional encryption scheme* FE *described in Figure 1 satisfies these properties:*

(a) *For any* $(pk, sk) \leftarrow f\mathsf{Setup}(1^\kappa, d, \mathcal{Z})$*, for any message* $\mathsf{Msg} \in \mathbb{Z}_p^*$*, for any point* $\boldsymbol{x} \in [\mathcal{Z}]^d$*, for any rectangular range* $\mathbf{R} \subseteq [\mathcal{Z}]^d$*, if* $\mathsf{CT} \leftarrow f\mathsf{Enc}(\mathsf{Msg}, \boldsymbol{x}, sk)$ *and* $\boldsymbol{\delta} \leftarrow f\mathsf{KeyGen}(\mathbf{R}, \rho, sk)$*, then*

$$f\mathsf{Dec}(\mathsf{CT},\ \boldsymbol{x},\ \mathbf{R},\ \boldsymbol{\delta},\ pk) = \begin{cases} f_\rho(\mathsf{Msg}) & (\textit{if } \boldsymbol{x} \in \mathbf{R}) \\ \perp & (\textit{otherwise}) \end{cases} \tag{5}$$

(b) *For any* $(pk, sk) \leftarrow f\mathsf{Setup}(1^\kappa, d, \mathcal{Z})$*, 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* $y \in \widetilde{\mathbb{G}}$*, if* $\mathsf{CT} \leftarrow f\mathsf{Enc}(\mathsf{Msg}, \boldsymbol{x}, sk)$ *and* $\boldsymbol{\delta} \leftarrow f\mathsf{KeyGen}(\mathbf{R}, \rho, sk)$*, then*

$$f\mathsf{Dec}(\mathsf{Mult}(\mathsf{CT}, y, pk),\ \boldsymbol{x},\ \mathbf{R},\ \boldsymbol{\delta},\ pk) = \begin{cases} y \cdot f_\rho(\mathsf{Msg}) & (\textit{if } \boldsymbol{x} \in \mathbf{R}) \\ \perp & (\textit{otherwise}) \end{cases} \tag{6}$$

(The proof is in Appendix C.)

We formulize the security requirement of our functional encryption scheme by modifying the IND-sID-CPA security game [2]. The resulting weak-IND-sID-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 identity $\boldsymbol{x}^*$ from the identity 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: $\mathcal{C}$ chooses two plaintexts $\mathsf{Msg}_0, \mathsf{Msg}_1$ at random from the message space $\mathbb{Z}_p^*$, and 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 $\mathcal{A}$.
Learning Phase: $\mathcal{A}$ adaptively issues queries to $\mathcal{C}$, where each query is one of the following:

– Delegation key query $(\mathbf{R}, \rho)$, where $\boldsymbol{x}^* \notin \mathbf{R}$: $\mathcal{C}$ responds by running 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 $(\mathbf{R})$: $\mathcal{C}$ responds by choosing ρ at random from the function key space $\mathbb{Z}_p^*$ and running algorithm $f\mathsf{KeyGen}(\mathbf{R}, \rho, sk)$ to generate the delegation key $\boldsymbol{\delta}$, and sends $\boldsymbol{\delta}$ to $\mathcal{A}$.
– Encryption query $(\mathsf{Msg}, \boldsymbol{x})$: $\mathcal{C}$ responds by running $f\mathsf{Enc}(\mathsf{Msg}, \boldsymbol{x}, sk)$ 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 such an 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

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

Theorem 2 *(Informal) If the BBG HIBE scheme is* IND-sID-CPA *secure (as defined in [2]), then the functional encryption scheme* FE *constructed in Figure 1 is* weak-IND-sID-CPA *secure. That is, there is no PPT adversary that can win the* weak-IND-sID-CPA *game against the scheme* FE *with non-negligible advantage* $\mathsf{Adv}_{\mathsf{FE},\mathcal{A}}^{\mathsf{weak\text{-}IND\text{-}sID\text{-}CPA}}$. *(The formal statement of this theorem and its proof appear in Appendix D).*

Most of previous functional encryption schemes (e.g. attribute-based encryption [52], and predicate encryption [53]), if not all, allow the decryptor to obtain the original plaintext Msg in "good" case (e.g. if the attribute of plaintext and/or the decryption key satisfy the designated predicate) from a ciphertext of Msg, 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. Unlike [22,23], our functional encryption scheme is a symmetric key system. Our security formulation is weaker than previous works (e.g. [22,23]), but it is sufficient for our main result in Theorem 3.

6 The Main Construction

By incorporating the newly constructed functional encryption scheme FE into the preliminary scheme presented in Section 3, we construct a $\mathcal{RC}$ protocol $\mathcal{E} = (\mathsf{KGen}, \mathsf{DEnc}, \langle \mathsf{Eval}, \mathsf{Ext} \rangle)$ in Figure 2, to authenticate aggregate count query over multidimensional dataset.

Remark.

1. To understand the verifications in $\mathsf{CollRes}$, one may consider a homomorphic tag function Tag defined as below: Let y be the input, $\mathcal{K} = (\beta, \gamma, \theta)$ be the key, and v, w be random coins.

$$\mathsf{Tag}_{\mathcal{K}}(y; v, w) = (\theta^y v,\ v^\beta,\ w,\ v^\gamma w^\rho);$$

$$\prod_i \mathsf{Tag}_{\mathcal{K}}(y_i; v_i, w_i) = \mathsf{Tag}_{\mathcal{K}}\left(\sum_i y_i;\ \prod_i v_i,\ \prod_i w_i\right).$$

   Note that the first three component of $\mathsf{Tag}_{\mathcal{K}}(1; v_i, w_i)$ are just the three components of vector $\boldsymbol{t}_i$ generated in equation (8), and the fourth component $v^\gamma w^\rho$ is the output of $f\mathsf{Dec}(\mathsf{CT}_i, \boldsymbol{x}_i, \mathbf{R}, \boldsymbol{\delta}, pk)$ w.r.t. the random nonce ρ (i.e. $\boldsymbol{\delta} \leftarrow f\mathsf{KeyGen}(\mathbf{R}, \rho, sk)$). In the algorithm $\mathsf{CollRes}$, Bob computes the product of Tag values of data points within the query range as the proof of the query result X. Next, Alice verifies whether the proof $(\Psi_1, \Psi_2, \Psi_3, \Psi_4)$ is a valid Tag value for X, with key $\mathcal{K}$ and without knowing the values of random coins v_j's and w_j's.
2. A simple alternative construction of Tag is as follows, as in the preliminary scheme in Section 3:

$$\mathsf{Tag}'_{\mathcal{K}}(y; v) = (\theta^y v,\ v^\beta,\ v^\rho),$$

   where v^ρ is the output of $f\mathsf{Dec}$. However, we encounter difficulty in proving the security of the constructed scheme if we adopt Tag'. Thus we introduce a new random coin w and change Tag' to Tag. It is not clear whether such additional component plays a crucial role in achieving security or simply helps in simplifying the proof. The role of the additional random coin w_i is as follows: In our proof, given the first two components $\{(\theta^{y_i} v_i, v_i^\beta) : i \in [N]\}$ of all tag values, a simulator can simulate Alice in our scheme. Next, the simulator invokes a malicious Bob to interact with Alice to produce a forgery. For the alternative construction with Tag', to simulate DEnc, the simulator has to find a ciphertext for some message $W_i \in \mathbb{Z}_p^*$, such that $f_1(W_i) = \Omega^{W_i} = v_i^\beta$, which could be infeasible due to **DLP** (Discrete Log Problem). In our construction with Tag, since the additional term w_i is independent on the first two components of tag $\boldsymbol{t}_i$, the simulator can choose W_i freely, and generate $w_i \leftarrow f_1(W_i)$ and the ciphertext $\mathsf{CT}_i \leftarrow \mathsf{Mult}\left(f\mathsf{Enc}\left(W_i, \boldsymbol{x}_i, sk\right),\ v_i^{\beta\cdot\gamma'},\ pk\right)$ where $\gamma' \in \mathbb{Z}_p^*$ is randomly chosen. Consequently, if $\boldsymbol{x}_i \in \mathbf{R}$, then by Lemma 1, $f\mathsf{Dec}\left(\mathsf{CT}_i, \boldsymbol{x}_i, \mathbf{R}, f\mathsf{KeyGen}(\mathbf{R}, \rho, sk), pk\right) = v_i^{\beta\gamma'} f_\rho(W_i) = v_i^\gamma w_i^\rho$ as desired (taking γ as $\beta\gamma'$). Thus the simulation of DEnc can be done.
3. To ensure completeness and prevent over-counting or under-counting, we need to run $\mathsf{CollRes}$ on the complement query range $\mathbf{R}^{\complement}$. Since our functional encryption scheme FE only supports high dimensional *rectangular* ranges, we have to divide range $\mathbf{R}^{\complement}$ into multiple high dimensional rectangular ranges, and then run $\mathsf{CollRes}$ on each of them.
4. The $(2d + 1)$ invocations of $\mathsf{CollRes}$ can be executed in parallel. As a result, the round complexity of our scheme is exactly 1.
5. In Step 2 of ProVer, in the extreme case that $\mathbf{R}_\ell = \emptyset$, Alice can save the execution of $\mathsf{CollRes}$ on range $\mathbf{R}_\ell$, since Alice can predict the correct result ($\zeta_\ell = \mathtt{accept}, X_\ell = 0, \Psi_1 = \Psi_2 = \Psi_3 = \Psi_4 = 1 \in \widetilde{\mathbb{G}}$).
6. Like [6,7], in our scheme, Alice's accept/reject decisions are hidden from Bob.

Fig. 1: Construction of Functional Encryption Scheme $\mathsf{FE} = (f\mathsf{Setup}, f\mathsf{Enc}, f\mathsf{KeyGen}, f\mathsf{Dec}, \mathsf{Mult})$ based on BBG HIBE Scheme [2] (Setup, KeyGen, Encrypt, Decrypt)

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

1. Let $\ell = \lceil \log \mathcal{Z} \rceil$. Run algorithm $\mathsf{Setup}(\ell, \kappa)$ to obtain bilinear groups $(p, \mathbb{G}, \widetilde{\mathbb{G}}, e)$, public key $\mathtt{params} = (g, g_1, g_2, g_3, h_1, \ldots, h_\ell, \Omega = e(g_1, g_2))$ and private key $\mathtt{master\text{-}key} = g_2^\alpha$, such that p is a κ 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 5.2.
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).

$f\mathsf{Enc}(\mathsf{Msg}, \boldsymbol{x}, sk)$: `message Msg,` 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
$$\boldsymbol{c}_j \leftarrow \mathsf{Encrypt}(\mathtt{params},\ \mathsf{ID}_j(x_j),\ \sigma_j;\ s_j). \tag{7}$$
5. Output ciphertext $\mathsf{CT} = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_d)$.

$f\mathsf{KeyGen}(\mathbf{R}, \rho, sk)$: 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 $\mathbf{A}_1 \times \mathbf{A}_2 \ldots \times \mathbf{A}_d$, 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 $j \in [d]$, 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 $\mathsf{master\text{-}key}^{\rho\tau_j}$ 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 $\boldsymbol{\delta} = (\delta_1, \delta_2, \ldots, \delta_d)$.

$f\mathsf{Dec}(\mathsf{CT}, \boldsymbol{x}, \mathbf{R}, \boldsymbol{\delta}, pk)$: `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 $\mathbf{A}_1 \times \mathbf{A}_2 \ldots \times \mathbf{A}_d$, 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 $j \in [d]$, 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 $\widetilde{t} = \prod_{1 \le j \le d} \widetilde{t}_j$.

$\mathsf{Mult}(\mathsf{CT}', y, pk)$: `ciphertext CT',` $y \in \widetilde{\mathbb{G}}$

1. Let us write the ciphertext CT' as $(\boldsymbol{c}'_1, \ldots, \boldsymbol{c}'_d)$.
2. Choose d random elements $\eta_1, \ldots, \eta_d$ from $\widetilde{\mathbb{G}}$ with constraint $\prod_{j=1}^d \eta_j = y \in \widetilde{\mathbb{G}}$.
3. For each $j \in [d]$: parse $\boldsymbol{c}'_j$ as (A, B, C) and set $\boldsymbol{c}_j = (A \cdot \eta_j,\ B,\ C)$.
4. Output ciphertext $\mathsf{CT} = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_d)$.
   Note: Both $\mathbf{c}'_j$ and $\boldsymbol{c}_j$ are valid BBG ciphertexts for different plaintexts under the same identity.

Fig. 2: Construction of $\mathcal{RC}$ protocol $\mathcal{E} = (\mathsf{KGen}, \mathsf{DEnc}, \langle \widetilde{\mathsf{Eval}, \mathsf{Ext}} \rangle)$ based on functional encryption scheme FE constructed in Figure 1, where $\langle \mathsf{Eval}, \mathsf{Ext} \rangle$ (namely ProVer) invokes $\langle \widetilde{\mathsf{Eval}}, \widetilde{\mathsf{Ext}} \rangle$ (namely $\mathsf{CollRes}$) as a subroutine.

(Alice) $\mathsf{KGen}(1^\kappa)$:

Step 1: Run $f\mathsf{Setup}(1^\kappa)$ to obtain public/private key pair (pk', sk), where $pk' = (p, \mathbb{G}, \widetilde{\mathbb{G}}, e, \varOmega)$. Set $pk = (p, \mathbb{G}, \widetilde{\mathbb{G}}, e)$.

Note: e is a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \widetilde{\mathbb{G}}$, $\varOmega \in \widetilde{\mathbb{G}}$, and both $\mathbb{G}$ and $\widetilde{\mathbb{G}}$ are multiplicative groups of prime order p.

Step 2: Choose β, γ at random from $\mathbb{Z}_p^*$, and θ at random from $\widetilde{\mathbb{G}}$. Let $\mathcal{K} = (pk', sk, \beta, \gamma, \theta)$.

Step 3: Output $(\mathcal{K}, pk)$.

(Alice) $\mathsf{DEnc}(\mathbf{D}; \mathcal{K})$:

Step 1: Choose N random elements $W_1, \ldots, W_N$ from $\mathbb{Z}_p^*$ independently and N random elements $v_1, \ldots, v_N$ from $\widetilde{\mathbb{G}}$ independently.

Step 2: Dataset $\mathbf{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$. For each $i \in [N]$, generate tag $\boldsymbol{t}_i \in \widetilde{\mathbb{G}}^3$:

$$\boldsymbol{t}_i \leftarrow \left(\theta v_i,\ v_i^\beta,\ w_i = f_1(W_i)\right). \tag{8}$$

Note: Alice can evaluate functions $\{f_\rho(\cdot)\}_{\rho \in \mathbb{Z}_p^*}$, since Alice has $\varOmega \in \widetilde{\mathbb{G}}$.

Step 3: For each $i \in [N]$:

(a) Encrypt message W_i under point $\boldsymbol{x}_i$: $\mathsf{CT}'_i \leftarrow f\mathsf{Enc}(W_i, \boldsymbol{x}_i, sk)$;

(b) Apply the homomorphic property of FE to attach v_i^γ to ciphertext: $\mathsf{CT}_i \leftarrow \mathsf{Mult}(\mathsf{CT}'_i, v_i^\gamma, pk')$.

Step 4: Send $\mathbf{D}_\mathtt{B} = (\mathbf{D},\ \mathbf{T} = \{\boldsymbol{t}_i : i \in [N]\},\ \mathbf{C} = \{\mathsf{CT}_i : i \in [N]\},\ pk)$ to Bob, and keep *only* key $\mathcal{K}$ and $\mathbf{D}_\mathtt{A} = \left(N, d, \varDelta = \prod_{i \in [N]} v_i^\beta\right)$ in local storage.

(Alice, Bob) $\mathsf{ProVer} = \langle \mathsf{Eval}(\mathbf{D}_\mathtt{B}),\ \mathsf{Ext}(\mathbf{D}_\mathtt{A},\ \mathbf{R},\ \mathcal{K}) \rangle$: $\mathbf{D}_\mathtt{A} = (N, d, \varDelta)$, $\mathbf{D}_\mathtt{B} = (\mathbf{D}, \mathbf{T}, \mathbf{C}, pk)$

Precondition: The query range $\mathbf{R} \subset [\mathcal{Z}]^d$ is a rectangular range.

Step 1: Alice partitions the complement range $\mathbf{R}^\mathtt{C}$ into $2d$ rectangular ranges $\{\mathbf{R}_\ell \subset [\mathcal{Z}]^d : \ell \in [1, 2d]\}$, and sets $\mathbf{R}_0 = \mathbf{R}$.

Step 2: For $0 \le \ell \le 2d$, Alice and Bob invokes $\mathsf{CollRes}$ on range $\mathbf{R}_\ell$. Denote the output as $(\zeta_\ell, X_\ell, \varPsi_2^{(\ell)})$.

Step 3: Alice sets $\zeta = \mathtt{accept}$, if the following equalities hold

$$\forall 0 \le \ell \le 2d, \zeta_\ell \stackrel{?}{=} \mathtt{accept}, \qquad \prod_{0 \le \ell \le 2d} \varPsi_2^{(\ell)} \stackrel{?}{\equiv} \varDelta; \tag{9}$$

otherwise sets $\zeta = \mathtt{reject}$. Alice outputs $(\zeta, X_0, \varDelta)$.

(Alice, Bob) $\mathsf{CollRes} = \left\langle \widetilde{\mathsf{Eval}}(\mathbf{D}_\mathtt{B}),\ \widetilde{\mathsf{Ext}}(\mathbf{D}_\mathtt{A},\ \mathbf{R},\ \mathcal{K}) \right\rangle$: $\mathbf{D}_\mathtt{A} = (N, d, \varDelta)$, $\mathbf{D}_\mathtt{B} = (\mathbf{D}, \mathbf{T}, \mathbf{C}, pk)$

Precondition. The query range $\mathbf{R} \subset [\mathcal{Z}]^d$ is a rectangular range.

Step A1: (Alice's first step) Alice chooses a random nonce ρ from $\mathbb{Z}_p^*$ and produces challenge-message $\boldsymbol{\delta}$ for range $\mathbf{R}$ by running algorithm $f\mathsf{KeyGen}$: $\boldsymbol{\delta} \leftarrow f\mathsf{KeyGen}(\mathbf{R}, \rho, sk)$. Alice sends $(\mathbf{R}, \boldsymbol{\delta})$ to Bob.

Step B1: (Bob's first step) Bob computes the query result X and proof $(\varPsi_1, \varPsi_2, \varPsi_3, \varPsi_4)$ as follows

$$X \leftarrow |\mathbf{D} \cap \mathbf{R}|; \qquad (\varPsi_1, \varPsi_2, \varPsi_3) \leftarrow \bigotimes_{\boldsymbol{x}_i \in \mathbf{D} \cap \mathbf{R}} \boldsymbol{t}_i; \qquad \varPsi_4 \leftarrow \prod_{\boldsymbol{x}_i \in \mathbf{D} \cap \mathbf{R}} f\mathsf{Dec}(\mathsf{CT}_i, \boldsymbol{x}_i, \mathbf{R}, \boldsymbol{\delta}, pk). \tag{10}$$

Bob sends $(X, \varPsi_1, \varPsi_2, \varPsi_3, \varPsi_4)$ to Alice.

Note: For $\boldsymbol{x}_i \in \mathbf{D} \cap \mathbf{R}$, $f\mathsf{Dec}(\mathsf{CT}_i, \boldsymbol{x}_i, \mathbf{R}, \boldsymbol{\delta}, pk)$ is supposed to output $v_i^\gamma f_1(W_i)^\rho = v_i^\gamma w_i^\rho$; the operator $\bigotimes$ denotes component-wise multiplication of vectors of the same dimension.

Step A2: (Alice's second step) Let $\varLambda \leftarrow \frac{\varPsi_1}{\theta^X}$. Alice sets $\zeta = \mathtt{accept}$, if the following equalities hold

$$\varLambda^\beta \stackrel{?}{=} \varPsi_2, \qquad \varLambda^\gamma \varPsi_3^\rho \stackrel{?}{=} \varPsi_4. \tag{11}$$

Otherwise sets $\zeta = \mathtt{reject}$. Alice outputs $(\zeta, X, \varPsi_2)$.

7 Security Analysis

7.1 Our main theorem

Theorem 3 (Main Theorem) *Suppose Assumption 1 and Assumption 2 hold, and* BBG *[2] HIBE scheme is* IND-sID-CPA *secure. Then the* $\mathcal{RC}$ *protocol* $\mathcal{E} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{ProVer})$ *constructed in Figure 2 is* $\mathcal{VRC}$ *w.r.t. function* $F(\cdot, \cdot)$ *as defined in Section 4.1, under Definition 2. Namely,* $\mathcal{E}$ *is* correct *and* sound *w.r.t. function* F*. (The proof is in appendix.)*

7.2 Overview of Proof

To process a query, our scheme (particularly the algorithm ProVer) invokes $(2d+1)$ instances of interactive algorithm $\mathsf{CollRes}$. In each instance of $\mathsf{CollRes}$, Bob is supposed to return a 5-tuple $(X, \Psi_1, \Psi_2, \Psi_3, \Psi_4)$ where X is the query result and $(\Psi_1, \Psi_2, \Psi_3, \Psi_4)$ is the proof, and Alice will verify whether the proof is valid w.r.t. the query result. Furthermore, after all of $(2d+1)$ invocations, Alice will perform one additional verification (equation (9)) to ensure completeness and prevent over-counting or under-counting. In order to fool Alice with a wrong query result, an adversary has to provide a valid 5-tuple for each invocation of $\mathsf{CollRes}$ and pass the equation (9). Therefore, an adversary against $\mathcal{E} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{ProVer})$ is also an adversary against $\widetilde{\mathcal{E}} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{CollRes})$.

We consider various types of PPT adversaries against $\mathcal{E}$ or $\widetilde{\mathcal{E}}$, which interacts with Alice by playing the role of Bob and intends to output a wrong query result and a forged but valid proof:

- Type I adversary: This adversary is not confined in any way in its attack strategy and produces a 5-tuple $(X, \Psi_1, \Psi_2, \Psi_3, \Psi_4)$ on a query range $\mathbf{R}$.
- Type II adversary: A restricted adversary which can produce the same forgery[6] from the same input as Type I adversary, and can find N integers[7] μ_i's, $1 \le i \le N$, such that $\Psi_2 = \prod_{i\in[N]} \left(v_i^{\beta}\right)^{\mu_i}$, where β and v_i's are as in Figure 2.
- Type III adversary: The same as Type II adversary, with additional constraint: $\mu_i = 0$ for $\boldsymbol{x}_i \in \mathbf{D} \cap \mathbf{R}^{\mathsf{C}}$.
- Type IV adversary: The same as Type III adversary, with additional constraint: $\mu_i = 1$ for $\boldsymbol{x}_i \in \mathbf{D} \cap \mathbf{R}$.

Note that the Type II (or Type III, Type IV) adversary *explicitly* outputs $\{\mu_1, \ldots, \mu_N\}$, and *implicitly* outputs $(X, \Psi_1, \Psi_2, \Psi_3, \Psi_4)$ which is exactly the output of the corresponding Type I adversary. This is similar to **KEA** extractor and **KEA** adversary.

Basically, our proof framework is like this:

- Lemma 6: The existence of Type I adversary implies the existence of Type II adversary, under **GKEA** Assumption 2, where Type I adversary is a counterpart of adversary $\mathcal{A}$ in **GKEA** and Type II adversary is a counterpart of the extractor $\bar{\mathcal{A}}$ in **GKEA**.
- Theorem 7: If there exists a Type II adversary which is not in Type III, then there exists a PPT algorithm to break the weak-IND-sID-CPA security of the functional encryption scheme FE.
- Theorem 8: If there exists a Type III adversary which is not in Type IV, then there exists a PPT algorithm to break Discrete Log Problem.
- (Part of)Theorem 3: If there exists a Type IV adversary which breaks our scheme, then there exists a PPT algorithm to break Assumption 1.

Informally, by combining all together, Theorem 3 states that if there exists a Type I adversary which outputs result X and a valid proof, then X has to be equal to the correct query result with o.h.p, under related computational assumptions. Note that Lemma 6 and Theorem 7 focus on the partial scheme $\widetilde{\mathcal{E}}$ and Theorem 8 and Theorem 3 focus on the whole scheme $\mathcal{E}$.

The structure of our proof or the relationships among all assumptions, lemmas and theorems are shown as below. Note that the proof of correctness (Lemma 1) does not rely on any computational assumption.

$$\left.\begin{array}{r}\left.\begin{array}{r}\left.\begin{array}{r}\textbf{Assumption } 1\\ \textsf{BBG is IND-sID-CPA secure [2]} \Rightarrow \textbf{ Theorem } 2\\ \textbf{Assumption } 2 \Rightarrow \textbf{ Lemma } 5 \Rightarrow \textbf{ Lemma } 6\end{array}\right\} \Rightarrow \textbf{ Theorem } 7\\ \textbf{Assumption } 1 \Rightarrow \textbf{DLP Assumption}\end{array}\right\} \Rightarrow \textbf{Theorem } 8\\ \\ \textbf{Assumption } 1\\ \textbf{Lemma } 1\end{array}\right\} \Rightarrow \textbf{Theorem } 3$$

[6] This is possible, if the Type II adversary just invokes Type I adversary as a subroutine using the same random coin.

[7] Note that μ_i can take negative integer value, and $\mu_i > 1$ ($\mu_i < 1$, respectively) corresponds to the case of over-counting (under-counting, respectively) point $\boldsymbol{x}_i$.

7.3 The Preliminary Scheme is Secure

In this subsection, we prove that the preliminary scheme described in Section 3 is secure, following the proof framework in Section 7.2. This proof sketch serves as an illustration of our proof strategy and as a warm up of our full proof for the main scheme in appendix.

Theorem 4 *Suppose Assumption 1 and Assumption 2 hold for the cyclic multiplicative group G of order p and $\mathsf{F}_s(\cdot)$ is a random oracle. The preliminary scheme described in Section 3 is a $\mathcal{VRC}$ w.r.t. function $F(\cdot,\cdot)$ as defined in Section 4.1, under Definition 2. Namely, the preliminary scheme is* correct *and* sound *w.r.t. function F.*

Proof (sketch of Theorem 4). The correctness part is straightforward. We just focus on soundness.
Part I: *The existence of Type I adversary implies the existence of Type II adversary, under **GKEA** Assumption 2.* Suppose there exists Type I adversary $\mathcal{B}$ against the preliminary scheme. We try to construct a Type II adversary $\bar{\mathcal{B}}$ based on **GKEA** Assumption. We follow the proof framework for the statement that **KEA3** implies **KEA1** by Bellare *et al.* [49]. First, we construct a **GKEA** adversary $\mathcal{A}_1$ based on the Type I adversary $\mathcal{B}$:

Construction of **GKEA** adversary $\mathcal{A}_1$ based on the Type I adversary $\mathcal{B}$

1. The input is $\{(u_i, u_i^\beta) : u_i \in G, 1 \le i \le m\}$, where $\beta \in \mathbb{Z}_p$ is unknown.
2. Choose two independent random elements $R_1, R_2 \in G$. There exist some unknown $\theta, v_0 \in G$, such that $R_1 = \theta v_0, R_2 = v_0^\beta$.
3. Let $\mathbf{D} = \{x_1, x_2, \ldots, x_{m+1}\} \subset [\mathcal{Z}]^d$ be the dataset. Let $u_{m+1} = 1$. Define function F_s: For any $x_i \in \mathbf{D}$, $\mathsf{F}_s(x_i) = u_i v_0$; for any $x \in [\mathcal{Z}]^d \setminus \mathbf{D}$, choose $z_x \in \mathbb{Z}_p^*$ at random and set $\mathsf{F}_s(x) = u_1^{z_x}$. *Note that $\mathsf{F}_s(x)^\beta$ still can be computed, although β is unknown.*
4. Invoke the preliminary scheme (Alice's part) with parameters β, θ and function F_s. *Note that tag $t_i = (\theta \mathsf{F}_s(x_i), \mathsf{F}_s(x_i)^\beta) = (u_i R_1, u_i^\beta R_2)$ still can be computed without knowing the values of $\theta, \beta, \mathsf{F}_s(x_i)$.*
5. Invoke the adversary $\mathcal{B}$ (Bob's part) to interact with Alice. For any query $\mathbf{R}$ made by $\mathcal{B}$, generate challenge-message from $\{\mathsf{F}_s(x)^\beta\}$ in this way: choose $\rho' \in \mathbb{Z}_p^*$ at random, and send $\{\mathsf{F}_s(x)^{\beta\rho'} : x \in \mathbf{R}\}$. *Note the actual random nonce $\rho = \beta\rho'$ is unknown.*
6. Obtain output $(X, \Psi_1, \Psi_2, \Psi_3)$ from $\mathcal{B}$, and output $\left(\frac{\Psi_1}{R_1^X}, \frac{\Psi_2}{R_2^X}\right)$.

If the adversary $\mathcal{B}$'s output $(X, \Psi_1, \Psi_2, \Psi_3)$ can pass Alice's verification step 1, i.e. $\left(\frac{\Psi_1}{\theta^X}\right)^\beta = \Psi_2$, then the **GKEA** adversary $\mathcal{A}_1$'s output is valid:

$$\left(\frac{\Psi_1}{R_1^X}\right)^\beta = \frac{\Psi_1^\beta}{\theta^{X\beta} v_0^{X\beta}} = \frac{\Psi_2}{v_0^{X\beta}} = \frac{\Psi_2}{R_2^X}.$$

By **GKEA** Assumption, there exists an extractor $\bar{\mathcal{A}}_1$, which outputs $\{\mu_i : 1 \le i \le m\}$ from the same input[8] of $\mathcal{A}$, such that $\frac{\Psi_2}{R_2^X} = \prod_{i=1}^m u_i^{\beta\mu_i}$. Then we can construct an adversary $\mathcal{B}_2$ based on $\bar{\mathcal{A}}_1$ which just outputs $\{\mu_i : 1 \le i \le m+1\}$, where $\mu_{m+1} = X - \sum_{i=1}^m \mu_i \mod p$. We conclude that $\mathcal{B}_2$ is a Type II adversary against the preliminary scheme, since

$$\prod_{i=1}^{m+1} \mathsf{F}_s(x_i)^{\beta\mu_i} = \mathsf{F}_s(x_{m+1})^{\beta\mu_{m+1}} \prod_{i=1}^{m} \mathsf{F}_s(x_i)^{\beta\mu_i} = R_2^{X-\sum_{i=1}^m \mu_i} \prod_{i=1}^{m} \left(u_i^\beta R_2\right)^{\mu_i} = R_2^X \prod_{i=1}^{m} u_i^{\beta\mu_i} = \Psi_2.$$

Part II: *If there exists a Type II adversary which is not in Type III, then there exists a PPT algorithm to break Computational Diffie Hellman Assumption 1.*

Construction of adversary $\mathcal{A}_2$ against Computational Diffie Hellman Problem

1. The input is $(v, v^a, u) \in G^3$ where $a \in \mathbb{Z}_p$. The goal is to find u^a.
2. Define function F_s: For each $x_i \in \mathbf{D}$, choose z_i at random from $\mathbb{Z}_p$ and set $\mathsf{F}_s(x_i) = v^{z_i} \in G$.
3. Choose i^* from $[N]$ at random and redefine $\mathsf{F}_s(x_{i^*})$: $\mathsf{F}_s(x_{i^*}) = u$.
4. Invoke the preliminary scheme (Alice's part) with function F_s and invoke the Type II adversary (Bob's part) to interact with Alice. The adversary's adaptive queries can be answered in the same way as in Step 5 of algorithm $\mathcal{A}_1$.
5. Let $\mathbf{R}$ be the adversary's challenging query range. If $x_{i^*} \in \mathbf{R}$, abort and fail. Otherwise, generate challenge-message with random nonce $\rho = a$: the value $\mathsf{F}_s(x_i)^a = (v^a)^{z_i}$ for $x_i \in \mathbf{R}$ can be computed, although a is unknown.

[8] Including the random coin.

6. Let $(X, \Psi_1, \Psi_2, \Psi_3, \mu_1, \ldots, \mu_N)$ be the output of adversary. If $\mu_{i^*} \neq 0$, then compute φ as below and output $\varphi^{\mu_{i^*}^{-1}}$ *(This is the success case).*
$$\varphi \leftarrow \frac{\Psi_3}{\prod_{1 \le i \le N}^{i \neq i^*} \mathsf{F}_s(x_i)^{a\mu_i}}$$
Otherwise, abort and fail.

Let $S_{\#} = \{i : \mu_i \neq 0, x_i \in \mathbf{D} \cap \mathbf{R}^{\complement}\}$. Since the adversary is not in Type III, $S_{\#} \neq \emptyset$. It is easy to verify that, in the success case, i.e. when the adversary's output pass Alice's verifications and $\Psi_2 = \prod_{i=1}^{N} \mathsf{F}_s(x_i)^{\beta\mu_i}$ and $i^* \in S_{\#}$, then the output $\varphi^{\mu_{i^*}^{-1}} = \mathsf{F}_s(x_{i^*})^a = u^a$. Since the index i^* is uniformly random in $[N]$ and tags for all N points are identically distributed, there is non-negligible probability that the success case will be reached, and thus the value of u^a can be found.
Part III: *If there exists a Type III adversary which is not in Type IV, then there exists a PPT algorithm to break Discrete Log Assumption.*

Construction of adversary $\mathcal{A}_3$ against Discrete Log Problem

1. The input is $(v, v^a) \in G^2$. The goal is to find $a \in \mathbb{Z}_p$.
2. Define function F_s: For each $x_i \in \mathbf{D}$, choose y_i, z_i at random from $\mathbb{Z}_p$ and set $\mathsf{F}_s(x_i) = (v^a)^{y_i} \cdot v^{z_i} \in G$; otherwise, set $\mathsf{F}_s(x)$ to a random number in G.
3. Invoke the preliminary scheme (Alice's part) with function F_s and invoke the Type III adversary (Bob's part) to interact with Alice. The adversary's adaptive queries can be answered in the same way as in the preliminary scheme. *Note the simulator has all of private key.*
4. Let $\mathbf{R}$ be the challenging query range. Let $(X, \Psi_1, \Psi_2, \Psi_3, \{\mu_i : x_i \in \mathbf{D} \cap \mathbf{R}\})$ be the output of adversary for range $\mathbf{R}$ and let $(\hat{X}, \hat{\Psi}_1, \hat{\Psi}_2, \hat{\Psi}_3, \{\mu_i : x_i \in \mathbf{D} \cap \mathbf{R}^{\complement}\})$ be the output of adversary for the complement range $\mathbf{R}^{\complement}$.
5. If the adversary succeeds, we have
$$\prod_{i=1}^{N} \mathsf{F}_s(x_i)^{\beta\mu_i} = \prod_{x_i \in \mathbf{D}\cap\mathbf{R}} \mathsf{F}_s(x_i)^{\beta\mu_i} \prod_{x_i \in \mathbf{D}\cap\mathbf{R}^{\complement}} \mathsf{F}_s(x_i)^{\beta\mu_i} = \Psi_2 \cdot \hat{\Psi}_2 = \Delta = \prod_{i=1}^{N} \mathsf{F}_s(x_i)^{\beta}$$
6. Since the adversary is not Type IV, there exists some i, such that $\mu_i \neq 1$. Consequently, a univariable equation in unknown a of order 1 can be formed from the above equation. Solve this equation to get root a' and output a'.

Note that Computational Diffie Hellman Assumption 1 implies Discrete Log Assumption.
Part IV: *If there exists a Type IV adversary which breaks our scheme, then there exists a PPT algorithm to break Assumption 1.* Given input (u, u^β, v^β), we can construct an algorithm to find v. Choose a random number R. There exists some θ, such that $R = \theta v$. Similar as the construction of **GKEA** adversary $\mathcal{A}_1$ in Part I, from input $(u, u^\beta, \theta v, v^\beta)$, we can simulate the preliminary scheme (Alice's part). Let $\mathbf{R}$ be the challenging query range. let $(X', \Psi'_1, \Psi'_2, \Psi'_3, \{\mu_i : x_i \in \mathbf{D}\cap\mathbf{R}\})$ be an output of a Type IV adversary on query $\mathbf{R}$ and let $(X, \Psi_1, \Psi_2, \Psi_3, \{\mu_i : x_i \in \mathbf{D}\cap\mathbf{R}\})$ be the output of an honest Bob on query $\mathbf{R}$. If the Type IV adversary succeeds, then $\Psi'_2 = \prod_{x_i \in \mathbf{D}\cap\mathbf{R}} \mathsf{F}_s(x_i)^{\beta\mu_i}$, where $\mu_i = 1, 1 \le i \le N$, and $\left(\frac{\Psi'_1}{\theta^{X'}}\right)^\beta = \Psi'_2$. On the other hand, the output from an honest Bob also passes Alice's verifications: $\left(\frac{\Psi_1}{\theta^X}\right)^\beta = \Psi_2$ and $\Psi_2 = \prod_{x_i \in \mathbf{D}\cap\mathbf{R}} \mathsf{F}_s(x_i)^\beta = \Psi'_2$. Combining the above equations, we have $\left(\frac{\Psi'_1}{\Psi_1}\right)^{(X-X')^{-1}} = \theta$. As a result, we find the value of v: $v = \frac{R}{\theta}$.

Combining the results in Part I, II, III and IV, we conclude that no efficient adversary against the preliminary scheme can output a wrong result and forged a valid proof. In other words, the preliminary scheme is sound. □

8 Performance and Extension

8.1 Performance

In the setup phase, the computation complexity on Alice's side is $O(dN \log \mathcal{Z})$ and the dominant step is Step 3 of DEnc in Figure 2. In the query phase, the communication overhead (in term of bits) per query is $O(d^2 \log^2 \mathcal{Z})$: (1) In $\mathsf{CollRes}$, the communication overhead is dominated by the size of challenge-message $\boldsymbol{\delta}$, which is in $O(d \log^2 \mathcal{Z})$, i.e. $O(\log \mathcal{Z})$ decryption keys for each dimension, and each decryption key of size $O(\log \mathcal{Z})$; (2) There are $O(d)$ invocations of $\mathsf{CollRes}$ to process one query. Computation complexity on Bob's side is $O(dN \log \mathcal{Z})$ (bilinear map): (1) In $\mathsf{CollRes}$, $O(d|\mathbf{D} \cap \mathbf{R}| \log \mathcal{Z})$ computation is required for query range $\mathbf{R}$ and the dominant computation step is Step B1 of $\mathsf{CollRes}$ in Figure 2; (2) In total, $\sum_{\ell=0}^{2d} O(d|\mathbf{D} \cap \mathbf{R}_\ell| \log \mathcal{Z}) = O(dN \log \mathcal{Z})$, where $\{\mathbf{R}_\ell : \ell \in [0, 2d]\}$ is a partition of the domain $[\mathcal{Z}]^d$. The computation complexity per query on Alice's side is $O(d^2 \log^2 \mathcal{Z})$ (group multiplications). The dominant computation step is Step A1 of $\mathsf{CollRes}$ in Figure 2. The storage overhead on Bob's side, is $O(dN)$. The storage cost on Alice's side, i.e. the size of key and $\mathbf{D}_{\mathtt{A}}$, is $O(d + \ell)$, which can be reduced to $O(1)$ (precisely $O(1)$ number of seeds and each seed with length equal to the security parameter κ) using a pseudorandom function.

8.2 Extension: Support Range Selection

With the help of an aggregate signature scheme (e.g. BLS signature [54, 55]), it is straightforward to extend our scheme to authenticate range selection query with range $\mathbf{R}$, which asks for the set $\{\boldsymbol{x} : \boldsymbol{x} \in \mathbf{D} \cap \mathbf{R}\}$, with $O(d^2 \log^2 \mathcal{Z})$ communication overhead. To the best of our knowledge, this is the first solution that authenticates multidimensional range selection query with sublinear communication overhead in the worst case and polynomial storage on Bob's side.

We assume the dataset $\mathbf{D}$ is a set of *distinct* points. The authentication scheme for range selection query is as follows:

Authenticating Multidimensional Range Selection Query

1. In the setup, Alice generates a signature $\mathsf{Sig}(\boldsymbol{x})$ for each data point $\boldsymbol{x} \in \mathbf{D}$ using an aggregate signature scheme, and sends all signatures to Bob.
2. To answer a range selection query with range $\mathbf{R}$, Bob finds the set $S = \{\boldsymbol{x} : \boldsymbol{x} \in \mathbf{D} \cap \mathbf{R}\}$ and computes an aggregated signature $\mathsf{Sig}(S)$ for set S from signatures $\mathsf{Sig}(\boldsymbol{x})$'s for point $\boldsymbol{x} \in \mathbf{D} \cap \mathbf{R}$, using the aggregate signature scheme. Bob sends $(S, \mathsf{Sig}(S))$ to Alice.
3. Alice verifies: (1) Is S a set of distinct points? (2) Is S a subset of query range $\mathbf{R}$? (3) Is $\mathsf{Sig}(S)$ a valid signature for S?
4. Alice issues a count query with range $\mathbf{R}$ to Bob using our scheme presented in Section 6 and gets authenticated result N_0.
5. Alice verifies whether $|S| = N_0$.
6. Alice accepts S as the query result, if all verifications succeed.

We remark that the extra aggregate signature for each data point is actually unnecessary, since our authentication tag can take the role of aggregate signature in this particular application. The details are in our technique report.

9 Conclusion

We proposed a scheme to authenticate aggregate range query over static multidimensional outsourced dataset, and the communication complexity (in term of bits) is $O(d^2 \log^2 \mathcal{Z})$ (d is the dimension and each data point is in domain $[\mathcal{Z}]^d$). Aggregate operations that our scheme can (potentially) support include counting, summing, and finding of the minimum or maximum or median. Our authentication scheme and techniques can be useful in other applications, if suitable functional encryption scheme can be constructed. The proposed functional encryption scheme and the idea of implementing functional encryption by exploiting the polymorphic property of existing encryption schemes may have independent interests.

References

1. Wu, J., Stinson, D.: An Efficient Identification Protocol and the Knowledge-of-Exponent Assumption. Cryptology ePrint Archive, Report 2007/479 (2007) `http://eprint.iacr.org/`.
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. Devanbu, P.T., Gertz, M., Martel, C.U., Stubblebine, S.G.: Authentic Third-party Data Publication. In: Proceedings of the IFIP TC11/ WG11.3 Fourteenth Annual Working Conference on Database Security. (2001) 101–112
4. Hacigümüş, H., Iyer, B., Li, C., Mehrotra, S.: Executing SQL over encrypted data in the database-service-provider model. In: SIGMOD '02: ACM SIGMOD International conference on Management of data. (2002) 216–227
5. Mykletun, E., Narasimha, M., Tsudik, G.: Authentication and Integrity in Outsourced Databases. Trans. Storage **2**(2) (2006) 107–138
6. Gennaro, R., Gentry, C., Parno, B.: Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In: CRYPTO '10: Annual International Cryptology Conference on Advances in Cryptology. (2010) 465–482

7. Chung, K.M., Kalai, Y., Vadhan, S.P.: Improved Delegation of Computation Using Fully Homomorphic Encryption. In: CRYPTO '10: Annual International Cryptology Conference on Advances in Cryptology. (2010) 483–501
8. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC '09: ACM symposium on Theory of computing. (2009) 169–178
9. Martel, C., Nuckolls, G., Devanbu, P., Gertz, M., Kwong, A., Stubblebine, S.G.: A General Model for Authenticated Data Structures. Algorithmica **39**(1) (2004) 21–41
10. Pang, H., Jain, A., Ramamritham, K., Tan, K.L.: Verifying completeness of relational query results in data publishing. In: SIGMOD '05: ACM SIGMOD International conference on Management of data. (2005) 407–418
11. Cheng, W., Tan, K.L.: Query assurance verification for outsourced multi-dimensional databases. J. Comput. Secur. **17**(1) (2009) 101–126
12. Li, F., Hadjieleftheriou, M., Kollios, G., Reyzin, L.: Dynamic authenticated index structures for outsourced databases. In: SIGMOD '06: ACM SIGMOD International conference on Management of data. (2006) 121–132
13. Atallah, M.J., Cho, Y., Kundu, A.: Efficient Data Authentication in an Environment of Untrusted Third-Party Distributors. In: ICDE '08: IEEE International Conference on Data Engineering. (2008) 696–704
14. Mouratidis, K., Sacharidis, D., Pang, H.: Partially materialized digest scheme: an efficient verification method for outsourced databases. The VLDB Journal **18**(1) (2009) 363–381
15. Pang, H., Zhang, J., Mouratidis, K.: Scalable Verification for Outsourced Dynamic Databases. Proc. VLDB Endow. **2** (2009) 802–813
16. Goodrich, M.T., Tamassia, R., Triandopoulos, N.: Super-Efficient Verification of Dynamic Outsourced Databases. In: CT-RSA '08: The Cryptographer's Track at the RSA Conference on Topics in Cryptology. (2008) 407–424
17. Pang, H., Tan, K.L.: Verifying Completeness of Relational Query Answers from Online Servers. ACM Trans. Inf. Syst. Secur. **11**(2) (2008) 1–50
18. Thompson, B., Yao, D., Haber, S., Horne, W.G., Sander, T.: Privacy-Preserving Computation and Verification of Aggregate Queries on Outsourced Databases. In: PETS '09: Privacy Enhancing Technologies Symposium. (2009)
19. Li, F., Hadjieleftheriou, M., Kollios, G., Reyzin, L.: Authenticated index structures for aggregation queries. ACM Trans. Inf. Syst. Secur. **13** (2010) 32:1–32:35
20. Xu, J.: Authenticating aggregate range queries over dynamic multidimensional dataset. Cryptology ePrint Archive, Report 2010/244 (2010) `http://eprint.iacr.org/`.
21. Chen, H., Ma, X., Hsu, W.W., Li, N., Wang, Q.: Access Control Friendly Query Verification for Outsourced Data Publishing. In: ESORICS '08: European Symposium on Research in Computer Security. (2008) 177–191
22. 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`.
23. O'Neill, A.: Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556 (2010) `http://eprint.iacr.org/`.
24. Preparata, F.P., Shamos, M.I.: Computational geometry: an introduction. Springer-Verlag New York, Inc. (1985)
25. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully Homomorphic Encryption over the Integers. In: EUROCRYPT '10: Annual International Conference on Advances in Cryptology. (2010) 24–43
26. Hacigümüs, H., Iyer, B.R., Mehrotra, S.: Efficient Execution of Aggregation Queries over Encrypted Relational Databases. In: DASFAA. (2004) 125–136
27. Mykletun, E., Tsudik, G.: Aggregation Queries in the Database-As-a-Service Model. In: IFIP WG 11.3 Working Conference on Data and Applications Security. (2006) 89–103
28. Ge, T., Zdonik, S.B.: Answering Aggregation Queries in a Secure System Model. In: VLDB '07: International Conference on Very Large Data Bases. (2007) 519–530
29. Devanbu, P., Gertz, M., Martel, C., Stubblebine, S.G.: Authentic data publication over the internet. J. Comput. Secur. **11**(3) (2003) 291–314
30. Sion, R.: Query Execution Assurance for Outsourced Databases. In: VLDB '05: International Conference on Very Large Data Bases. (2005) 601–612
31. Xie, M., Wang, H., Yin, J., Meng, X.: Integrity auditing of outsourced data. In: VLDB '07: International conference on Very large data bases. (2007) 782–793
32. Yang, Y., Papadias, D., Papadopoulos, S., Kalnis, P.: Authenticated join processing in outsourced databases. In: SIGMOD '09: ACM SIGMOD International conference on Management of data. (2009) 5–18
33. Haber, S., Horne, W., Sander, T., Yao, D.: Privacy-Preserving Verification of Aggregate Queries on Outsourced Databases. Technical report, HP Laboratories (2006) HPL-2006-128.
34. Gentry, C.: Toward Basing Fully Homomorphic Encryption on Worst-Case Hardness. In: CRYPTO '10: Annual International Cryptology Conference on Advances in Cryptology. (2010) 116–137
35. 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
36. 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
37. 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

38. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D.: Provable data possession at untrusted stores. In: CCS '07: ACM conference on Computer and communications security. (2007) 598–609
39. Chang, E.C., Xu, J.: Remote Integrity Check with Dishonest Storage Server. In: ESORICS '08: European Symposium on Research in Computer Security. (2008) 223–237
40. Shacham, H., Waters, B.: Compact Proofs of Retrievability. In: ASIACRYPT '08: International Conference on the Theory and Application of Cryptology and Information Security. (2008) 90–107
41. Bowers, K.D., Juels, A., Oprea, A.: HAIL: A High-Availability and Integrity Layer for Cloud Storage. Cryptology ePrint Archive, Report 2008/489 (2008) `http://eprint.iacr.org/`.
42. Dodis, Y., Vadhan, S., Wichs, D.: Proofs of Retrievability via Hardness Amplification. In: TCC '09: Theory of Cryptography Conference. (2009) 109–127
43. Ateniese, G., Kamara, S., Katz, J.: Proofs of Storage from Homomorphic Identification Protocols. In: ASIACRYPT '09: International Conference on the Theory and Application of Cryptology and Information Security. (2009) 319–333
44. Damgård, I.: Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks. In: CRYPTO '91: Annual International Cryptology Conference on Advances in Cryptology. (1992) 445–456
45. Hada, S., Tanaka, T.: On the Existence of 3-Round Zero-Knowledge Protocols. In: CRYPTO '98: Annual International Cryptology Conference on Advances in Cryptology. (1998) 408–423
46. Bellare, M., Palacio, A.: Towards Plaintext-Aware Public-Key Encryption Without Random Oracles. In: ASIACRYPT '04: International Conference on the Theory and Application of Cryptology and Information Security. (2004) 48–62
47. Krawczyk, H.: HMQV: A High-Performance Secure Diffie-Hellman Protocol. In: CRYPTO '05: Annual International Cryptology Conference on Advances in Cryptology. (2005) 546–566
48. Dent, A.W.: The Cramer-Shoup Encryption Scheme Is Plaintext Aware in the Standard Model. In: EUROCRYPT '06: Annual International Conference on Advances in Cryptology. (2006) 289–307
49. Bellare, M., Palacio, A.: The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols. In: CRYPTO '04: Annual International Cryptology Conference on Advances in Cryptology. (2004) 273–289
50. Groth, J.: Short Pairing-Based Non-interactive Zero-Knowledge Arguments. In: ASIACRYPT '10: International Conference on the Theory and Application of Cryptology and Information Security. (2010) 321–340
51. Abe, M., Fehr, S.: Perfect NIZK with adaptive soundness. In: TCC '07: Theory of Cryptography Conference. (2007)
52. Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: EUROCRYPT '05: Annual International Conference on Advances in Cryptology. (2005) 457–473
53. 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
54. Boneh, D., Lynn, B., Shacham, H.: Short Signatures from the Weil Pairing. J. Cryptol. **17**(4) (2004) 297–319
55. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In: EUROCRYPT '03: Annual International Conference on Advances in Cryptology. (2003) 416–432

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 κ 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

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

B Two Propositions

Some analysis in our proof is based on the following propositions (*We do not claim the discovery of Proposition 1 or Proposition 2.*)

Proposition 1 *If event A implies event B, then* $\mathsf{Pr}[A] \leq \mathsf{Pr}[B]$.

Proof. Since $A \Rightarrow B$, we have $\mathsf{Pr}[\neg A \vee B] = 1$ and $\mathsf{Pr}[A \wedge \neg B] = 0$. Therefore,

$$\mathsf{Pr}[A] = \mathsf{Pr}[A \wedge \neg B] + \mathsf{Pr}[A \wedge B] = 0 + \mathsf{Pr}[A \wedge B] = \mathsf{Pr}[A|B]\mathsf{Pr}[B] \leq \mathsf{Pr}[B].$$

□

Proposition 2 *For any n events $A_1, \ldots, A_n$, it always holds that* $\mathsf{Pr}[\bigwedge_{1\leq i\leq n} A_i] \geq 1 - \sum_{i=1}^n \mathsf{Pr}[\neg A_i]$.

Proof.

$$\mathsf{Pr}[\bigwedge_{1\leq i\leq n} A_i] = 1 - \mathsf{Pr}[\bigvee_{1\leq i\leq n} \neg A_i] \geq 1 - \sum_{i=1}^n \mathsf{Pr}[\neg A_i].$$

□

C Proof of Lemma 1

Lemma 1 *The functional encryption scheme FE described in Figure 1 satisfies these properties:*

(a) *For any $(pk, sk) \leftarrow f\mathsf{Setup}(1^\kappa, d, \mathcal{Z})$, for any message $\mathsf{Msg} \in \mathbb{Z}_p^*$, for any point $\boldsymbol{x} \in [\mathcal{Z}]^d$, for any rectangular range $\mathbf{R} \subseteq [\mathcal{Z}]^d$, if $\mathsf{CT} \leftarrow f\mathsf{Enc}(\mathsf{Msg}, \boldsymbol{x}, sk)$ and $\boldsymbol{\delta} \leftarrow f\mathsf{KeyGen}(\mathbf{R}, \rho, sk)$, then*

$$f\mathsf{Dec}(\mathsf{CT},\ \boldsymbol{x},\ \mathbf{R},\ \boldsymbol{\delta},\ pk) = \begin{cases} f_\rho(\mathsf{Msg}) & (\textit{if } \boldsymbol{x} \in \mathbf{R}) \\ \perp & (\textit{otherwise}) \end{cases} \tag{14}$$

(b) *For any* $(pk, sk) \leftarrow f\mathsf{Setup}(1^\kappa, d, \mathcal{Z})$*, 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* $y \in \widetilde{\mathbb{G}}$*, if* $\mathsf{CT} \leftarrow f\mathsf{Enc}(\mathsf{Msg}, \boldsymbol{x}, sk)$ *and* $\boldsymbol{\delta} \leftarrow f\mathsf{KeyGen}(\mathbf{R}, \rho, sk)$*, then*

$$f\mathsf{Dec}(\mathsf{Mult}(\mathsf{CT}, y, pk),\ \boldsymbol{x},\ \mathbf{R},\ \boldsymbol{\delta},\ pk) = \begin{cases} y \cdot f_\rho(\mathsf{Msg}) & (\mathit{if}\ \boldsymbol{x} \in \mathbf{R}) \\ \perp & (\mathit{otherwise}) \end{cases} \tag{15}$$

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 $\mathsf{CT} = \widehat{\mathsf{CT}}$:

$$\begin{aligned} &\mathsf{CT} = \mathsf{Encrypt}(\mathsf{params}, \mathsf{id}, M; s) = \left(\Omega^s \cdot M,\ g^s,\ \left(h_1^{I_1} \cdots h_k^{I_k} \cdot g_3\right)^s\ \right) \\ &\qquad \text{under key: } \mathtt{params} = (g, g_1, g_2, g_3, h_1, \ldots, h_\ell, \Omega = e(g_1, g_2)),\ \ \mathtt{master\text{-}key} = g_2^\alpha \\ &\widehat{\mathsf{CT}} = \mathsf{Encrypt}(\widehat{\mathsf{params}}, \mathsf{id}, \widehat{M}; sz) = \left(\Omega^{sz} \cdot \widehat{M},\ \widehat{g}^{sz},\ \left(\widehat{h}_1^{I_1} \cdots \widehat{h}_k^{I_k} \cdot \widehat{g}_3\right)^{sz}\ \right), \\ &\qquad \text{under key: } \widehat{\mathtt{params}} = (\widehat{g}, g_1, g_2, \widehat{g}_3, \widehat{h}_1, \ldots, \widehat{h}_\ell, \Omega = e(g_1, g_2)),\ \ \widehat{\mathtt{master\text{-}key}} = g_2^{\alpha z} \end{aligned} \tag{16}$$

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.

Proof of Lemma 1(a): Let $(pk, sk) \leftarrow f\mathsf{Setup}(1^\kappa)$, 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 (16)): 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 $f\mathsf{Dec}$ (Figure 1). 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

$$\widetilde{t}_j = \widehat{M} = \sigma_j \Omega^{s_j(1-\tau_j\rho)}, j \in [d]. \tag{17}$$

Combining all d dimensions, and applying the two equalities (see algorithm $f\mathsf{Enc}$ in Figure 1) $\mathsf{Msg} = -\sum_{j=1}^d s_j\tau_j \bmod p$ 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 Figure 1).

Proof of Lemma 1(b):

In case $\boldsymbol{x} \in \mathbf{R}$. Check the decryption algorithm $\mathsf{Decrypt}$ of BBG HIBE (See Appendix A), it is easy to verify that: For any $\eta \in \widetilde{\mathbb{G}}$, if $\mathsf{Decrypt}(d_{\mathsf{id}}, (A, B, C))$ outputs M, then $\mathsf{Decrypt}(d_{\mathsf{id}}, (A \cdot \eta, B, C))$ will output $\eta \cdot M$.

Let $\eta_1, \ldots, \eta_d$ be as in Step 2 of Mult in Figure 1. Similar as the argument for Lemma 1(a), for dimension $j \in [d]$, we have ($\widetilde{t}_j$ is as in equation (17) and $\widetilde{t}'_j$ is the counterpart of $\widetilde{t}_j$ for decrypting ciphertext $\mathsf{Mult}(\mathsf{CT}, y, pk)$)

$$\widetilde{t}'_j = \eta_j \widehat{M} = \eta_j \widetilde{t}_j.$$

Combining all d dimensions and applying the equation $\prod_{j=1}^{d} \eta_j = y$ (See Step 2 of Mult) and the result in Lemma 1(a), we have

$$f\mathsf{Dec}(\mathsf{Mult}(\mathsf{CT}, y, pk),\ \boldsymbol{x},\ \mathbf{R},\ \boldsymbol{\delta},\ pk) = \prod_{j=1}^{d} \widetilde{t}'_j = \prod_{j=1}^{d} \left(\eta_j \widetilde{t}_j\right) = \left(\prod_{j=1}^{d} \eta_j\right) \cdot f\mathsf{Dec}(\mathsf{CT},\ \boldsymbol{x},\ \mathbf{R},\ \boldsymbol{\delta},\ pk) = y \cdot f_\rho(\mathsf{Msg}).$$

In case $\boldsymbol{x} \notin \mathbf{R}$. The same argument for the case $\boldsymbol{x} \notin \mathbf{R}$ of Lemma 1(a) applies. □

D Proof of Theorem 2

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 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.

Proof.

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 b. Hence, in case of $b = 1$, $\mathcal{A}_{\mathsf{FE}}$ wins 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, Mult). 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 $\mathsf{id}^* = \mathsf{ID}_\xi(x_\xi) \in \left(\mathbb{Z}_p^*\right)^\ell$ 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

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

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

$$\boldsymbol{c}_j \leftarrow \mathsf{Encrypt}(\mathtt{params},\ \mathsf{ID}_j(x_j),\ \sigma_j;\ s_j). \tag{18}$$

5. Define $\boldsymbol{c}_\xi$ based on the BBG challenge ciphertext $\mathsf{CT} = (\Omega^s m_b, B, C)$:

$$\boldsymbol{c}_\xi = (\Omega^s m_b \cdot m_0^{-1} \cdot \sigma_\xi,\ B,\ C).$$

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 $\Omega^{s_\xi} = \Omega^s m_b \cdot m_0^{-1}$. Although the value Msg_a is unknown since s_ξ is unknown, $\mathcal{A}_{\mathsf{BBG}}$ can still compute $f_1(\mathsf{Msg}_a)$:

$$f_1(\mathsf{Msg}_a) = \Omega^{\mathsf{Msg}_a} = \left((\Omega^s m_b)^{-1} \cdot m_0\right)^{\tau_\xi} \cdot \Omega^{-\sum_{\substack{1\le j\le d \\ j\neq\xi}} s_j\cdot\tau_j}$$

$\mathcal{A}_{\mathsf{BBG}}$ computes $f_1(\mathsf{Msg}_{1-a}) = \Omega^{\mathsf{Msg}_{1-a}}$.

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 5.2: (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 $(K_0, K_1, \Upsilon_k, \ldots, \Upsilon_\ell)$ and set $d'_{\mathsf{id}} = (K_0^{\rho\tau_j}, K_1^{\rho\tau_j}, \Upsilon_k^{\rho\tau_j}, \ldots, \Upsilon_\ell^{\rho\tau_j})$.
      – 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 $\boldsymbol{c}_\xi$ 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 $m_0, m_1, \mathsf{params}$.

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 $b' = 1 \Leftrightarrow a \neq a'$. Applying the Proposition 1, 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 $\mathcal{A}_{\mathsf{FE}}$. □

E A valid proof should be generated from points within dataset D

The notion that a valid proof is essentially generated from points (and their tags) within the dataset $\mathbf{D}$, is formulized by Lemma 6. We prove Lemma 6 in two steps: first we show that the **GKEA** Assumption 2 implies Lemma 5 (which states that an alternative form of **GKEA** problem is hard); then we derive Lemma 6 from Lemma 5.

We remark that both the proof of Lemma 5 in Appendix E.1 and proof of Lemma 6 in Appendix E.2 follow the proof framework for statement that **KEA3** implies **KEA** in [49]. Their proof can be outlined as follows: Given any adversary algorithm $\mathcal{A}_{\mathbf{KEA}}$, construct an adversary algorithm $\mathcal{A}_{\mathbf{KEA3}}$. Then applying **KEA3** Assumption, there exists an extractor algorithm $\bar{\mathcal{A}}_{\mathbf{KEA3}}$. Based on $\bar{\mathcal{A}}_{\mathbf{KEA3}}$, construct extractor algorithm $\bar{\mathcal{A}}_{\mathbf{KEA}}$ for $\mathcal{A}_{\mathbf{KEA}}$. The key point is how to convert the input/output between $\bar{\mathcal{A}}_{\mathbf{KEA}}$ ($\mathcal{A}_{\mathbf{KEA}}$, respectively) and $\bar{\mathcal{A}}_{\mathbf{KEA3}}$ ($\mathcal{A}_{\mathbf{KEA3}}$, respectively).

E.1 Lemma 5 and Proof

Lemma 5 *Suppose Assumption 2 holds. For any PPT algorithm $\mathcal{A}$, there exists a PPT algorithm $\bar{\mathcal{A}}$, such that*

$$\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\mathbf{Lem\ 5}}(\kappa) \stackrel{\text{def}}{=} \mathsf{Pr}\left[\begin{array}{l} S_{m+1} \leftarrow \{(\theta v_i, v_i^\beta) : i \in [m+1], v_i \xleftarrow{\$} \widetilde{\mathbb{G}}, \theta \xleftarrow{\$} \widetilde{\mathbb{G}}, \beta \xleftarrow{\$} \mathbb{Z}_p^*\} \\ (\Psi_1, \Psi_2, X) \leftarrow \mathcal{A}(S_{m+1}; r); \\ (\Psi_1, \Psi_2, X, \mu_1, \mu_2, \ldots, \mu_m, \mu_{m+1}) \leftarrow \bar{\mathcal{A}}(S_{m+1}; r, \bar{r}) : \\ \quad \Psi_2 = \left(\frac{\Psi_1}{\theta^X}\right)^\beta \ \wedge \ \Psi_2 \neq \prod_{j=1}^{m+1} (v_j^\beta)^{\mu_j} \end{array}\right] \leq \nu_2(\kappa), \qquad (19)$$

where the probability is taken over all random coins used, m is polynomial in κ and function $\nu_2(\cdot)$ is as in Assumption 2.

Proof (of Lemma 5). Let adversary $\mathcal{A}$ be as in Lemma 5. We construct a **GKEA** adversary $\mathcal{A}_1$ based on an adversary $\mathcal{A}$.

Construction of **GKEA** adversary $\mathcal{A}_1$: Based on an adversary $\mathcal{A}$

1. The input is $W_m = \{(u_i, u_i^\beta) \in \widetilde{\mathbb{G}}^2 : 1 \le i \le m\}$ and the random coin is r_1.
2. Choose two independent random elements $R_1, R_2 \in \widetilde{\mathbb{G}}^2$ based on the random coin r_1. There exists some unknown $\theta, v_{m+1} \in \widetilde{\mathbb{G}}$, such that $R_1 = \theta v_{m+1}$ and $R_2 = v_{m+1}^\beta$.
3. For each $1 \le i \le m$, define $v_i = u_i v_{m+1}$ and compute $\theta v_i = u_i(\theta v_{m+1}) = u_i R_1$ and $v_i^\beta = (u_i v_{m+1})^\beta = u_i^\beta R_2$. Let $S_{m+1} = \{(\theta v_i, v_i^\beta) : 1 \le i \le m+1\}$.
4. Invoke the adversary $\mathcal{A}$ with random coin r derived from r_1: $(\Psi_1, \Psi_2, X) \leftarrow \mathcal{A}(S_{m+1}; r)$.
5. Output $(U_1 = \frac{\Psi_1}{R_1^X}, U_2 = \frac{\Psi_2}{R_2^X})$.

Since $\left(\frac{\Psi_1}{R_1^X}\right)^\beta = \frac{\Psi_1^\beta}{\theta^{X\beta} v_{m+1}^{X\beta}}$ and $\frac{\Psi_2}{R_2^X} = \frac{\Psi_2}{v_{m+1}^{X\beta}}$, we have

$$\left(\frac{\Psi_1}{\theta^X}\right)^\beta = \Psi_2 \qquad \Leftrightarrow \qquad \left(\frac{\Psi_1}{R_1^X}\right)^\beta = \frac{\Psi_2}{R_2^X}. \tag{20}$$

According to Assumption 2, there exists an extractor $\bar{\mathcal{A}}_1$ for the adversary $\mathcal{A}_1$, such that $\mathsf{Adv}^{\mathbf{GKEA}}_{\mathcal{A}_1, \bar{\mathcal{A}}_1}$ is negligible. Now we construct an extractor $\bar{\mathcal{A}}$ for $\mathcal{A}$ based on $\bar{\mathcal{A}}_1$.

Construction of extractor $\bar{\mathcal{A}}$ for $\mathcal{A}$: Based on **GKEA** extractor $\bar{\mathcal{A}}_1$

1. The input is $S_{m+1} = \{(\theta v_i, v_i^\beta) : 1 \le i \le m+1\}$. The random coin is $(r, \bar{r})$.
2. For each $1 \le i \le m$, set $u_i = \frac{\theta v_i}{\theta v_{m+1}}$ and compute $u_i^\beta = \frac{v_i^\beta}{v_{m+1}^\beta}$. Let $W_m = \{(u_i, u_i^\beta) : 1 \le i \le m\}$.
3. Invoke adversary $\mathcal{A}_1$ with random coin $r_1 = (\theta v_{m+1}, v_{m+1}^\beta, r)$: $(U_1 = \frac{\Psi_1}{R_1^X}, U_2 = \frac{\Psi_2}{R_2^X}) \leftarrow \mathcal{A}_1(W_m; r_1)$.
   Note: We can represent the random coin r_1 used by $\mathcal{A}_1$ as (R_1, R_2, r).
4. Invoke extractor $\bar{\mathcal{A}}_1$ with random coin $(r_1, \bar{r}_1 = \bar{r})$: $(\mu_1, \ldots, \mu_m) \leftarrow \bar{\mathcal{A}}_1(W_m; r_1, \bar{r}_1)$.
5. Define $\mu_{m+1} = X - \sum_{i=1}^m \mu_i$. Output $(\mu_1, \ldots, \mu_m, \mu_{m+1})$.

If $U_2 = \prod_{i=1}^m u_i^{\beta\mu_i}$, then we have

$$\prod_{i=1}^{m+1} v_i^{\beta\mu_i} = v_{m+1}^{\beta\mu_{m+1}} \prod_{i=1}^{m} v_i^{\beta\mu_i} = v_{m+1}^{\beta(X - \sum_{i=1}^m \mu_i)} \prod_{i=1}^{m} u_i^{\beta\mu_i} \prod_{i=1}^{m} v_{m+1}^{\beta\mu_i} = v_{m+1}^{\beta X} U_2 = R_2^X U_2 = \Psi_2.$$

That is,

$$U_2 = \prod_{i=1}^m u_i^{\beta\mu_i} \qquad \Rightarrow \qquad \prod_{i=1}^{m+1} v_i^{\beta\mu_i} = \Psi_2. \tag{21}$$

If $U_1^\beta = U_2 \;\Rightarrow\; U_2 = \prod_{i=1}^m u_i^{\beta\mu_i}$, combining with equation (20) and equation (21), we have

$$\left(\frac{\Psi_1}{\theta^X}\right)^\beta = \Psi_2 \Rightarrow U_1^\beta = U_2 \Rightarrow U_2 = \prod_{i=1}^m u_i^{\beta\mu_i} \Rightarrow \Psi_2 = \prod_{i=1}^{m+1} v_i^{\beta\mu_i}.$$

As a result,

$$\left(U_1^\beta = U_2 \Rightarrow U_2 = \prod_{i=1}^m u_i^{\beta\mu_i}\right) \Rightarrow \left(\left(\frac{\Psi_1}{\theta^X}\right)^\beta = \Psi_2 \Rightarrow \Psi_2 = \prod_{i=1}^{m+1} v_i^{\beta\mu_i}\right)$$

Note that the implications in equation (20) and equation (21) are *always* true, while the implication that $U_1^\beta = U_2 \Rightarrow U_2 = \prod_{i=1}^m u_i^{\beta\mu_i}$ is true only with certain probability.

Applying the Proposition 1 in Appendix B, we have

$$\mathsf{Pr}\left[U_1^\beta = U_2 \Rightarrow U_2 = \prod_{i=1}^m u_i^{\beta\mu_i}\right] = 1 - \mathsf{Adv}^{\mathbf{GKEA}}_{\mathcal{A}_1, \bar{\mathcal{A}}_1} \le \mathsf{Pr}\left[\left(\frac{\Psi_1}{\theta^X}\right)^\beta = \Psi_2 \Rightarrow \Psi_2 = \prod_{i=1}^{m+1} v_i^{\beta\mu_i}\right] = 1 - \mathsf{Adv}^{\mathbf{Lem\ 5}}_{\mathcal{A}, \bar{\mathcal{A}}}.$$

Hence,

$$\mathsf{Adv}^{\mathbf{Lem\ 5}}_{\mathcal{A}, \bar{\mathcal{A}}} \le \mathsf{Adv}^{\mathbf{GKEA}}_{\mathcal{A}_1, \bar{\mathcal{A}}_1} \le \nu_2.$$

□

E.2 Lemma 6 and Proof

Lemma 6 *Suppose Assumption 2 holds. For any PPT algorithm $\mathcal{A}$, there exists a PPT algorithm $\bar{\mathcal{A}}$, such that* $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem 6}}(1^\kappa) \leq \nu_2(\kappa)$*, where the advantage* $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem 6}}$ *of $\mathcal{A}$ against $\bar{\mathcal{A}}$ w.r.t. scheme $\widetilde{\mathcal{E}} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{CollRes})$ is defined as*

$$\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem 6}}(1^\kappa) \overset{\text{def}}{=} 1 - \Pr\left[\begin{array}{l}(\zeta, X, \Psi_2, \mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}, \mathbf{D}, \mathbf{R}) \leftarrow \mathsf{Exp}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}(1^\kappa);\\ \{\mu_i : i \in [N]\} \leftarrow \bar{\mathcal{A}}(\mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}) :\\ \qquad\qquad \zeta = \mathtt{accept} \ \Rightarrow \ \Psi_2 = \prod_{i\in[N]} \left(v_i^{\beta}\right)^{\mu_i}\end{array}\right],$$

where v_i^{β} is the second component of tag $\boldsymbol{t}_i$ for data point $\boldsymbol{x}_i \in \mathbf{D}$ (See Step 2 of DEnc *in Figure 2).*

Proof (of Lemma 6). Let $\mathcal{A}$ be any PPT adversary against scheme $\widetilde{\mathcal{E}} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{CollRes})$. We construct a PPT adversary $\mathcal{B}$ against Lemma 5 based on $\mathcal{A}$.

Adversary $\mathcal{B}$ against Lemma 5: Based on $\mathcal{A}$

1. The input is $S_m = \{(\theta v_j, v_j^{\beta}) \in \widetilde{\mathbb{G}}^2 : j \in [m]\}$. The random coin used in this algorithm is r.
2. Simulate Alice in the experiment $\mathsf{Exp}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$.
   (a) Invoke adversary $\mathcal{A}$ with a random coin derived from r. $\mathcal{A}$ chooses a set $\mathbf{D} = \{\boldsymbol{x}_i \in [\mathcal{Z}]^d : i \in [N]\}$ of $N = m$ d-dimensional data points. *Note: If $N > m$, $\mathcal{B}$ can generate more tuples $(\theta v_j, v_j^{\beta})$ for $j = m+1, \ldots, N$ from S_m. For simplicity, we just assume $N = m$.*
   (b) Simulate KGen:
      i. Invoke $f\mathsf{Setup}(1^\kappa)$ to obtain public/private key pair (pk, sk).
      ii. Choose γ' at random from $\mathbb{Z}_p^*$. $\mathcal{B}$ does not know the values of θ, β.
   (c) Simulate DEnc:
      i. Choose N random elements $W_1, \ldots, W_N$ from $\mathbb{Z}_p^*$.
      ii. For each $i \in [N]$, compute a tag $\boldsymbol{t}_i = \left(\theta v_i, v_i^{\beta}, f_1(W_i)\right)$.
      iii. For each $i \in [N]$, encrypt message W_i under point $\boldsymbol{x}_i$: $\mathsf{CT}'_i \leftarrow f\mathsf{Enc}(W_i, \boldsymbol{x}_i, sk)$; attach $v_i^{\beta\gamma'}$ to ciphertext by applying the homomorphic property of the functional encryption scheme: $\mathsf{CT}_i \leftarrow \mathsf{Mult}(\mathsf{CT}'_i, v_i^{\beta\gamma'}, pk)$.
      iv. Send $\mathbf{D}_{\mathsf{B}} = \{\mathbf{D}, \mathbf{T} = \{\boldsymbol{t}_i : i \in [N]\}, \mathbf{C} = \{\mathsf{CT}_i : i \in [N]\}, pk\}$ to adversary $\mathcal{A}$.
   (d) Simulate CollRes: For each query range $\mathbf{R}_j$ chosen by $\mathcal{A}$ during $\mathcal{A}$'s learning phase and challenging phase
      i. (Step A1) Choose random nonce $\rho \in \mathbb{Z}_p^*$, generate delegation key $\boldsymbol{\delta} \leftarrow f\mathsf{KeyGen}(\mathbf{R}_j, \rho, sk)$, and send $(\mathbf{R}_j, \boldsymbol{\delta})$ to adversary $\mathcal{A}$.
      ii. (Step A2) Do not perform verifications, after receiving reply from $\mathcal{A}$.
3. Receive output $(X, \Psi_1, \Psi_2, \Psi_3, \Psi_4)$ for the challenging query range $\mathbf{R}$ from $\mathcal{A}$. Output (X, Ψ_1, Ψ_2).

Remarks on Algorithm $\mathcal{B}$.

1. Adversary $\mathcal{B}$ has the private key sk, and can generate the challenge-message in the same way as in CollRes in Figure 2.
2. Adversary $\mathcal{B}$ has no knowledge of β or θ, and consequently cannot perform verification as in CollRes.
3. The view of adversary $\mathcal{A}$ after interacting with the simulated scheme is identically distributed with the the view $\mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$ of $\mathcal{A}$ after interacting with the real scheme in the experiment $\mathsf{Exp}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$.
   (a) KGen: The simulator $\mathcal{B}$ generates key (pk, sk) in the same way as the real scheme. The unknown secret key $\beta \in \mathbb{Z}_p^*, \theta \in \widetilde{\mathbb{G}}$ and $\gamma = (\beta\gamma')^{-1} \in \mathbb{Z}_p^*$ are independently and uniformly randomly distributed over corresponding domains.
   (b) DEnc: The dataset $\mathbf{D}$ is generated in the same way as in real experiment. The tags $\mathbf{T}$ are identically distributed as in real experiment. The ciphertexts $\mathbf{C}$ are generated in the same way as in real experiment. As a result, $\mathbf{D}_{\mathsf{B}} = \{\mathbf{D}, \mathbf{T}, \mathbf{C}\}$ that $\mathcal{A}$ received is identically distributed as in real experiment.
   (c) CollRes
      i. Step A1: The simulator just follows the procedure in Figure 2 with key sk and random nonce ρ to execute this step.
      ii. Step A2: Since the simulator does not know the values of secret key (β, γ, θ), it cannot perform the verifications. However, according to our scheme, the accept/reject decisions are always kept secret from Bob (or adversary).

4. If $\mathcal{A}$ is a successful adversary, then its output will indeed pass all verifications with non-negligible probability, although the simulator cannot perform the actual verifications and yet cannot know whether $\mathcal{A}$ succeeds or not in each single attack instance.

From the random coin r, $\mathcal{B}$ can simulate the experiment $\mathsf{Exp}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$ and produce a view view_r which is identically distributed as the view $\mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$ produced by a real experiment. In the other direction, information theoretically, the random coin r can be recovered from the adversary's view view_r, considering r as the collection of all (true) random bits flipped in the simulation. Consequently, we can view view_r as an alternative representation of random coin r.

By Lemma 5, there exists a PPT algorithm $\bar{\mathcal{B}}$ such that $\mathsf{Adv}_{\mathcal{B},\bar{\mathcal{B}}}^{\textbf{Lem 5}} \leq \nu_2$. We construct an extractor $\bar{\mathcal{A}}$ for adversary $\mathcal{A}$ based on $\bar{\mathcal{B}}$.

Extractor $\bar{\mathcal{A}}$: Based on $\bar{\mathcal{B}}$

1. The input is $\mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$, a state variable describing all random coins chosen and all message accessed by $\mathcal{A}$ during interactions with $\widetilde{\mathcal{E}}$ in the experiment $\mathsf{Exp}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$.
2. Recover $\mathbf{D}, \mathbf{T}, \mathbf{C}$ from $\mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$ and construct a set $S_N \leftarrow \{(\theta v_i, v_i^{\beta}) : i \in [N]\}$, where θv_i and v_i^{β} are the first two components of tag $\boldsymbol{t}_i \in \mathbf{T}$.
3. Invoke $\mathcal{B}$ on input S_N with random coin $\mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$. $\mathcal{B}$ extracts information from $\mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$ and replays[9] the interaction between Alice and Bob (i.e. the adversary $\mathcal{A}$) in the experiment $\mathsf{Exp}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$. Denote the output of experiment as $(\zeta, X, \boldsymbol{\Psi}, \mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}, \mathbf{D}, \mathbf{R})$. Recover the reply of $\mathcal{A}$ on the challenging query $\mathbf{R}$ from $\mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$, and denote it with $(X, \Psi_1, \Psi_2, \Psi_3, \Psi_4)$. $\mathcal{B}$ outputs (X, Ψ_1, Ψ_2).
4. Let $\bar{\mathcal{B}}$ be the extractor such that $\mathsf{Adv}_{\mathcal{B},\bar{\mathcal{B}}}^{\textbf{Lem 5}} \leq \nu_2$. Invoke $\bar{\mathcal{B}}$ on input S_N using random coin $\mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$, and obtain output $(\mu_1, \ldots, \mu_N)$ from $\bar{\mathcal{B}}$. Output $(\Psi_1, \Psi_2, X, \mu_1, \ldots, \mu_N)$.

Note that according to the algorithm $\mathsf{CollRes}$, $\zeta = \mathtt{accept}$ implies that the verification is passed and $\Psi_2 = \left(\frac{\Psi_1}{\theta^X}\right)^{\beta}$ (the first equality test in equation (11)). We have

$$\zeta = \mathtt{accept} \;\wedge\; \Psi_2 \neq \prod_{i\in[N]} (v_i^{\beta})^{\mu_i} \quad\Rightarrow\quad \Psi_2 = \left(\frac{\Psi_1}{\theta^X}\right)^{\beta} \;\wedge\; \Psi_2 \neq \prod_{i\in[N]} (v_i^{\beta})^{\mu_i}$$

Hence, by applying Proposition 1, we have

$$\begin{aligned}\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem 6}} &= \Pr\left[\zeta = \mathtt{accept} \;\wedge\; \Psi_2 \neq \prod_{i\in[N]} (v_i^{\beta})^{\mu_i}\right] \\ \leq \mathsf{Adv}_{\mathcal{B},\bar{\mathcal{B}}}^{\textbf{Lem 5}} &= \Pr\left[\Psi_2 = \left(\frac{\Psi_1}{\theta^X}\right)^{\beta} \;\wedge\; \Psi_2 \neq \prod_{i\in[N]} (v_i^{\beta})^{\mu_i}\right] \leq \nu_2.\end{aligned}$$

□

F A valid proof should be generated from points within intersection $\mathbf{D} \cap \mathbf{R}$

Theorem 7 *Suppose Assumption 1 and Assumption 2 hold, and* FE *scheme constructed in Figure 1 is* weak-IND-sID-CPA *secure. For any PPT algorithm $\mathcal{A}$, there exists PPT algorithm $\bar{\mathcal{A}}$, such that both* $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem 6}}$ *and* $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 7}}$ *are negligible, where the advantage* $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 7}}$ *of $\mathcal{A}$ against $\bar{\mathcal{A}}$ w.r.t. scheme* $\widetilde{\mathcal{E}} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{CollRes})$ *is defined as*

$$\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 7}}(1^{\kappa}) \overset{\text{def}}{=} 1 - \Pr\left[\begin{array}{l}(\zeta, X, \Psi_2, \mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}, \mathbf{D}, \mathbf{R}) \leftarrow \mathsf{Exp}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}(1^{\kappa}); \\ (\{\mu_i : i \in [N]\}) \leftarrow \bar{\mathcal{A}}(\mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}) : \\ \qquad \zeta = \mathtt{accept} \wedge \Psi_2 = \prod_{i\in[N]} \left(v_i^{\beta}\right)^{\mu_i} \;\Rightarrow\; \forall \boldsymbol{x}_i \in \mathbf{D} \cap \mathbf{R}^{\complement}, \mu_i = 0\end{array}\right],$$

where v_i^{β} is the second component of tag $\boldsymbol{t}_i$ for data point $\boldsymbol{x}_i \in \mathbf{D}$ (See Step 2 of DEnc *in Figure 2).*

Proof (of Theorem 7).

[9] Since $\mathcal{A}$ is invoked with the same random coin recovered from $\mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$, its behaviors become deterministic.

The idea of proof. For any PPT algorithm $\mathcal{A}$, applying Lemma 6, let $\bar{\mathcal{A}}$ be the PPT algorithm such that $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\mathbf{Lem\ 6}}$ is negligible. Using proof of contradiction, assume that $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\mathbf{Thm\ 7}}$ is non-negligible (**Hypothesis!**). Based on $\mathcal{A}$ and $\bar{\mathcal{A}}$, we construct a PPT algorithm $\mathcal{B}$, such that $\mathcal{B}$ breaks weak-IND-sID-CPA security of FE scheme in Figure 1 with non-negligible advantage

$$\mathsf{Adv}_{\mathsf{FE},\mathcal{B}}^{\text{weak-IND-sID-CPA}} \geq \frac{1}{4dN}\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\mathbf{Thm\ 7}} - \frac{1}{4}\nu_1.$$

where ν_1 is as in Assumption 1 and $\mathsf{Adv}_{\mathsf{FE},\mathcal{B}}^{\text{weak-IND-sID-CPA}}$ is defined in Section 5 The contradiction implies that our hypothesis is wrong, and thus Theorem 7 is proved.

weak-IND-sID-CPA adversary $\mathcal{B}$ against FE scheme Let $\widetilde{\mathcal{E}} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{CollRes})$. We construct the adversary $\mathcal{B}$, which simulates the experiment $\mathsf{Exp}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$ by invoking the adversary $\mathcal{A}$, where $\mathcal{B}$ takes the role of Alice and $\mathcal{A}$ takes the role of Bob. Note that $\mathcal{B}$ makes only one delegation key query.

---

weak-IND-sID-CPA adversary $\mathcal{B}$ against FE scheme

Commit : Initialize $\mathcal{A}$'s status $\mathsf{view}_{\mathcal{A}}$. Invoke adversary $\mathcal{A}(\mathsf{view}_{\mathcal{A}})$, and $\mathcal{A}$ chooses a set $\mathbf{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ of N d-dimensional points in $[\mathcal{Z}]^d$. $\mathcal{B}$ chooses $i^* \in [N]$ at random, and sends $\boldsymbol{x}_{i^*}$ to the challenger $\mathcal{C}$ as the target identity.

Setup : The challenger $\mathcal{C}$ runs the setup algorithm $f\mathsf{Setup}$ and gives $\mathcal{A}$ the resulting system parameters $pk = (p, \mathbb{G}, \widetilde{\mathbb{G}}, e, \Omega)$, keeping the secret key $sk = (pk, \mathsf{params}, \mathsf{master\text{-}key}, \boldsymbol{\tau})$ to itself.

Challenge : $\mathcal{C}$ chooses two plaintexts $\mathsf{Msg}_0, \mathsf{Msg}_1$ at random from the message space $\mathbb{Z}_p^*$, and a random bit $b \in \{0,1\}$. $\mathcal{C}$ sets the challenge ciphertext to $\mathsf{CT} = f\mathsf{Enc}(\mathsf{Msg}_b, \boldsymbol{x}_{i^*}, sk)$, and sends $(\mathsf{CT}, f_1(\mathsf{Msg}_0), f_1(\mathsf{Msg}_1))$ to $\mathcal{B}$.

Learning Phase : $\mathcal{B}$ (playing the role of Alice) interacts with $\mathcal{A}$ (playing the role of Bob) to simulate the experiment $\mathsf{Exp}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$. $\mathcal{B}$ proceeds as below.

KGen : Choose β, γ at random from $\mathbb{Z}_p^*$, and θ at random from $\widetilde{\mathbb{G}}$. Let (pk, sk) be the key pair generated by the challenger $\mathcal{C}$. Generate $\overline{pk}$ by removing Ω from pk, i.e. $\overline{pk} = (p, \mathbb{G}, \widetilde{\mathbb{G}}, e)$. Output $(\overline{pk}, sk)$ *Note: $\mathcal{B}$ knows pk, but not sk.*

DEnc :

1. Choose N random elements $W_1, \ldots, W_N$ from $\mathbb{Z}_p^*$ and N random elements $v_1, \ldots, v_N$ from $\widetilde{\mathbb{G}}$.
2. For each $i \in [N]$ except $i = i^*$, generate a tag $\boldsymbol{t}_i = (\theta v_i, v_i^\beta, f_1(W_i)) \in \widetilde{\mathbb{G}}$. *Note: With Ω, $\mathcal{B}$ can evaluate function $f_\rho(\cdot)$.*
3. For each $i \in [N]$ except $i = i^*$, generate ciphertext CT_i:
   - Issue an encryption query $(W_i, \boldsymbol{x}_i)$ to the challenger $\mathcal{C}$ and get reply CT'_i.
   - Apply the homomorphic property of the functional encryption scheme to attach v_i^γ to the ciphertext: $\mathsf{CT}_i \leftarrow \mathsf{Mult}(\mathsf{CT}'_i, v_i^\gamma, pk)$.
4. Define the tag $\boldsymbol{t}_{i^*}$ and ciphertext CT_{i^*} based on the challenge message $(\mathsf{CT}, f_1(\mathsf{Msg}_0), f_1(\mathsf{Msg}_1))$: Set $\boldsymbol{t}_{i^*} = (\theta v_{i^*}, v_{i^*}^\beta, f_1(\mathsf{Msg}_0))$ and $\mathsf{CT}_{i^*} \leftarrow \mathsf{Mult}(\mathsf{CT}, v_{i^*}^\gamma, pk)$.
5. Send $(\mathbf{D}, \mathbf{T} = \{\boldsymbol{t}_i : i \in [N]\}, \mathbf{C} = \{\mathsf{CT}_i : i \in [N]\}, \overline{pk})$ to $\mathcal{A}$ (Bob).

$\mathcal{A}$ in Learning Phase : $\mathcal{A}$ issues queries $\mathbf{R}_1, \mathbf{R}_2, \ldots$. For each of such queries, $\mathcal{B}$ simulates Alice in $\mathsf{CollRes}$ as below.

Step A1: $\mathcal{B}$ makes a corresponding *anonymous delegation key query* $(\mathbf{R}_i)$ to the challenger $\mathcal{C}$, and sends the reply message $\boldsymbol{\delta}_i$ to $\mathcal{A}$ (Bob).

Step A2: Do nothing. *Note: (1) According to our scheme and formulation, the accept/reject decision is always hidden from $\mathcal{A}$. So there is no need to do verification here. (2) $\mathcal{B}$ does not know the function key ρ_i for the delegation key $\boldsymbol{\delta}_i$, so is not able to perform all verifications in step A2 of* $\mathsf{CollRes}$.

$\mathcal{A}$ in Challenge Phase : $\mathcal{A}$ issues a query with range $\mathbf{R}$: If $\boldsymbol{x}_{i^*} \notin \mathbf{R}$, $\mathcal{B}$ simulates Alice in $\mathsf{CollRes}$ as below.

Step A1: $\mathcal{B}$ chooses a random element $\rho \in \mathbb{Z}_p^*$ and makes a corresponding *delegation key query* $(\mathbf{R}, \rho)$ to the challenger $\mathcal{C}$, and sends the reply message $\boldsymbol{\delta}$ to $\mathcal{A}$ (Bob).

Step A2: Receive response (ζ, X, Ψ_2) for count query $\mathbf{R}$ associated with challenge message $\boldsymbol{\delta}$) from $\mathcal{A}$. Perform all verifications as in Step A2 of $\mathsf{CollRes}$. *Note: In this case $\mathcal{B}$ does know the function key ρ for the delegation key $\boldsymbol{\delta}$ and secret values β, γ, so is able to perform all verifications in step A2 of* $\mathsf{CollRes}$.

Otherwise, if $\boldsymbol{x}_{i^*} \in \mathbf{R}$ then $\mathcal{B}$ abort and outputs a random bit $b' \in \{0,1\}$ (Denote this event as $\mathbf{E}_1$).

Guess : $\mathcal{B}$ outputs a guess bit b' as below.

1. Invoke the extractor $\bar{\mathcal{A}}(\mathsf{view}_{\mathcal{A}})$ for $\mathcal{A}$ and get output $\{\mu_i : i \in [N]\}$.
2. If $\zeta = \mathtt{accept}, \Psi_2 = \prod_{i \in [N]} \left(v_i^\beta\right)^{\mu_i}$ and $\mu_{i^*} \neq 0$, then output $b' = 0$ (Denote this event as $\mathbf{E}_2$).
3. Otherwise, output a random bit $b' \in \{0,1\}$ (Denote this event as $\mathbf{E}_3$).

Note that all three events $\mathbf{E}_1$, $\mathbf{E}_2$ and $\mathbf{E}_3$ are mutually exclusive, and only $\mathbf{E}_2$ is the success case, and both of $\mathbf{E}_1$ and $\mathbf{E}_3$ correspond to failure.

$$\begin{aligned}\mathsf{Pr}[b=b'] &= \mathsf{Pr}\left[\mathbf{E}_1 \vee \mathbf{E}_3\right]\mathsf{Pr}\left[b=b'|\mathbf{E}_1 \vee \mathbf{E}_3\right] + \mathsf{Pr}\left[b=b', \mathbf{E}_2\right] \\ &= (1-\mathsf{Pr}\left[\mathbf{E}_2\right]) \times \frac{1}{2} + \mathsf{Pr}\left[b=b', \mathbf{E}_2\right] \\ &= \frac{1}{2} + \mathsf{Pr}\left[b=b', \mathbf{E}_2\right] - \frac{1}{2}\mathsf{Pr}\left[\mathbf{E}_2\right] \end{aligned} \tag{22}$$

Therefore,

$$\mathsf{Adv}^{\mathsf{IND\text{-}sID\text{-}CPA}}_{\mathsf{FE},\mathcal{B}} = \left|\mathsf{Pr}\left[b=b', \mathbf{E}_2\right] - \frac{1}{2}\mathsf{Pr}\left[\mathbf{E}_2\right]\right| \tag{23}$$

$$\geq \left| \left|\frac{1}{2}\mathsf{Pr}\left[b=b', \mathbf{E}_2 \mid b=0\right] - \frac{1}{4}\mathsf{Pr}\left[\mathbf{E}_2 \mid b=0\right]\right| - \left|\frac{1}{2}\mathsf{Pr}\left[b=b', \mathbf{E}_2 \mid b=1\right] - \frac{1}{4}\mathsf{Pr}\left[\mathbf{E}_2 \mid b=1\right]\right| \right| \tag{24}$$

$\mathsf{Adv}^{\mathsf{weak\text{-}IND\text{-}sID\text{-}CPA}}_{\mathsf{FE},\mathcal{B}}$ conditional on $b=0$.
In case of $b=0$, the forged tag $\boldsymbol{t}_{i^*}$ and ciphertext CT_{i^*} are valid and consistent, and identical to those generated by DEnc. The simulated experiment $\mathsf{Exp}^{\widetilde{\mathcal{B}}}_{\mathcal{A}}$ by $\mathcal{B}$ is *identical* to a real one, to the view of $\mathcal{A}$ (even if $\mathcal{A}$ is computationally unbounded).

Recall that by the hypothesis, $\mathcal{A}$ is a Type II adversary but not a Type III adversary. That is, $\zeta = \mathtt{accept}$ and $\Psi_2 = \prod_{i\in[N]}\left(v_i^{\beta}\right)^{\mu_i}$ with o.h.p, and there exists $\boldsymbol{x}_i \in \mathbf{D}\cap\mathbf{R}^{\complement}$, such that $\mu_i \neq 0$ with non-negligible probability. We denote with $\mathbf{E}_4$ the event that $\zeta = \mathtt{accept}$ and $\Psi_2 = \prod_{i\in[N]}\left(v_i^{\beta}\right)^{\mu_i}$, and there exists $\boldsymbol{x}_i \in \mathbf{D}\cap\mathbf{R}^{\complement}$, such that $\mu_i \neq 0$.

$$\mathsf{Pr}[\mathbf{E}_4 \mid b=0] = \mathsf{Pr}\left[\zeta = \mathtt{accept} \wedge \Psi_2 = \prod_{i\in[N]}\left(v_i^{\beta}\right)^{\mu_i} \wedge \exists \boldsymbol{x}_i \in \mathbf{D}\cap\mathbf{R}^{\complement}, \mu_i \neq 0 \mid b=0\right] = \mathsf{Adv}^{\mathbf{Thm\ 7}}_{\mathcal{A},\bar{\mathcal{A}}}$$

Denote with $\mathbf{E}_5$ the event that $i^* \in S_{\#} = \{i \in [N] : \boldsymbol{x}_i \in \mathbf{D}\cap\mathbf{R}^{\complement}, \mu_i \neq 0\}$. In the case of $b=0$, the event $\mathbf{E}_2$ is equivalent to conjunctions of three events: $\neg\mathbf{E}_1$, $\mathbf{E}_4$, and $\mathbf{E}_5$, i.e. $\mathbf{E}_2 \equiv \neg\mathbf{E}_1 \wedge \mathbf{E}_4 \wedge \mathbf{E}_5$. Since the conjunctions of $\mathbf{E}_4$ and $\mathbf{E}_5$ implies that $\boldsymbol{x}_{i^*} \notin \mathbf{R}$ and $\xi \in [d]$ is independently and randomly chosen, we have

$$\mathsf{Pr}\left[\neg\mathbf{E}_1 \mid \mathbf{E}_4 \wedge \mathbf{E}_5 \wedge b=0\right] = \mathsf{Pr}\left[\boldsymbol{x}_{i^*}[\xi] \notin \mathbf{A}_{\xi} \mid \mathbf{E}_4 \wedge \mathbf{E}_5 \wedge b=0\right] \geq \frac{1}{d}.$$

Therefore,

$$\begin{aligned}\mathsf{Pr}\left[\mathbf{E}_2 \mid b=0\right] &= \mathsf{Pr}\left[\neg\mathbf{E}_1 \wedge \mathbf{E}_4 \wedge \mathbf{E}_5 \mid b=0\right] = \mathsf{Pr}\left[\neg\mathbf{E}_1 \mid \mathbf{E}_4 \wedge \mathbf{E}_5 \wedge b=0\right]\mathsf{Pr}\left[\mathbf{E}_4 \wedge \mathbf{E}_5 \mid b=0\right] \\ &\geq \frac{1}{d}\mathsf{Pr}\left[\mathbf{E}_4 \mid b=0\right]\mathsf{Pr}\left[\mathbf{E}_5 \mid \mathbf{E}_4, b=0\right] \\ &= \frac{1}{d}\mathsf{Adv}^{\mathbf{Thm\ 7}}_{\mathcal{A},\bar{\mathcal{A}}} \cdot \frac{1}{|S_{\#}|} \geq \frac{1}{dN}\mathsf{Adv}^{\mathbf{Thm\ 7}}_{\mathcal{A},\bar{\mathcal{A}}} \qquad (\textit{ Event } \mathbf{E}_4 \textit{ implies that } S_{\#} \neq \emptyset)\end{aligned}$$

According to the construction of $\mathcal{B}$, if $b=0$ and event $\mathbf{E}_2$ occurs, the algorithm $\mathcal{B}$ will output $b'=0$. That is, $\mathsf{Pr}\left[b=b'|\mathbf{E}_2, b=0\right]=1$.

Hence, conditional on $b=0$, the advantage of $\mathcal{B}$ is

$$\mathsf{Adv}^{\mathsf{IND\text{-}sID\text{-}CPA}}_{\mathsf{FE},\mathcal{B}}|_{b=0} = \left|\mathsf{Pr}\left[\mathbf{E}_2 \mid b=0\right](\mathsf{Pr}\left[b=b' \mid \mathbf{E}_2, b=0\right] - \frac{1}{2})\right| \geq \frac{1}{2dN}\mathsf{Adv}^{\mathbf{Thm\ 7}}_{\mathcal{A},\bar{\mathcal{A}}}. \tag{25}$$

$\mathsf{Adv}^{\mathsf{weak\text{-}IND\text{-}sID\text{-}CPA}}_{\mathsf{FE},\mathcal{B}}$ conditional on $b=1$.
Next we show that $\mathsf{Pr}[\mathbf{E}_2 \mid b=1]$ is negligible under Computational Diffie Hellman (**CDH**) assumption.

Claim F.01 *There exists a PPT algorithm which solves Computational Diffie Hellman problem with probability equal to* $\mathsf{Pr}[\mathbf{E}_2 \mid b=1]$.

Proof (of Claim F.01). The proof idea is: Given input (v, v^γ, u), we choose a random number R, and simulate the scheme $\widetilde{\mathcal{E}} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{CollRes})$ by embedding (u, R) into the tag/ciphertext for the target index i^* and embedding (v, v^γ) into tag/ciphertext for the other index, . If $b = 1$ and event $\mathbf{E}_2$ occurs, we try to compute u^γ with the help of adversary $\mathcal{A}$.

Algorithm $\mathcal{D}$: Break Computational Diffie Hellman problem

```
1. Input is $(v, v^a, u) \in \widetilde{\mathbb{G}}^3$, where the unknown exponent $a$ is uniformly randomly distributed over $\mathbb{Z}_p^*$. The goal is to output $u^a$.
2. Simulate the scheme $\widetilde{\mathcal{E}} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{CollRes})$:
   KGen: The same as in Figure 2, except that let $\gamma$ be the unknown value $a$: $\gamma \leftarrow a$.
   DEnc: (a) Choose $N$ random elements $W_1, \ldots, W_N$ from $\mathbb{Z}_p^*$. Choose $i^* \in [N]$ at random.
     (b) For each $i \in [N]$ except $i^*$:
        i. choose $z_i \in \mathbb{Z}_p^*$ at random and compute $v_i = v^{z_i}$ and $v_i^\gamma = (v^\gamma)^{z_i} = (v^a)^{z_i}$;
        ii. generate a tag $\boldsymbol{t}_i = (v_i, v_i^\beta, f_1(W_i)) \in \widetilde{\mathbb{G}}^3$;
        iii. generate a ciphertext $\mathsf{CT}_i$ as in Figure 2.
     (c) For $i^*$:
        i. generate a tag $\boldsymbol{t}_{i^*} = (v_{i^*}, v_{i^*}^\beta, f_1(W_{i^*}))$ where $v_{i^*} = u$;
        ii. generate a ciphertext $\mathsf{CT}_{i^*} = \mathsf{Mult}(\mathsf{CT}',\ R,\ pk)$, where $\mathsf{CT}' \leftarrow f\mathsf{Enc}(W_{i^*},\ \boldsymbol{x}_{i^*},\ sk)$ and $R$ is a random element in $\widetilde{\mathbb{G}}$.
     (d) Send all tags and ciphertexts and $pk$ to Bob as in Figure 2.
   CollRes: The same as in Figure 2, except that the simulator does not perform the verifications in step A2 of CollRes.
      Note: Since $\gamma$ is unknown, some verifications can not be done.
3. Invoke the adversary $\mathcal{A}$ and simulate the experiment $\mathsf{Exp}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$ using the above simulated scheme $\widetilde{\mathcal{E}}$.
   Let $(X, \Psi_1, \Psi_2, \Psi_3, \Psi_4)$ denote the reply returned by adversary $\mathcal{A}$ on the challenging query range $\mathbf{R}$ and $\rho$ be the corresponding random nonce. Note: When the adversary $\mathcal{A}$ is in challenging phase, the verification cannot be done, since $\gamma = a$ is unknown.
4. Let $\mathsf{view}_{\mathcal{A}}$ be the view of $\mathcal{A}$ after the experiment. Invoke the extractor $\bar{\mathcal{A}}$ w.r.t. $\mathcal{A}$, and obtain output: $\{\mu_i : i \in [N]\} \leftarrow \bar{\mathcal{A}}(\mathsf{view}_{\mathcal{A}})$.
5. Compute $\phi$ as below and output $\phi^{\mu_{i^*}^{-1}}$:
   $$\phi \leftarrow \frac{\Psi_4}{\Psi_3^\rho \cdot \prod_{i \in [N]}^{i \neq i^*} (v_i^\gamma)^{\mu_i}}$$
```

Denote the experiment $\mathsf{Exp}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$ simulated by $\mathcal{D}$ as $\mathsf{Exp}_{\mathcal{D}}$; denote the experiment $\mathsf{Exp}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$ simulated by $\mathcal{B}$ in the case of $b = 1$ as $\mathsf{Exp}_{\mathcal{B}}$. Both simulated experiments $\mathsf{Exp}_{\mathcal{D}}$ and $\mathsf{Exp}_{\mathcal{B}}$ are *identical*, to the view of adversary $\mathcal{A}$ (even if $\mathcal{A}$ is computationally unbounded):

- In both simulated experiments, for each $i \in [N]$ except i^*, the tag $\boldsymbol{t}_i$ and ciphertext CT_i are consistent and *identical* as those generated by the algorithm DEnc in Figure 2.
- In both simulated experiments, the ciphertext CT_{i^*} is *independent* on the tag $\boldsymbol{t}_{i^*}$:
  - In $\mathsf{Exp}_{\mathcal{D}}$, $\boldsymbol{t}_{i^*} = (v_{i^*},\ v_{i^*}^\beta,\ f_1(W_{i^*}))$ and ciphertext $\mathsf{CT}_{i^*} = \mathsf{Mult}(\mathsf{CT}',\ R,\ pk)$, where $\mathsf{CT}' \leftarrow f\mathsf{Enc}(W_{i^*},\ \boldsymbol{x}_{i^*},\ sk)$ and R is a random element in $\widetilde{\mathbb{G}}$. That is, the ciphertext CT_{i^*} is randomized due to the independent randomness R in the execution of Mult.
  - In $\mathsf{Exp}_{\mathcal{B}}$, $\boldsymbol{t}_{i^*} = (v_{i^*},\ v_{i^*}^\beta,\ f_1(\mathsf{Msg}_0))$ and $\mathsf{CT}_{i^*} \leftarrow \mathsf{Mult}(\mathsf{CT}, v_{i^*}^\gamma, pk)$, where $\mathsf{CT} \leftarrow f\mathsf{Enc}(\mathsf{Msg}_1,\ \boldsymbol{x}_{i^*},\ sk)$ is the ciphertext of Msg_1, and Msg_0, Msg_1 are two independent random elements in $\mathbb{Z}_p^*$. That is, the ciphertext CT_{i^*} is randomized[10] due to the independent randomness Msg_1 in the execution of $f\mathsf{Enc}$.
- In both simulated experiments, for any range query $\mathbf{R}$, $\mathcal{A}$ receives the same (identically distributed) reply as in $\mathsf{CollRes}$ in Figure 2.

We remark that the differences in the capabilities of verifications in the two simulated experiments, are invisible to $\mathcal{A}$, since all accept/reject decisions are completely hidden from $\mathcal{A}$.

Suppose $b = 1$ and event $\mathbf{E}_2$ occurs[11], that is, $\zeta = \mathtt{accept}$ and $\Psi_2 = \prod_{i \in [N]} \left(v_i^\beta\right)^{\mu_i}$ and $\mu_{i^*} \neq 0$. It is easy to show that

$$\phi = v_{i^*}^{\gamma \mu_{i^*}}; \quad \phi^{\mu_{i^*}^{-1}} = v_{i^*}^\gamma = u^\gamma = u^a.$$

[10] One can verify that randomization in Mult is equivalent to randomization in $f\mathsf{Enc}$, by checking the constructions of Mult and $f\mathsf{Enc}$ and the underlying BBG HIBE scheme. Note that public key params of the underlying BBG HIBE scheme and W_i's (Random numbers as in Step 1 of DEnc in Figure 2) are unknown to the adversary $\mathcal{A}$.

[11] Note that the algorithm $\mathcal{D}$ cannot tell whether $\mathbf{E}_2$ occurs or not, since $\mathcal{D}$ does not know γ thus cannot perform some verifications. $\mathcal{D}$ simply guesses that event $\mathbf{E}_2$ does occur, and this guess will be correct with probability $\Pr[\mathbf{E}_2 | b = 1]$

Hence, the above algorithm $\mathcal{D}$ solve the **CDH** problem with probability

$$\mathsf{Pr}[\mathbf{E}_2 \mid b=1]\ \mathsf{Pr}[\phi^{\mu_{i^*}^{-1}} = u^a \mid \mathbf{E}_2, b=1] = \mathsf{Pr}[\mathbf{E}_2 \mid b=1].$$

□

Therefore, under **CDH** assumption, $\mathsf{Pr}[\mathbf{E}_2 \mid b=1] \leq \nu_1$, where $\nu_1(\cdot)$ is some negligible function. As a result, conditional on $b=1$, the advantage of $\mathcal{B}$ in breaking the FE scheme is

$$\mathsf{Adv}_{\mathsf{FE},\mathcal{B}}^{\text{weak-IND-sID-CPA}}|_{b=1} = \left|\mathsf{Pr}\left[\mathbf{E}_2 \mid b=1\right]\left(\mathsf{Pr}\left[b=b' \mid \mathbf{E}_2, b=1\right] - \frac{1}{2}\right)\right| \leq \frac{1}{2}\nu_1. \tag{26}$$

$$\mathsf{Adv}_{\mathsf{FE},\mathcal{B}}^{\text{weak-IND-sID-CPA}} \geq \left|\frac{1}{2}\mathsf{Adv}_{\mathsf{FE},\mathcal{B}}^{\text{weak-IND-sID-CPA}}|_{b=0} - \frac{1}{2}\mathsf{Adv}_{\mathsf{FE},\mathcal{B}}^{\text{weak-IND-sID-CPA}}|_{b=1}\right| \geq \frac{1}{4dN}\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 7}} - \frac{1}{4}\nu_1.$$

□

G A valid proof should be generated by processing each point within intersection $\mathbf{D} \cap \mathbf{R}$ *for exactly once*

Theorem 8 *Suppose Assumption 1 and Assumption 2 hold, and BBG [2] HIBE scheme is IND-sID-CPA secure. For any PPT algorithm $\mathcal{A}$, there exists a PPT adversary $\bar{\mathcal{A}}$, such that all of $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem 6}}$, $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 7}}$, and $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 8}}$ are negligible, where the advantage $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 8}}$ of $\mathcal{A}$ against $\bar{\mathcal{A}}$ w.r.t. scheme $\mathcal{E} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{ProVer})$ is defined as*

$$\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 8}}(1^\kappa) \stackrel{\text{def}}{=} 1 - \mathsf{Pr}\left[\begin{array}{l}(\zeta, X, \varDelta, \mathsf{view}_{\mathcal{A}}^{\mathcal{E}}, \mathbf{D}, \mathbf{R}) \leftarrow \mathsf{Exp}_{\mathcal{A}}^{\mathcal{E}}(1^\kappa);\\ (\{\mu_i : i \in [N]\}) \leftarrow \bar{\mathcal{A}}(\mathsf{view}_{\mathcal{A}}^{\mathcal{E}}) :\\ \qquad \zeta = \mathtt{accept} \Rightarrow \left(\varDelta = \prod_{i\in[N]}\left(v_i^\beta\right)^{\mu_i} \wedge \forall i \in [N], \mu_i = 1\right)\end{array}\right],$$

where v_i^β is the second component of tag $\mathbf{t}_i$ for data point $\boldsymbol{x}_i \in \mathbf{D}$ (See Step 2 of DEnc *in Figure 2).*

Proof (of Theorem 8).

Idea of proof. For any PPT algorithm $\mathcal{A}$, applying Theorem 7, let $\bar{\mathcal{A}}$ be the PPT algorithm, such that $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem 6}} \leq \epsilon_5$ and $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 7}} \leq \epsilon_6$ for some negligible functions $\epsilon_5(\cdot)$ and $\epsilon_6(\cdot)$. Using proof of contradiction, assume that $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 8}} \geq \epsilon_7$ for some non-negligible function $\epsilon_7(\cdot)$. We construct a PPT algorithm $\mathcal{B}$ based on $\mathcal{A}$ and $\bar{\mathcal{A}}$, such that $\mathcal{B}$ breaks Discrete Log Problem with non-negligible advantage $\epsilon_7 - (2d+1)(\epsilon_5 + \epsilon_6)$.

Denote with $\mathbf{E}_1$ the event that $\zeta = \mathtt{accept} \bigwedge \varDelta \neq \prod_{i\in[N]}\left(v_i^\beta\right)^{\mu_i}$, and with $\mathbf{E}_2$ the event that $\zeta = \mathtt{accept} \bigwedge \varDelta = \prod_{i\in[N]}\left(v_i^\beta\right)^{\mu_i} \wedge \exists j \in [N], \mu_j \neq 1$. We can split the probability $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 8}}$ into two parts,

$$\begin{aligned}\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 8}} &= \mathsf{Pr}\left[\begin{array}{l}(\zeta, X, \boldsymbol{\varPsi}, \mathsf{view}_{\mathcal{A}}^{\mathcal{E}}, \mathbf{D}, \mathbf{R}) \leftarrow \mathsf{Exp}_{\mathcal{A}}^{\mathcal{E}}(1^\kappa);\\ (\{\mu_i : i \in [N]\}) \leftarrow \bar{\mathcal{A}}(\mathsf{view}_{\mathcal{A}}^{\mathcal{E}}) :\\ \qquad \zeta = \mathtt{accept} \bigwedge \varDelta \neq \prod_{i\in[N]}\left(v_i^\beta\right)^{\mu_i}\end{array}\right] + \mathsf{Pr}\left[\begin{array}{l}(\zeta, X, \boldsymbol{\varPsi}, \mathsf{view}_{\mathcal{A}}^{\mathcal{E}}, \mathbf{D}, \mathbf{R}) \leftarrow \mathsf{Exp}_{\mathcal{A}}^{\mathcal{E}}(1^\kappa);\\ (\{\mu_i : i \in [N]\}) \leftarrow \bar{\mathcal{A}}(\mathsf{view}_{\mathcal{A}}^{\mathcal{E}}) :\\ \qquad \zeta = \mathtt{accept} \bigwedge \varDelta = \prod_{i\in[N]}\left(v_i^\beta\right)^{\mu_i}\\ \wedge \exists j \in [N], \mu_j \neq 1\end{array}\right]\\ &= \mathsf{Pr}[\mathbf{E}_1] + \mathsf{Pr}[\mathbf{E}_2].\end{aligned}$$

Part I: $\mathsf{Pr}[\mathbf{E}_1] \leq (2d+1)\left(\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem 6}} + \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 7}}\right)$. Suppose that: (1) The challenging query range is $\mathbf{R}$. (2) Alice partitions $\mathbf{R}^{\complement}$ into $2d$ rectangular ranges $\mathbf{R}_1, \ldots, \mathbf{R}_{2d}$ and sets $\mathbf{R}_0 = \mathbf{R}$. (3) For $0 \leq \ell \leq 2d$, denote with $(\zeta_\ell, X_\ell, \varPsi_2^{(\ell)})$ the reply returned by adversary $\mathcal{A}$ in the execution of $\mathsf{CollRes}$ on range $\mathbf{R}_\ell$. (4) Denote with $(\zeta, X, \varDelta)$ the output of Alice in the execution of ProVer. (5) Recall that Alice keeps the value $\varDelta = \prod_{i\in[N]} v_i^\beta$.

According to the construction in Figure 2 (i.e. Step 3 of ProVer), we have

$$\left(\bigwedge_{\ell\in[0,2d]} \zeta_\ell = \mathtt{accept}\right) \wedge \varDelta = \prod_{\ell\in[0,2d]} \varPsi_2^{(\ell)} \Leftrightarrow \zeta = \mathtt{accept} \qquad (\textit{Denoted as statement } A) \tag{27}$$

In additional to statement A, let us define statement A_ℓ and B_ℓ as below:

A_ℓ**:** $\zeta_\ell = \texttt{accept} \;\;\Rightarrow\;\; \Psi_2^{(\ell)} = \prod_{i\in[N]} v_i^{\beta\mu_i}$, $0 \le \ell \le 2d$.
B_ℓ**:** $\zeta_\ell = \texttt{accept} \wedge \Psi_2^{(\ell)} = \prod_{i\in[N]} v_i^{\beta\mu_i} \;\;\Rightarrow\;\; \forall \boldsymbol{x}_i \in \mathbf{D} \cap \mathbf{R}^{\complement}, \mu_i = 0$, $0 \le \ell \le 2d$.

The conjunctions of statements A, A_ℓ's ($0 \le \ell \le 2d$), and B_ℓ's ($0 \le \ell \le 2d$), directly imply that

$$\zeta = \texttt{accept} \qquad \Rightarrow \qquad \Delta = \prod_{\boldsymbol{x}_i \in \mathbf{D} \cap \left(\bigcup_{0\le\ell\le 2d} \mathbf{R}_\ell\right)} v_i^{\beta\mu_i} = \prod_{\boldsymbol{x}_i\in\mathbf{D}} v_i^{\beta\mu_i}. \tag{28}$$

Applying Proposition 1 and Proposition 2 in Appendix B, we have

$$\begin{aligned}\mathsf{Pr}\left[\zeta = \texttt{accept} \Rightarrow \Delta = \prod_{\boldsymbol{x}_i\in\mathbf{D}} v_i^{\beta\mu_i}\right] &\ge \mathsf{Pr}\left[A \wedge A_0 \wedge \ldots \wedge A_{2d} \wedge B_0 \wedge \ldots \wedge B_{2d}\right] \\ &\ge 1 - \mathsf{Pr}[\neg A] - \sum_{\ell=0}^{2d} \mathsf{Pr}\left[\neg A_\ell\right] - \sum_{\ell=0}^{2d} \mathsf{Pr}\left[\neg B_\ell\right] \\ &\ge 1 - 0 - \sum_{\ell=0}^{2d} \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem 6}} - \sum_{\ell=0}^{2d} \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 7}} \\ &= 1 - (2d+1)\left(\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem 6}} + \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 7}}\right).\end{aligned}$$

Therefore,

$$\mathsf{Pr}[\mathbf{E}_1] = 1 - \mathsf{Pr}\left[\zeta = \texttt{accept} \Rightarrow \Delta = \prod_{\boldsymbol{x}_i\in\mathbf{D}} v_i^{\beta\mu_i}\right] \le (2d+1)\left(\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem 6}} + \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 7}}\right).$$

Part II: Break Discrete Log Problem. Applying the result in Part I, we have $\mathsf{Pr}[\mathbf{E}_2] = \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 8}} - \mathsf{Pr}[\mathbf{E}_1] \ge \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 8}} - (2d+1)\left(\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem 6}} + \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 7}}\right)$. We construct the following algorithm to break the Discrete Log Problem.

DLP Adversary $\mathcal{B}$

1. The input is $(v, v^a) \in \widetilde{\mathbb{G}}^2$. The goal is to find $a \in \mathbb{Z}_p$.
2. Invoke scheme $\mathcal{E} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{ProVer})$ with f_2 defined as above, with the following modification:
   – In DEnc, for each $i \in [N]$, choose $y_i, z_i \in \mathbb{Z}_p$ at random and set $v_i = (v^a)^{y_i} \cdot v^{z_i} \in \widetilde{\mathbb{G}}$.
   Note: $\mathcal{B}$ has full information of private key.
3. Simulate the experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathcal{E}}$, by invoking the adversary $\mathcal{A}$ (playing the role Bob) to interact with Alice in $\mathcal{E}$. Then invoke $\bar{\mathcal{A}}(\mathsf{view}_{\mathcal{A}}^{\mathcal{E}})$ to obtain $\{\mu_i : i \in [N]\}$.
4. With probability equal to $\mathsf{Pr}[\mathbf{E}_2]$, it holds that $\zeta = \texttt{accept} \;\bigwedge\; \Delta = \prod_{i\in[N]} \left(v_i^\beta\right)^{\mu_i} \;\bigwedge\; \exists j \in [N], \mu_j \ne 1$.
5. According to our scheme in Figure 2 (Step 4 of DEnc), $\Delta = \prod_{i\in[N]} v_i^\beta$. So a univariable equation in the unknown variable a of order 1 in group $\mathbb{Z}_p$ can be formed by substituting $v_j = v^{ay_j+z_j}$. Solve this equation and get a root a^*. Output a^*.

The PPT algorithm $\mathcal{B}$ constructed as above breaks **DLP** with probability $\mathsf{Pr}[\mathbf{E}_2]$. Therefore, under Computational Diffie Hellman Assumption 1, **DLP** is infeasible and thus $\mathsf{Pr}[\mathbf{E}_2]$ has to be negligible.

Combining results in Part I and II, we have

$$\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 8}} \le (2d+1)\left(\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem 6}} + \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 7}}\right) + \mathsf{Adv}_{\mathcal{B}}^{\textbf{DLP}}.$$

□

H Proof of Main Theorem 3

Theorem 3 (Main Theorem) *Suppose Assumption 1 and Assumption 2 hold, and BBG [2] HIBE scheme is IND-sID-CPA secure. Then the $\mathcal{RC}$ protocol $\mathcal{E} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{ProVer})$ constructed in Figure 2 is $\mathcal{VRC}$ w.r.t. function $F(\cdot,\cdot)$ as defined in Section 4.1, under Definition 2. Namely, $\mathcal{E}$ is* correct *and* sound *w.r.t. function F.*

Proof (of Theorem 3). The correctness is straightforward once we have Lemma 1. Here we save the details and focus on the soundness part.

Suppose $\mathcal{E}$ is not sound, i.e. there exists a PPT algorithm $\mathcal{A}$, with non-negligible advantage ϵ_6 against $\mathcal{E}$:

$$\mathsf{Adv}^{\mathcal{E}}_{\mathcal{A}} = \mathsf{Pr}\left[\begin{array}{c}(\zeta, X, \boldsymbol{\Psi}, \mathsf{view}^{\mathcal{E}}_{\mathcal{A}}, \mathbf{D}, \mathbf{R}) \leftarrow \mathsf{Exp}^{\mathcal{E}}_{\mathcal{A}}(1^{\kappa});\\ \zeta = \mathtt{accept} \ \bigwedge \ X \neq F(\mathbf{D}, \mathbf{R}) \pmod p\end{array}\right] \geq \epsilon_6.$$

Applying Theorem 8, let $\bar{\mathcal{A}}$ be the extractor for $\mathcal{A}$ such that all of $\mathsf{Adv}^{\mathbf{Lem\ 6}}_{\mathcal{A},\bar{\mathcal{A}}}$, $\mathsf{Adv}^{\mathbf{Thm\ 7}}_{\mathcal{A},\bar{\mathcal{A}}}$, and $\mathsf{Adv}^{\mathbf{Thm\ 8}}_{\mathcal{A},\bar{\mathcal{A}}}$ are negligible.

We intend to construct a PPT algorithm $\mathcal{B}$ based on $\mathcal{A}$ to break Assumption 1 (Computational Diffie-Hellman Problem), and argue that $\mathcal{B}$ succeeds with probability about ϵ_6, with the help of $\bar{\mathcal{A}}$, under Assumption 1, Assumption 2, and the assumption that BBG [2] HIBE is IND-sID-CPA secure. The contradiction will imply that such adversary $\mathcal{A}$ does not exist and the constructed scheme $\mathcal{E}$ is sound.

Adversary $\mathcal{B}$ against Computational Diffie-Hellman Problem

1. The input is $(u, u^{\beta}, v^{\beta}) \in \widetilde{\mathbb{G}}$. The goal is to find v.
2. Choose a random number R_1 from $\widetilde{\mathbb{G}}$. Then $R_1 = v\theta$ for some unknown $\theta \in \widetilde{\mathbb{G}}$.
3. For $1 \leq j \leq m$, choose z_j at random from $\mathbb{Z}_p^*$ and set $u_j \leftarrow u^{z_j}$ and compute $u_j^{\beta} = \left(u^{\beta}\right)^{z_j}$. Let $W_m = (\{u_j, u_j^{\beta} : j \in [m]\})$.
4. Convert $(W_m, R_1, R_2 = v^{\beta})$ to $S_{m+1} = \{(\theta v_i, v_i^{\beta})\}_{i=0}^{m}$ in the same way as in construction of algorithm $\mathcal{A}_1$ in the proof of Lemma 5 in Appendix E.1.
5. From S_{m+1}, simulate the scheme $\mathcal{E}$ just as adversary $\mathcal{B}$ in the proof of Lemma 6 in Appendix E.2.
6. Invoke the adversary $\mathcal{A}$ and simulate the experiment $\mathsf{Exp}^{\mathcal{E}}_{\mathcal{A}}$. Let $(X, \bar{\Psi}_1, \bar{\Psi}_2, \bar{\Psi}_3, \bar{\Psi}_4)$ be the reply returned by adversary $\mathcal{A}$ on challenging query range $\mathbf{R}$ in the execution of CollRes.
7. Simulate the experiment $\mathsf{Exp}^{\mathcal{E}}_{\mathcal{A}}$ honestly (just using the algorithm Eval instead of adversary $\mathcal{A}$) and get query result $Y = |\mathbf{D} \cap \mathbf{R}|$ and proof $(\Psi_1, \Psi_2, \Psi_3, \Psi_4)$.
8. Let Z be the inverse of $(X - Y)$ modulo p and compute $\theta' = \left(\frac{\bar{\Psi}_1}{\Psi_1}\right)^{Z}$.
   Note: (1) $Y = F(\mathbf{D}, \mathbf{R})$. (2) If $\mathcal{A}$ succeeds, then $X \neq F(\mathbf{D}, \mathbf{R}) \pmod p$. Recall the definition of function $F : \mathbb{D} \times \mathbb{R} \to \mathbb{Z}_p$ in Section 4.1.
9. Output $\frac{R_1}{\theta'}$.

Note that as in proof of Lemma 6, the simulated scheme $\mathcal{E}$ is identical to a real one from the view of adversary $\mathcal{A}$.

For the constructed adversary $\mathcal{B}$, we make the following claim:

Claim H.02 *Suppose Assumption 1 and Assumption 2 hold, and BBG [2] HIBE scheme is IND-sID-CPA secure. If $\mathcal{A}$ succeeds, it holds with o.h.p. (i.e. with probability $(1 - negl)$) that $\left(\frac{\bar{\Psi}_1}{\theta^X}\right)^{\beta} = \bar{\Psi}_2 = \Psi_2 = \left(\frac{\Psi_1}{\theta^Y}\right)^{\beta}$.*

Proof (of Claim H.02). If $\mathcal{A}$ succeeds, then its output $(X, \bar{\Psi}_1, \bar{\Psi}_2, \bar{\Psi}_3, \bar{\Psi}_4)$ will pass all verifications in the scheme $\mathcal{E}$ (Step A2 of CollRes and Step 3 in ProVer in Figure 2). So we have

$$\left(\frac{\bar{\Psi}_1}{\theta^X}\right)^{\beta} = \bar{\Psi}_2, \ \zeta = \mathtt{accept}. \tag{29}$$

where $\zeta \in \{\mathtt{accept}, \mathtt{reject}\}$ denotes the corresponding decision (a part of output of ProVer) regarding $\mathcal{A}$'s reply on the challenging query.

Let $(\mu_1, \ldots, \mu_N)$ be the output of extractor $\bar{\mathcal{A}}$. Under Assumption 1, Assumption 2 and the assumption that BBG [2] HIBE scheme is IND-sID-CPA secure, by applying Lemma 6, Theorem 7 and Theorem 8, the following implications hold with o.h.p.,

$$\zeta = \mathtt{accept} \ \Rightarrow \ \left(\Delta = \prod_{i \in [N]} \left(v_i^{\beta}\right)^{\mu_i} \ \wedge \ \forall i \in [N], \mu_i = 1\right);$$

$$\zeta = \mathtt{accept} \ \Rightarrow \ \bar{\Psi}_2 = \prod_{\boldsymbol{x}_i \in \mathbf{D} \cap \mathbf{R}} \left(v_i^{\beta}\right)^{\mu_i}.$$

Hence, conditional on $\mathcal{A}$ succeeds, with o.h.p. we have

$$\bar{\Psi}_2 = \prod_{\boldsymbol{x}_i \in \mathbf{D} \cap \mathbf{R}} \left(v_i^{\beta}\right)^{\mu_i} = \prod_{\boldsymbol{x}_i \in \mathbf{D} \cap \mathbf{R}} v_i^{\beta}. \tag{30}$$

The output $(X, \Psi_1, \Psi_2, \Psi_3, \Psi_4)$ returned by an honest Bob also passes all verifications (Since the scheme $\mathcal{E}$ is *correct*).

$$\left(\frac{\Psi_1}{\theta^Y}\right)^{\beta} = \Psi_2, \text{ where } \Psi_2 = \prod_{\boldsymbol{x}_i \in \mathbf{D} \cap \mathbf{R}} v_i^{\beta} \quad \text{is computed following the scheme.} \tag{31}$$

Combing equations (29)(30)(31), Claim H.02 can be implied directly:

$$\left(\frac{\bar{\Psi}_1}{\theta^X}\right)^{\beta} = \bar{\Psi}_2 = \Psi_2 = \left(\frac{\Psi_1}{\theta^Y}\right)^{\beta}.$$

□

From Claim H.02, it is straightforward that

$$\mathsf{Pr}\left[\frac{R_1}{\theta'} = v\right] = \mathsf{Pr}\left[\theta' = \theta\right] \geq \mathsf{Pr}\left[\mathcal{A} \text{ succeeds}\right](1 - negl) \geq \epsilon_6(1 - negl),$$

where $negl(\cdot)$ is some negligible function. Therefore, the constructed algorithm $\mathcal{B}$ breaks Assumption 1 with non-negligible probability $\epsilon_6(1 - negl)$. The contradiction implies that our hypothesis is wrong: such adversary $\mathcal{A}$ does not exist. Thus, the constructed scheme $\mathcal{E}$ is sound and Theorem 3 is proved. □