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 a 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 method that requires $O(d^2 \log^2 \mathcal{Z})$ communication bits to authenticate an aggregate count query conditional on d-dimensional range selection, where each data point is in the domain $[1, \mathcal{Z}]^d$ and $\mathcal{Z}$ is an integer. Our solution is based on a functional encryption scheme which we specially design by exploiting a special property of BBG [1] HIBE scheme. We prove that our solution is "secure" under Generalized Knowledge of Exponent Assumption proposed by Wu and Stinson [2]. Besides counting, our solution can be extended to support summing, finding of the minimum and usual (non-aggregate) range selection with similar complexity.

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.

We are concerned about the communication cost and the storage overhead on Alice/Bob's side. Such requirements exclude the following two 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.

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].

Recently, Gennaro *et al.* [6] and Chung *et al.* [7] showed that, in principle any outsourced/delegated function can be verified efficiently (i.e. via a PPT algorithm) using a private verification key in the outsourced computation model, by using the Fully Homomorphic Encryption [8]. It is meaningful to devise more efficient scheme for small class of functions, without using fully homomorphic encryption.

1.1 Our results

We propose a scheme to authenticate aggregate range query over static multidimensional outsourced dataset. For a dataset $\mathbf{D} \subset [1, \mathcal{Z}]^d$ with N d-dimensional points, the number of communication bits required is in $O(d^2 \log^2 \mathcal{Z})$ per query. The storage overhead on Bob's side is $O(dN)$, which is linear w.r.t. the storage size of $\mathbf{D}$. If the dataset $\mathbf{D}$ is *normalized*[1] [9], 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

[1] For any dataset with size N, one can normalized [9] 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.

knowledge, this is the first solution with worst case communication overhead sublinear in the number of points in the dataset and with constant storage on verification side and polynomial storage on server side, without using fully homomorphic encryption scheme [8, 10].

We now illustrate our main ideas in three steps: (1) We describe a preliminary scheme to authenticate count query. This preliminary scheme requires large amount of communication bits and computation overhead on client side, but can be proved secure (Theorem 4) under certain assumptions. (2) We describe our strategy and key technique used in the main scheme (Section 4) to reduce the communication overhead and computation overhead. (3) We describe the computational assumptions required by our security proofs of the preliminary scheme and the main scheme.

Preliminary Scheme Let $\mathbf{D} \subset [\mathcal{Z}]^d$ be a set of d-dimensional points. Let G be a cyclic multiplicative group of prime order p. During setup, Alice chooses $\beta \in \mathbb{Z}_p^*$, $\theta \in G$ and a secret function $f : [\mathcal{Z}]^d \to G$ as the private key. From the private key, Alice generates a tag value $t_x = (t_{x,1}, t_{x,2}) = (\theta f(x), f(x)^\beta)$ 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 $(\beta, \theta, f(\cdot))$ from her storage.

Let us consider a count query conditional on range $\mathbf{R} \subset [\mathcal{Z}]^d$, which asks for the the size of intersection set $\mathbf{D} \cap \mathbf{R}$. 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 process this query, Alice chooses two random nonces[2] ρ and $\overline{\rho}$, computes and sends auxiliary messages[3] (called as *Help-Info*) $\Phi = \{f(x)^\rho : x \in \mathbf{R}\}$ and $\overline{\Phi} = \{f(x)^{\overline{\rho}} : x \in \mathbf{R}^{\complement}\}$ to Bob , in order to help Bob to generate a proof for the query result. Bob is expected to compute $X = |\mathbf{D} \cap \mathbf{R}|$ and $\overline{X} = |\mathbf{D} \cap \mathbf{R}^{\complement}|$, and generate the proof $(\Psi_1, \Psi_2, \Psi_3, \overline{\Psi}_1, \overline{\Psi}_2, \overline{\Psi}_3)$ as below:

Step 1: Bob multiplies all tags t_x for point $x \in \mathbf{D} \cap \mathbf{R}$ to obtain Ψ_1, Ψ_2

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

Step 2: Bob multiplies all values $f(x)^\rho$ from the *Help-Info* Φ for point $x \in \mathbf{D} \cap \mathbf{R}$ to obtain Ψ_3:

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

Step 3: Bob repeats Step 1 and Step 2 for data points $x \in \mathbf{D} \cap \mathbf{R}^{\complement}$ using *Help-Info* $\overline{\Phi}$ to obtain $\overline{\Psi}_1, \overline{\Psi}_2, \overline{\Psi}_3$ correspondingly.

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

Step 1: Is (Ψ_1, Ψ_2) indeed an aggregated multiplication of valid tags?

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

Step 2: Is Ψ_2 computed using *only* points inside $\mathbf{D} \cap \mathbf{R}$?

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

Step 3: Repeat Step 1 and Step 2 to verify $(\overline{X}, \overline{\Psi}_1, \overline{\Psi}_2, \overline{\Psi}_3)$ using private key and secret random nonce $\overline{\rho}$.
Step 4: Is every point counted for *exactly* once?

$$\Delta \stackrel{?}{=} \Psi_2 \cdot \overline{\Psi}_2. \tag{1}$$

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

[2] Here the secret random nonces prevent Bob from abusing this *Help-Info* for other queries.
[3] Note that Alice is able to enumerate all points within the query range $\mathbf{R}$. But she is not able to enumerate points within $\mathbf{D}$ after the setup phase.

Remark.

- Intuitively, for point $x \notin \mathbf{R}$, Bob does not have $f(x)^\rho$; for point $x \notin \mathbf{D}$, Bob does not have t_x. Only for point $x \in \mathbf{D} \cap \mathbf{R}$, Bob can provide both t_x and $f(x)^\rho$.
- In the computations of Ψ_1 and Ψ_2, an adversary (playing the role of Bob) may multiply tags for some points within $\mathbf{D} \cap \mathbf{R}$ multiple times, and/or ignore some points within $\mathbf{D} \cap \mathbf{R}$, and try to pass the equality test in equation (1). If such adversary succeeds, that means, the adversary can find integers μ_x's, $x \in \mathbf{D}$, such that $\prod_{x\in\mathbf{D}} t_{x,2}^{\mu_x} = \Delta = \prod_{x\in\mathbf{D}} t_{x,2}$ and for some x, $\mu_x \neq 1$, where $\mu_x > 1$ indicates that the point x is *double counted*, $\mu_x < 1$ indicates that the point x is *undercounted*, and μ_x might take negative integer value. This adversary could be utilized to solve DLP (Discrete Log Problem) (See the sketch proof of Theorem 4 in Section 5.3). In other words, such adversary does not exist under assumption that DLP is hard.
- In the preliminary scheme, the size of *Help-Info* is linear w.r.t. the size of query range $\mathbf{R}$, which could be huge and possibly comparable to the domain size $\mathcal{Z}^d$ of a data point. This implies large computation cost on client (Alice) side (to generate *Help-Info*), and large communication cost (to send *Help-Info*).
- The second component $t_{x,2} = f(x)^\beta$ 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 *Help-Info* by itself from $\{f(x)^\beta : x \in \mathbf{D}\}$, and the forged *Help-Info* is identically distributed to the *Help-Info* generated by Alice.

Deliver *Help-Info* efficiently and securely To reduce complexity, Alice needs a way to deliver the information $\Phi = \{f(x)^\rho : x \in \mathbf{R}\}$ to Bob by sending only some auxiliary data $\boldsymbol{\delta}$ of much smaller size, and Bob should not know the value of $f(x)^\rho$ 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 [1], satisfies a *polymorphic property*: From a pair of keys $(pk, sk) \in \mathsf{KeyGen}(1^\kappa)$, a plaintext M, an identity $\mathtt{id}$, and a random coin r, one can efficiently find multiple tuples $(pk_j, sk_j, M_j, r_j), 1 \leq j \leq n$, such that for any $1 \leq j \leq n$, $(pk_j, sk_j) \in \mathsf{KeyGen}(1^\kappa)$ is a valid key pair and

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

From the opposite point of view, a ciphertext can be decrypted into different values using different decryption keys. We can view these decrypted values as a function of the original plaintext which is used to produce the ciphertext. Hence, such polymorphic property may lead to a new way to construct functional encryption schemes [11, 12].

Overview. Alice can deliver the *Help-Info* in this way: For simplicity, assume all data points are in 1D and the size of dataset $\mathbf{D}$ is N. 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, Alice computes some ciphertexts $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_N$, where each ciphertext $\boldsymbol{c}_i$ 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 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 *Help-Info* to Bob. With this delegation key, Bob is able to decrypt $\boldsymbol{c}_i$ to obtain $M_{i,\rho}$ if $x_i \in \mathbf{D} \cap \mathbf{R}$. By carefully choosing parameters, we may have $M_{i,\rho} = f(x_i)^\rho$ as desired.

In a HIBE scheme, every identity corresponds to a tree node (either leaf node or internal node). Due to the tree structure of the hierarchy of identities, we can find a set S' of identities, such that (1) The size $|S'| = O(\log \text{TreeSize}) = O(\log \mathcal{Z})$; (2) The collection of leaf nodes *covered*[4] by tree nodes corresponding to identities in S', is the same as the collection of leaf nodes corresponding to identities in S. If Alice derives the delegation key $\boldsymbol{\delta}$ w.r.t. S' instead of S, then the new *Help-Info* will contain only $O(\log \mathcal{Z})$ subkeys, where one subkey corresponds to one identity in S'.

For high dimensional cases, we perform the above procedure for each dimension, and have a way to prevent collusion attack [13]. The security of this method can be reduced to the IND-sID-CPA security of the underlying HIBE scheme.

[4] We say a leaf node u is covered by a tree node v, if v is in the path from leaf u to the root node. It is possible that $u = v$.

Assumptions Our security proof relies on Computational Diffie-Hellman[5] Assumption 1 [14] and Generalized Knowledge of Exponent (**GKEA**) Assumption 2 [2], where both assumptions are over the target group of a bilinear map. The **GKEA** assumption is an extension of **KEA1** [15,16,17,18,19] and **KEA3** [20], and proposed by Wu and Stinson [2]. 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 $\prod_{i=1}^m u_i^{\mu_i} = U_1$.*

GKEA can be proved secure in the generic group model, using the same technique for proof of **KEA1** and **KEA3** by M. Abe and S. Fehr [21].

Additionally, the (Decision) ℓ-wBDHI Assumption [1] is required for the IND-sID-CPA security of the underlying BBG HIBE scheme.

Contribution

1. We propose a functional encryption scheme in Section 3 by exploiting a special property (we call it "polymorphic property") of the BBG HIBE scheme [1]. In this functional encryption scheme, a message is encrypted under a d-dimensional point using the *private*[6] key. A delegation key w.r.t. a d-dimensional range $\mathbf{R}$ and a random nonce ρ can be generated from the private key. With this delegation key and a ciphertext for message Msg under point $\boldsymbol{x}$, the decryption algorithm will output a function[7] value $f_1(\mathsf{Msg})^\rho$ of the message Msg iff $\boldsymbol{x} \in \mathbf{R}$. The size of a private key is in $O(\ell + d)$, the size of a ciphertext is in $O(d)$, and the size of a delegation key is in $O(d\ell^2)$, where $\ell = \lceil \log \mathcal{Z} \rceil$ is the depth of the identity hierarchy tree of BBG HIBE scheme and each point is in domain $[\mathcal{Z}]^d$.
2. We prove that the proposed functional encryption scheme is weak-IND-sID-CPA secure (as defined in Section 3.3), if BBG HIBE scheme [1] is IND-sID-CPA secure (See Theorem 2).
3. We propose a new authentication scheme, by incorporating the functional encryption scheme into the preliminary scheme in Section 4. The resulting scheme is efficient in authenticating multidimensional aggregate count query: Communication overhead is $O(d^2 \log^2 \mathcal{Z})$ for a d-dimensional aggregate range query, and the storage overhead on Bob's side is $O(dN)$.
4. We prove that the proposed authentication scheme is secure (Theorem 3) under reasonable assumptions (Assumption 1, Assumption 2 and ℓ-wBDHI Assumption [1]). We describe our proof strategy in Section 5 and illustrate it by proving that the preliminary scheme in Section 1 is secure. Due to the space constraint, we put the full proof for the main scheme in appendix.

We remark that our scheme can be extended to support other types of aggregate range query with similar complexity, including summing, finding of minimum, maximum or median, and even non-aggregate range selection query. For the simplicity of presentation, we focus on counting in this paper.

1.2 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. [4,22,23,24], and (2) integrity (i.e. authenticate the soundness and completeness of query results returned by the service provider) e.g. [3, 25, 26, 27, 5, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39]. In the latter aspect, a lot of works are done for "identity query" [29], 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. [28, 38, 39]) are devoted to the authentication of aggregate query.

There are roughly four categories of approaches for outsourced database authentication in the literatures [3, 25, 26, 27, 5, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37]. (1) (Homomorphic and/or aggregatable) Cryptographic primitives, like collision-resistant hash, digital signature, commitment [5, 40, 38]. (2) Geometry partition and authenticated

[5] Note that Diffie-Hellman Assumption implies Discrete Log Assumption.

[6] Unlike [11,12], our functional encryption scheme is a symmetric key encryption system. However, in the case that dimension $d = 1$, our functional encryption scheme can become a public key encryption scheme.

[7] The function $f_1(\cdot)$ is defined later in Section 3.3.

data structure [25,30,33,35,31,39]. For example, Merkle Hash Tree (typically for 1D case) and variants, KD-tree with chained signature [28], R-Tree with chained signature [30], and authenticated B-Tree/R-Tree [39]. (3) Authenticated precomputed partial result, e.g. authenticated prefix sum [33,39] (the static case solution in [39]) (4) Inserting and auditing fake tuples [32]. Instead of leveraging on the standard or existing cryptographic primitives (e.g. digital signature scheme, cryptographic hash) like most of previous works, this paper proposes a new functional encryption scheme, and uses it to construct a new homomorphic authentication scheme. Consequently, this paper achieves very good asymptotic performance, but the proof of security becomes much more challenging.

To the best of our knowledge, the existing few works (e.g. [28, 38, 39]) on authentication of aggregate query either only deal with 1D case, or have communication overhead[8] linear (or even superlinear) w.r.t. the number of data points in the query range, and/or exponential in dimension. Even for multidimensional (non-aggregate) range selection query, the communication overhead is still in $O(\log^{d-1} N + |S|)$ (Martel *et al.* [25], Chen *et al.* [41]), 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) function, based on fully homomorphic encryption [8,10,42]. 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 complexity 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 [43]) sill requires $O(d^2 \log^2 \mathcal{Z})$ communication cost.

Most of previous functional encryption schemes (e.g. attribute-based encryption, ciphertext-policy encryption, or predicate encryption), if not all, allow the decryptor to obtain the original plaintext in "good" case (e.g. if the plaintext and/or the decryption key satisfy some predicate), and nothing otherwise. In contrast, the functional encryption scheme proposed in this paper only allows the decryptor to obtain $f_1(\mathsf{Msg})^\rho$ in "good" case, where Msg is the plaintext and $f_1(\cdot)$ is a function defined in Section 3.3. Our security formulation is weaker than previous works (e.g. [11,12]), but it is sufficient for our construction of the authentication scheme in Section 4. Note that in high dimensional case, the proposed functional encryption scheme is symmetric key system, and in 1D case it is public key system.

It might be possible to construct a functional encryption scheme satisfying the requirement in this paper based on MRQED, with different tradeoff in complexities since it also satisfies the polymorphic property. In the other direction, it could be also possible to construct an alternative solution to MRQED problem using techniques in this paper.

Several works [44, 45, 46, 47, 48, 49] 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.

2 Formulation

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

2.1 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 $[\mathcal{Z}]^d$. Let $\mathbf{R} = [a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_d, b_d] \subseteq [\mathcal{Z}]^d$ be a rectangular range. In this paper, we focus on aggregate count query function $F : F(\mathbf{D}, \mathbf{R}) \stackrel{\text{def}}{=} |\mathbf{D} \cap \mathbf{R}| \pmod p$, where p is a large prime.

2.2 Security Model

We formulize the authentication problem described in Section 1, as a special case of *Verifiable Computation* [6] particular for the function (i.e. query over a dataset) we are interested in this paper. Let us view a (generic)

[8] The original papers either do not provide a tight theoretical asymptotic bound, or do not relate the bound to generic parameters, including database size, domain size, dimension and security parameter.

query on a database as the function $F : \mathbb{D} \times \mathbb{R} \rightarrow \{0,1\}^*$, where $\mathbb{D}$ is the domain of databases, $\mathbb{R}$ is the domain of ranges, and the output of F is represented by a binary string. We define a remote computing protocol as follow:

Definition 1 ($\mathcal{RC}$) *A $\mathcal{RC}$ (*Remote Computing*) protocol for a function $F : \mathbb{D} \times \mathbb{R} \rightarrow \{0,1\}^*$, 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:*

1. *Given security parameter κ, Alice generates a key K: $K \leftarrow \mathsf{KGen}(1^\kappa)$.*
2. *Alice encodes database $\mathbf{D} \in \mathbb{D}$: $(\mathbf{D}_\mathtt{B}, \mathbf{D}_\mathtt{A}) \leftarrow \mathsf{DEnc}(\mathbf{D}, K)$, then sends $\mathbf{D}_\mathtt{B}$ to Bob and keeps $\mathbf{D}_\mathtt{A}$.*
3. *Alice selects a query $\mathbf{R} \in \mathbb{R}$.*
4. *Algorithm $\mathsf{Eval}(\mathbf{D}_\mathtt{B})$ on Bob's side, interacts with algorithm $\mathsf{Ext}(\mathbf{D}_\mathtt{A}, \mathbf{R}, K)$ on Alice'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 (partial) proof of result X. If $\zeta = \mathtt{reject}$, then Alice rejects. Otherwise, Alice accepts and believes that X is equal to $F(\mathbf{D}, \mathbf{R})$.*

In the setup phase, Alice executes Step (1) and (2). The query phase consists of multiple query sessions. In each query session, Alice and Bob execute Step (3) and (4).

We are interested in efficient $\mathcal{RC}$ protocol such that: (1) the size of K and $\mathbf{D}_\mathtt{A}$ are both in $O(\kappa)$; (2) communication complexity is $O(poly(d, \log|\mathbf{D}|))$; (3) the size of $\mathbf{D}_\mathtt{B}$ is $O(poly(d, |\mathbf{D}|))$ (this implies the complexity of DEnc is in $O(poly(d, |\mathbf{D}|))$). Similar as in [6,7], to make the outsourcing/delegation meaningful, we require that verification (i.e. the algorithm Ext) of Alice is more efficient than computing F.

We say a $\mathcal{RC}$ protocol is provable, 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 leant in a state variable.

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

- *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 $\langle \mathsf{Eval}(\mathbf{D}_\mathtt{B}), \mathsf{Ext}(\mathbf{D}_\mathtt{A}, r, K) \rangle = (\mathtt{accept},\ F(\mathbf{D},\ \mathbf{R}), \boldsymbol{\Psi})$ for some $\boldsymbol{\Psi}$, where $(\mathbf{D}_\mathtt{B}, \mathbf{D}_\mathtt{A}) \leftarrow \mathsf{DEnc}(\mathbf{D}, K)$.*
- *soundness: for any PPT (adaptive) adversary $\mathcal{A}$, the advantage $\mathsf{Adv}_{\mathcal{E},\mathcal{A}}(1^\kappa) \leq negl(\kappa)$ (asymptotically less or equal).*

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

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

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

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

The probability is taken over all random coins used by related algorithms, $negl(\cdot)$ is some negligible function, and $\mathsf{view}^{\mathcal{E}}_{\mathcal{A}}$ is a state variable[9] *describing all random coins chosen by $\mathcal{A}$ and all messages $\mathcal{A}$ can access during previous interactions with $\mathcal{E}$.*

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

We remark that the security model is also related to the formulation of $\mathcal{POR}$ (Proof of Retrievability) [50] and it is not surprising that our scheme (e.g. the extension for the summing query) could imply a (ρ, λ)-valid $\mathcal{POR}$ system, with some proper parameters ρ and λ.

2.3 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 (multiplicative) cyclic groups of order p.

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

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

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.

Assumption 2 (Generalized KEA [15, 20, 2, 51]) *Let $\mathcal{A}$ and $\bar{\mathcal{A}}$ be two algorithms. We define the* **GKEA***-advantage of $\mathcal{A}$ against $\bar{\mathcal{A}}$ as*

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

where the probability is taken over all random coins used. 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. $\mathsf{Adv}^{\mathbf{GKEA}}_{\mathcal{A},\bar{\mathcal{A}}}(\kappa) \leq \nu_2(\kappa)$, where m is polynomial in κ.*

Remark.

- Basically, the assumption says, if a PPT algorithm can output a valid pair $(U_1, U_2 = U_1^\beta)$, then the algorithm *essentially knows* some μ_i's, such that $U_1 = \prod_{j=1}^m (u_j)^{\mu_j}$ and $U_2 = \prod_{j=1}^m \left(u_j^\beta\right)^{\mu_j}$. The pair of adversary $\mathcal{A}$ and extractor $\bar{\mathcal{A}}$ is a way to formulize the notion of "essentially know". Note that the extractor $\bar{\mathcal{A}}$ can repeat the execution of $\mathcal{A}(W_m; r)$ *exactly*, since $\bar{\mathcal{A}}$ has access to the random coin r used by $\mathcal{A}$.
- **GKEA** is a natural extension of **KEA1** and **KEA3**, in the sense that **GKEA** $\Rightarrow$ **KEA3** $\Rightarrow$ **KEA1**. M. Abe and S. Fehr [21] 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 [51].

Furthermore, the (Decision) ℓ-wBDHI Assumption [1] is required for the IND-sID-CPA security of the underlying BBG HIBE scheme.

3 Functional Encryption Scheme

We construct a functional encryption [11, 12] scheme by exploiting the polymorphic property of BBG [1] HIBE scheme, following the overview given in Section 1.1.

3.1 Polymorphic Property of BBG HIBE Scheme

We observe that the BBG HIBE scheme [1] 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 $\mathsf{CT} = \widehat{\mathsf{CT}}$:

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

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

3.2 Define Identities based on Binary Interval Tree

An identity is a sequence of elements from $\mathbb{Z}_p^*$. To apply HIBE scheme, we intend to construct two mappings to associate identities to integers or integer intervals: (1) $\mathsf{ID}(\cdot)$ maps an integer $x \in [\mathcal{Z}]$ into an identity $\mathsf{ID}(x) \in \left(\mathbb{Z}_p^*\right)^\ell$. (2) $\mathsf{IdSet}(\cdot)$ maps an integer interval $[a, b] \subseteq [\mathcal{Z}]$ into a set of identities. The two mappings ID and IdSet satisfy the property: For any $x \in [a, b] \subseteq [\mathcal{Z}]$, there is a unique identity $\mathtt{id}$ in the set $\mathsf{IdSet}([a, b])$, such that identity $\mathtt{id}$ is a prefix of identity $\mathsf{ID}(x)$. If $x \notin [a, b]$, then there is no such identity $\mathtt{id}$ in $\mathsf{IdSet}([a, b])$. For each dimension $j \in [d]$, we will construct (distinct) such mappings ID_j and IdSet_j using a *binary interval tree* [13], and make these mappings public throughout the whole paper.

Binary Interval Tree. We construct a binary interval tree as below: First, we build a complete ordered binary with $\mathcal{Z} = 2^\ell$ leaf nodes. Then 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 associate 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 just $[1, \mathcal{Z}]$.

Constructions of Mappings ID *and* IdSet. Let $\mathcal{H} : \mathbb{Z}_{2^{\ell+1}} \times \mathbb{Z}_{2^{\ell+1}} \to \mathbb{Z}_p^*$ be a collision resistant hash function. Let $(\mathtt{v}_1, \mathtt{v}_2, \ldots, \mathtt{v}_m)$ be the unique simple path from the root node $\mathtt{v}_1$ to the node $\mathtt{v}_m$ in the binary interval tree. We associate to node $\mathtt{v}_m$ the identity $(\mathcal{H}(a_1, b_1), \mathcal{H}(a_2, b_2), \ldots, \mathcal{H}(a_m, b_m)) \in \left(\mathbb{Z}_p^*\right)^m$, 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 $\mathsf{ID}(x)$ 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 interval $[a, b]$, and then define $\mathsf{IdSet}([a, b])$ as the set $\{\mathtt{id}_j : \mathtt{id}_j \text{ is the identity associated to node } \mathtt{v}_j, 1 \le j \le n\}$. One can verify that the newly constructed mappings ID and IdSet satisfy the property mentioned in the beginning of Section 3.2. Furthermore, the size of set $\mathsf{IdSet}([a, b])$ is in $O(\ell)$.

3.3 Construction of Functional Encryption Scheme based on HIBE

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

$$f_\rho(\mathsf{Msg}) = \Omega^{\rho \cdot \mathsf{Msg}}$$

where $\Omega \in \widetilde{\mathbb{G}}$ is as in $f\mathsf{Setup}$ of Figure 1. The function f_ρ satisfies the following property:

$$f_\rho(\mathsf{Msg}) = f_1(\mathsf{Msg})^\rho \tag{4}$$

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*

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

(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*

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

(The proof is in Appendix C.)

We formulize the security requirement of our functional encryption sheme by modifying the IND-sID-CPA security game [1] for the sake of our usage of the functional encryption scheme in our final construction in Section 4. 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

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

Theorem 2 *(Informal) If the BBG HIBE scheme is IND-sID-CPA secure (as defined in [1]), 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* $\mathsf{Adv}_{\mathsf{FE},\mathcal{A}}^{\text{weak-IND-sID-CPA}}$*. (The formal version of this theorem and its proof appear in Appendix D).*

We remark that, unlike [11, 12], our functional encryption scheme is a symmetric key system, since the encryption algorithm $f\mathsf{Enc}$ requires the secret value $\boldsymbol{\tau}$, which (together with the random nonce ρ) is responsible to defeat collusion attack [13]. However, in the extreme case that dimension $d = 1$, the collusion attack is no longer possible, so we can save the secret value $\boldsymbol{\tau}$ and remove it from the secret key. As a result, our functional encryption scheme becomes a public key system in 1D case.

Fig. 1: Construction of Functional Encryption Scheme FE based on BBG [1] HIBE Scheme (Setup, KeyGen, Encrypt, Decrypt)

Functional Encryption Scheme FE

$f\mathsf{Setup}(1^\kappa, d, \mathcal{Z})$: `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 $\texttt{params} = (g, g_1, g_2, g_3, h_1, \ldots, h_\ell, \Omega = e(g_1, g_2))$ and private key $\texttt{master-key} = g_2^\alpha$, 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_j and IdSet_j, $j \in [d]$, be the mappings as in Section 3.2.
3. Choose d random elements $\tau_1, \ldots, \tau_d$ from $\mathbb{Z}_p^*$ and let $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_d)$.
4. Let $pk = (p, \mathbb{G}, \widetilde{\mathbb{G}}, e)$ and $sk = (pk, \texttt{params}, \texttt{master-key}, \boldsymbol{\tau})$. Make ID_j's and IdSet_j's public and output (pk, sk).

$f\mathsf{Enc}(\mathsf{Msg}, \boldsymbol{x}, sk)$: `message` Msg, d-`dimensional point` $\boldsymbol{x}$

1. Parse the d-dimensional point $\boldsymbol{x}$ as $(x_1, \ldots, x_d) \in [\mathcal{Z}]^d$; parse the private key sk as $(\mathsf{params}, \mathsf{master\text{-}key}, \boldsymbol{\tau})$, 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

$$\boldsymbol{c}_j \leftarrow \mathsf{Encrypt}(\texttt{params},\ \mathsf{ID}_j(x_j),\ \sigma_j;\ s_j). \tag{7}$$

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. Parse the d-dimensional 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 $j \in [d]$; parse the private key sk as $(\mathsf{params}, \mathsf{master\text{-}key}, \boldsymbol{\tau})$, 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 $\texttt{id} \in \mathsf{IdSet}_j(\mathbf{A}_j)$, generate the private key $d_{\texttt{id}}$, using algorithm KeyGen and taking $\mathsf{master\text{-}key}^{\rho\tau_j}$ as the master key.
   (b) Set $\delta_j \leftarrow \{d_{\texttt{id}} : \texttt{id} \in \mathsf{IdSet}_j(\mathbf{A}_j)\}$.
3. Output delegation decryption 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 range` $\mathbf{R}$, `delegated decryption key` $\boldsymbol{\delta}$

1. Parse the d-dimensional 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 $j \in [d]$; parse the ciphertext CT as $(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_d)$; parse 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 $\texttt{id}^* \in \mathsf{IdSet}_j(\mathbf{A}_j)$ such that $\texttt{id}^*$ 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 $d_{\texttt{id}^*} \in \delta_j = \{d_{\texttt{id}} : \texttt{id} \in \mathsf{IdSet}_j(\mathbf{A}_j)\}$ for identity $\texttt{id}^*$.
   (c) Generate the private key d_j for the identity $\mathsf{ID}_j(x_j)$ from private key $d_{\texttt{id}^*}$, using algorithm KeyGen.
   (d) Decrypt $\boldsymbol{c}_j$ using algorithm 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. Parse 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 $\boldsymbol{c}'_j$ and $\boldsymbol{c}_j$ are valid BBG ciphertexts for different plaintexts under the same identity.

Fig. 2: Construction of $\mathcal{RC}$ protocol $\mathcal{E} = (\mathsf{KGen}, \mathsf{DEnc}, \langle \mathsf{Ext}, \mathsf{Eval} \rangle)$ based on functional encryption scheme FE constructed in Figure 1, where $\langle \mathsf{Ext}, \mathsf{Eval} \rangle$ (namely ProVer) invokes $\langle \widetilde{\mathsf{Ext}}, \widetilde{\mathsf{Eval}} \rangle$ (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 $pk' = (p, \mathbb{G}, \widetilde{\mathbb{G}}, e, \Omega)$. Set $pk = (p, \mathbb{G}, \widetilde{\mathbb{G}}, e)$.
Note: e is a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \widetilde{\mathbb{G}}$, $\Omega \in \widetilde{\mathbb{G}}$, 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 function $f(\cdot)$, since Alice has $\Omega \in \widetilde{\mathbb{G}}$.
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 the functional encryption scheme 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 key $\mathcal{K}$ and $\mathbf{D}_\mathtt{A} = \left(N, d, \Delta = \prod_{i \in [N]} v_i^\beta\right)$ in local storage.

(Alice, Bob) $\mathsf{ProVer} = \langle \mathsf{Ext}(N, \Delta;\ \mathbf{R};\ \mathcal{K}), \mathsf{Eval}(\mathbf{D}, \mathbf{T}, \mathbf{C}, pk) \rangle$:
Precondition: The query range $\mathbf{R} \subset [\mathcal{Z}]^d$ is a rectangular range.

Step 1: Alice partitions the complement range $\mathbf{R}^\complement$ into $2d$ rectangular ranges $\{\mathbf{R}_\ell \subset [\mathcal{Z}]^d : \ell \in [1, 2d]\}$, and sets $\mathbf{R}_0 = \mathbf{R}$.
Step 2—Reduction: For $0 \le \ell \le 2d$, Alice and Bob invokes $\mathsf{CollRes}$ on range $\mathbf{R}_\ell$. Denote the output as $(\zeta_\ell, X_\ell, \Psi_2^{(\ell)})$.
Step 3: Alice sets $\zeta = \mathtt{accept}$, if the following equalities hold

$$\forall 0 \le \ell \le 2d, \zeta_\ell \stackrel{?}{=} \mathtt{accept}, \qquad \prod_{0 \le \ell \le 2d} \Psi_2^{(\ell)} \stackrel{?}{\equiv} \Delta; \tag{9}$$

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

(Alice, Bob) $\mathsf{CollRes} = \left\langle \widetilde{\mathsf{Ext}}\left(\mathbf{D}_\mathtt{A};\ \mathbf{R};\ \mathcal{K}\right), \widetilde{\mathsf{Eval}}\left(\mathbf{D}_\mathtt{B}\right) \right\rangle$: $\mathbf{D}_\mathtt{A} = (N, d, \Delta)$, $\mathbf{D}_\mathtt{B} = (\mathbf{D}, \mathbf{T}, \mathbf{C}, pk)$
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 *Help-Info* $\boldsymbol{\delta}$ 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 $(\Psi_1, \Psi_2, \Psi_3, \Psi_4)$ as follows

$$X \leftarrow |\mathbf{D} \cap \mathbf{R}|; \qquad (\Psi_1, \Psi_2, \Psi_3) \leftarrow \bigotimes_{\boldsymbol{x}_i \in \mathbf{D} \cap \mathbf{R}} \boldsymbol{t}_i; \qquad \Psi_4 \leftarrow \prod_{\boldsymbol{x}_i \in \mathbf{D} \cap \mathbf{R}} f\mathsf{Dec}(\mathsf{CT}_i, \boldsymbol{x}_i, \mathbf{R}, \boldsymbol{\delta}, pk). \tag{10}$$

Bob sends $(X, \Psi_1, \Psi_2, \Psi_3, \Psi_4)$ 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 $\Lambda \leftarrow \frac{\Psi_1}{\theta^X}$. Alice sets $\zeta = \mathtt{accept}$, if the following equalities hold

$$\Lambda^\beta \stackrel{?}{=} \Psi_2, \qquad \Lambda^\gamma \Psi_3^\rho \stackrel{?}{=} \Psi_4. \tag{11}$$

Otherwise sets $\zeta = \mathtt{reject}$. Alice outputs (ζ, X, Ψ_2).

4 The Main Construction

We construct a $\mathcal{RC}$ protocol $\mathcal{E} = (\mathsf{KGen}, \mathsf{DEnc}, \langle \mathsf{Ext}, \mathsf{Eval} \rangle)$ 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 points within the query range as the proof of the query result X. Then Alice verifies whether the returned proof $(\Psi_1, \Psi_2, \Psi_3, \Psi_4)$ is a valid Tag value for the returned query result X, with key $\mathcal{K}$ and without knowing the values of random coins v_j's and w_j's. Since the fourth component of Tag is dynamic and depends on the random nonce ρ, we separate it out from the other three components.
2. Intuitively, a more straightforward alternative construction of Tag is like this[10]

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

   where v^ρ is the output of $f\mathsf{Dec}$. We give up this alternative and introduce a new component in our construction for the sake of proof. A part of our proof is like this: Given the first two components of all tag values $\{(\theta^{y_i} v_i, v_i^\beta) : i \in [N]\}$, one can simulate Alice in our scheme. Then invoke a malicious Bob to interact with Alice to produce a forgery. For the alternative construction with Tag', to simulate the functional encryption scheme FE (particularly $f\mathsf{Enc}$), 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, since the additional term w_i is independent on the first two components of $\boldsymbol{t}_i$, the simulator can choose W_i freely, and generate $w_i \leftarrow f_1(W_i)$ and the ciphertext $\mathsf{CT}_i \leftarrow \mathsf{Mult}\left(f\mathsf{Enc}(W_i, \boldsymbol{x}_i, sk),\ \left(v_i^\beta\right)^{\gamma'},\ pk\right)$ where $\gamma' \in \mathbb{Z}_p^*$ is randomly chosen. Consequently, if $\boldsymbol{x}_i \in \mathbf{R}$, then by Lemma 1, $f\mathsf{Dec}(\mathsf{CT}_i, \boldsymbol{x}_i, \mathbf{R}, f\mathsf{KeyGen}(\mathbf{R}, \rho, sk), pk) = v_i^{\beta\gamma'} f_\rho(W_i) = v_i^\gamma w_i^\rho$ as desired (taking γ as $\beta\gamma'$). Thus the simulation of $f\mathsf{Enc}$ can be done.
3. To ensure completeness and prevent double counting or undercounting, 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. 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 $(\zeta_\ell = \mathtt{accept}, X_\ell = 0, \Psi_1 = \Psi_2 = \Psi_3 = \Psi_4 = 1 \in \widetilde{\mathbb{G}})$, without any help from Bob.
5. The $(2d+1)$ invocations of $\mathsf{CollRes}$ can be executed in parallel.
6. Like [6, 7], in our scheme, Alice's accept/reject decisions should be hidden from Bob. This is due to the limitation of our proof.

5 Security Analysis

5.1 Our main theorem

Theorem 3 (Main Theorem) *Suppose Assumption 1 and Assumption 2 hold, and BBG [1] 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 $\mathcal{PRC}$ w.r.t. function $F(\cdot,\cdot)$ as defined in Section 2.1, under Definition 2. Namely, $\mathcal{E}$ is* correct *and* sound *w.r.t. function F. (The proof is in appendix.)*

[10] Actually, the preliminary scheme in Section 1 is exactly like this.

5.2 Overview of Proof

To process a query, our scheme (particularly the algorithm ProVer) invokes $(2d+1)$ instances of protocol $\mathsf{CollRes}$. 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 double counting or undercounting. 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[11] from the same input as Type I adversary, additionally, it finds N integers[12] μ_i's, $1 \leq i \leq N$, 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 $\boldsymbol{x}_i \in \mathbf{D} \cap \mathbf{R}^{\complement}$.
- 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.

To prove the main Theorem 3, we introduce and prove Lemma 6, Theorem 7, and Theorem 8. Due to the space constraint, we put these lemmas/theorems together with their full proof in appendix: Lemma 6 is in Appendix E.2, Theorem 7 is in Appendix F, and Theorem 8 is in Appendix G. Additionally, the proof of Theorem 3 based on the above lemmas/theorems is in Appendix H. 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 $\mathsf{IND\text{-}sID\text{-}CPA}$ security of the underlying BBG HIBE scheme.
- 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.

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

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

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

5.2.1 Some remarks on our proof. Like many other proofs, the proof in this paper reduces a successful adversary to an algorithm which breaks a hard problem: Given an input of some hard problem, we simulate the behaviors of Alice in our scheme, and invoke the adversary (who plays the role of Bob) to interact with Alice to produce a forgery. Then we convert the forgery to a result of the hard problem. We argue that if the forgery is valid, then the result to the hard problem is correct. As a result, we conclude that such adversary does not exist based on the hard problem assumption.

In our proof, the simulated scheme is *identical* to the real scheme, from the view of adversary. That is, all messages that the adversary receives from Alice in the simulated scheme, are identically distributed as messages they will receive in the case of real scheme. However, in some cases, the simulator may not have full information of private key and thus may not be able to perform some verification steps. We just exploit the fact that the output of a successful adversary should pass all verifications with non-negligible probability[13], although we cannot tell whether the adversary succeeds or not in each single instance. This limitation of the simulation in our proof implies that we are in the same situation as [6, 7]: The accept/reject decision for each query processing should be hidden from Bob (or adversary).

5.3 The Preliminary Scheme is secure

In this subsection, we prove that the preliminary scheme described in Section 1.1 is secure, following the proof framework in Section 5.2. 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 $f(\cdot)$ is a random oracle. The preliminary scheme described in Section 1.1 is a $\mathcal{PRC}$ w.r.t. function $F(\cdot,\cdot)$ as defined in Section 2.1, 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: 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.* [20]. 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: For any $x_i \in \mathbf{D}$, $f(x_i) = u_i v_0$; for any $x \in [\mathcal{Z}]^d \setminus \mathbf{D}$, choose $z_x \in \mathbb{Z}_p^*$ at random and set $f(x) = u_1^{z_x}$. *Note that $f(x)^{\beta}$ still can be computed, although β is unknown.*
4. Invoke the preliminary scheme (Alice's part) with parameters β, θ and function f. *Note that tag $t_i = (\theta f(x_i), f(x_i)^{\beta}) = (u_i R_1, u_i^{\beta} R_2)$ still can be computed without knowing the values of $\theta, \beta, f(x_i)$.*
5. Invoke the adversary $\mathcal{B}$ (Bob's part) to interact with Alice. For any query $\mathbf{R}$ made by $\mathcal{B}$, generate *Help-Info* from $\{f(x)^{\beta}\}$ in this way: choose $\rho' \in \mathbb{Z}_p^*$ at random, and send $\{f(x)^{\beta\rho'} : x \in \mathbf{R}\}$. *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[14] 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

[13] This is just the definition of "successful adversary".
[14] Including the random coin.

$\{\mu_i : 1 \le i \le m+1\}$, 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

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

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: For each $x_i \in \mathbf{D}$, choose z_i at random from $\mathbb{Z}_p$ and set $f(x_i) = v^{z_i} \in G$.
3. Choose i^* from $[N]$ at random and redefine $f(x_{i^*})$: $f(x_{i^*}) = u$.
4. Invoke the preliminary scheme (Alice's part) with function f 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 *Help-Info* with random nonce $\rho = a$: the value $f(x_i)^a = (v^a)^{z_i}$ for $x_i \in \mathbf{R}$ can be computed, although a is unknown.
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).*

$$\varphi \leftarrow \frac{\Psi_3}{\prod_{1\le i\le N}^{i\neq i^*} f(x_i)^{a\mu_i}}$$

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 $\Psi_2 = \prod_{i=1}^{N} f(x_i)^{\beta\mu_i}$ and $i^* \in S_{\#}$, then the output $\varphi^{\mu_{i^*}^{-1}} = f(x_{i^*})^a = u^a$. 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 thre 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: For each $x_i \in \mathbf{D}$, choose y_i, z_i at random from $\mathbb{Z}_p$ and set $f(x_i) = (v^a)^{y_i} v^{z_i} \in G$; otherwise, set $f(x)$ to a random number in G.
3. Invoke the preliminary scheme (Alice's part) with function f 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 $(\overline{X}, \overline{\Psi}_1, \overline{\Psi}_2, \overline{\Psi}_3, \{\mu_i : x_i \in \mathbf{D} \cap \mathbf{R}^{\complement}\})$ be the output of adversary for the complement range $\mathbf{R}^{\complement}$.
5. If the adversary succeeds, we have

$$\prod_{i=1}^{N} f(x_i)^{\beta\mu_i} = \prod_{x_i\in\mathbf{D}\cap\mathbf{R}} f(x_i)^{\beta\mu_i} \prod_{x_i\in\mathbf{D}\cap\mathbf{R}^{\complement}} f(x_i)^{\beta\mu_i} = \Psi_2 \cdot \overline{\Psi}_2 = \Delta = \prod_{i=1}^{N} f(x_i)^{\beta}$$

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 $\Psi_2' = \prod_{x_i \in \mathbf{D} \cap \mathbf{R}} f(x_i)^{\beta \mu_i}$, 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 $\Psi_2 = \prod_{x_i \in \mathbf{D} \cap \mathbf{R}} f(x_i)^\beta = \Psi_2'$. 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 adversary against the preliminary scheme can output a wrong result and forged a valid proof. In other words, the preliminary scheme is sound. □

6 Performance and Extension

6.1 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 CollRes, the communication overhead is dominated by the size of *Help-Info* $\boldsymbol{\delta}$, 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 CollRes to process one query. Computation complexity on Bob's side is $O(dN \log \mathcal{Z})$ (bilinear map): (1) In CollRes, $O(d|\mathbf{D} \cap \mathbf{R}| \log \mathcal{Z})$ computation is required on query range $\mathbf{R}$ and the dominant computation step is Step B1 of 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 CollRes in Figure 2. The storage overhead on Bob's side, is $O(dN)$. The storage overhead on Alice's side, i.e. the key size, is $O(d)$.

6.2 Extension

We can extend our scheme to authenticate other types of aggregate range query, including summing, averaging, finding min/max/median. Furthermore, our scheme can be extended to authenticate d-dimensional usual (non-aggregate) range selection query with $O(d^2 \log^2 \mathcal{Z})$ bits communication overhead, improving known results that require $O(\log^{d-1} N)$ communication overhead, where N is the number of data points in the dataset. Due to the space constraint, we will report these results in the full version of this paper.

7 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. The proposed functional encryption scheme and the idea of implementing functional encryption by exploiting the polymorphic property of existing encrypton schemes may have independent interest.

References

1. Boneh, D., Boyen, X., Goh, E.J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: EUROCRYPT. (2005) 440–456
2. 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/`.
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, Deventer, The Netherlands, The Netherlands, Kluwer, B.V. (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: Proceedings of the 2002 ACM SIGMOD international conference on Management of data, New York, NY, USA, ACM (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. (2010) 465–482
7. Chung, K.M., Kalai, Y., Vadhan, S.P.: Improved Delegation of Computation Using Fully Homomorphic Encryption. In: CRYPTO. (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. Preparata, F.P., Shamos, M.I.: Computational geometry: an introduction. Springer-Verlag New York, Inc., New York, NY, USA (1985)
10. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully Homomorphic Encryption over the Integers. In: EUROCRYPT. (2010) 24–43
11. 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`.
12. O'Neill, A.: Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556 (2010) `http://eprint.iacr.org/`.
13. Shi, E., Bethencourt, J., Chan, T.H.H., Song, D., Perrig, A.: Multi-Dimensional Range Query over Encrypted Data. In: SP '07: Proceedings of the 2007 IEEE Symposium on Security and Privacy, Washington, DC, USA, IEEE Computer Society (2007) 350–364
14. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory **22**(6) (1976) 644 – 654
15. Damgård, I.: Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks. In: CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology. (1992) 445–456
16. Hada, S., Tanaka, T.: On the Existence of 3-Round Zero-Knowledge Protocols. In: CRYPTO '98: Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology. (1998) 408–423
17. Bellare, M., Palacio, A.: Towards Plaintext-Aware Public-Key Encryption Without Random Oracles. In: ASIACRYPT. (2004) 48–62
18. Krawczyk, H.: HMQV: A High-Performance Secure Diffie-Hellman Protocol. In: CRYPTO. (2005) 546–566
19. Dent, A.W.: The Cramer-Shoup Encryption Scheme Is Plaintext Aware in the Standard Model. In: EUROCRYPT. (2006) 289–307
20. Bellare, M., Palacio, A.: The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols. In: CRYPTO. (2004) 273–289
21. Abe, M., Fehr, S.: Perfect NIZK with adaptive soundness. In: TCC '07: Theory of Cryptography Conference. (2007)
22. Hacigümüs, H., Iyer, B.R., Mehrotra, S.: Efficient Execution of Aggregation Queries over Encrypted Relational Databases. In: DASFAA. (2004) 125–136
23. 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. (July 2006) 89–103
24. Ge, T., Zdonik, S.B.: Answering Aggregation Queries in a Secure System Model. In: VLDB. (2007) 519–530
25. 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
26. Devanbu, P., Gertz, M., Martel, C., Stubblebine, S.G.: Authentic data publication over the internet. J. Comput. Secur. **11**(3) (2003) 291–314
27. Pang, H., Jain, A., Ramamritham, K., Tan, K.L.: Verifying completeness of relational query results in data publishing. In: SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, New York, NY, USA, ACM (2005) 407–418
28. Pang, H., Tan, K.L.: Verifying Completeness of Relational Query Answers from Online Servers. ACM Trans. Inf. Syst. Secur. **11**(2) (2008) 1–50
29. Sion, R.: Query Execution Assurance for Outsourced Databases. In: VLDB. (2005) 601–612
30. Cheng, W., Tan, K.L.: Query assurance verification for outsourced multi-dimensional databases. J. Comput. Secur. **17**(1) (2009) 101–126
31. Li, F., Hadjieleftheriou, M., Kollios, G., Reyzin, L.: Dynamic authenticated index structures for outsourced databases. In: SIGMOD Conference. (2006) 121–132
32. Xie, M., Wang, H., Yin, J., Meng, X.: Integrity auditing of outsourced data. In: VLDB '07: Proceedings of the 33rd international conference on Very large data bases, VLDB Endowment (2007) 782–793
33. Atallah, M.J., Cho, Y., Kundu, A.: Efficient Data Authentication in an Environment of Untrusted Third-Party Distributors. In: ICDE '08: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering, Washington, DC, USA, IEEE Computer Society (2008) 696–704
34. Yang, Y., Papadias, D., Papadopoulos, S., Kalnis, P.: Authenticated join processing in outsourced databases. In: SIGMOD Conference. (2009) 5–18
35. 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
36. Pang, H., Zhang, J., Mouratidis, K.: Scalable Verification for Outsourced Dynamic Databases. In: Will appear in 35th International Conference on Very Large Data Bases (VLDB09)
37. Goodrich, M.T., Tamassia, R., Triandopoulos, N.: Super-Efficient Verification of Dynamic Outsourced Databases. In: CT-RSA. (2008) 407–424

38. Thompson, B., Yao, D., Haber, S., Horne, W.G., Sander, T.: Privacy-Preserving Computation and Verification of Aggregate Queries on Outsourced Databases. In: Privacy Enhancing Technologies Symposium (PETS). (Aug 2009)
39. Li, F., Hadjieleftheriou, M., Kollios, G., Reyzin, L.: Authenticated Index Structures for Aggregation Queries. *will appear in* ACM Transactions on Information and System Security (2010)
40. 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.
41. Chen, H., Ma, X., Hsu, W.W., Li, N., Wang, Q.: Access Control Friendly Query Verification for Outsourced Data Publishing. In: ESORICS. (2008) 177–191
42. Gentry, C.: Toward Basing Fully Homomorphic Encryption on Worst-Case Hardness. In: CRYPTO. (2010) 116–137
43. XU, J.: Authenticating aggregate range queries over dynamic multidimensional dataset. Cryptology ePrint Archive, Report 2010/244 (2010) `http://eprint.iacr.org/`.
44. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D.: Provable data possession at untrusted stores. In: CCS '07: Proceedings of the 14th ACM conference on Computer and communications security, New York, NY, USA, ACM (2007) 598–609
45. Chang, E.C., Xu, J.: Remote Integrity Check with Dishonest Storage Server. In: ESORICS '08: Proceedings of the 13th European Symposium on Research in Computer Security, Berlin, Heidelberg, Springer-Verlag (2008) 223–237
46. Shacham, H., Waters, B.: Compact Proofs of Retrievability. In: ASIACRYPT '08: International Conference on the Theory and Application of Cryptology and Information Security, Berlin, Heidelberg, Springer-Verlag (2008) 90–107
47. Bowers, K.D., Juels, A., Oprea, A.: HAIL: A High-Availability and Integrity Layer for Cloud Storage. Cryptology ePrint Archive, Report 2008/489 (2008)
48. Dodis, Y., Vadhan, S., Wichs, D.: Proofs of Retrievability via Hardness Amplification. In: TCC '09: Theory of Cryptography Conference. (2009) 109–127
49. 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
50. Juels, A., Kaliski, Jr., B.S.: Pors: proofs of retrievability for large files. In: CCS '07: Proceedings of the 14th ACM conference on Computer and communications security, New York, NY, USA, ACM (2007) 584–597
51. Groth, J.: Short Pairing-Based Non-interactive Zero-Knowledge Arguments. In: Advances in Cryptology - ASIACRYPT. (2010) 321–340

A HIBE

We restate the HIBE scheme proposed by Boneh *et al.* [1], 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 $k \leq \ell$, 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*

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

(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 [1] 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 $\mathsf{CT} = \widehat{\mathsf{CT}}$:

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

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 [1] 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 polymorphism 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). 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) $\mathsf{Msg} = -\sum_{j=1}^d s_j\tau_j \mod p$ and $\prod_{j=1}^d \sigma_j = \Omega^{-\sum_{j=1}^d s_j}$ we have,

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

In case $\boldsymbol{x} \notin \mathbf{R}$. Let $\mathbf{R} = \mathbf{A}_1 \times \mathbf{A}_2 \ldots \times \mathbf{A}_d$ as in Step 1 of $f\mathsf{Dec}$. If $\boldsymbol{x} \notin \mathbf{R}$, then for some dimension $j \in [d]$, $\boldsymbol{x}[j] \notin \mathbf{A}_j$, and $f\mathsf{Dec}$ will output $\perp$ (Step 2 of $f\mathsf{Dec}$ in 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 $\mathsf{Decrypt}(d_{\mathtt{id}}, (A, B, C))$ outputs M, then $\mathsf{Decrypt}(d_{\mathtt{id}}, (A\cdot\eta, B, C))$ 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), we have

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

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 *If there exists a* weak-IND-sID-CPA *adversary* $\mathcal{A}_{\mathsf{FE}}$*, which has non-negligible advantage* ϵ *against the scheme* FE *with one chosen delegation key query and* N_{aq} *chosen anonymous delegation key queries and* N_{enc} *chosen encryption queries and runs in time* t_{FE}*, then there exits an* IND-sID-CPA *adversary* $\mathcal{A}_{\mathsf{BBG}}$*, which has advantage* $\frac{\epsilon}{4d}$ *against the* BBG *HIBE scheme [1] 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. Using proof by contradiction, assume that there exists a PPT algorithm $\mathcal{A}_{\mathsf{FE}}$ which has non-negligible advantage ϵ against the FE scheme in the weak-IND-sID-CPA game with one delegation key query and polynomial number of anonymous delegation key queries.

The proof idea. 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 ouput 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 $1/2$.

Now let us construct the BBG adversary $\mathcal{A}_{\mathsf{BBG}}$. $\mathcal{A}_{\mathsf{BBG}}$ 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 $\mathcal{A}_{\mathsf{BBG}}$ 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 : $\mathcal{A}_{\mathsf{BBG}}$ does nothing.

BBG Challenge : $\mathcal{A}_{\mathsf{BBG}}$ chooses m_0, m_1 at random from the plaintext space $\widetilde{\mathbb{G}}$, and sends (m_0, m_1) to $\mathcal{C}_{\mathsf{BBG}}$. The challenger $\mathcal{C}_{\mathsf{BBG}}$ picks a random bit $b \in \{0,1\}$ and sends the challenge ciphertext $\mathsf{CT} = \mathsf{Encrypt}(\mathsf{params}, \mathsf{id}^*, m_b)$ 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 $(p, \mathbb{G}, \widetilde{\mathbb{G}}, e)$ be a part of params, where p is a prime, both $\mathbb{G}$ and $\widetilde{\mathbb{G}}$ are cyclic multiplicative group of order p, and $e : \mathbb{G} \times \mathbb{G} \to \widetilde{\mathbb{G}}$ is a bilinear map. Let $pk = (p, \mathbb{G}, \widetilde{\mathbb{G}}, e)$ and $sk = (pk, \mathsf{params}, \text{master-key}, \boldsymbol{\tau})$. $\mathcal{A}_{\mathsf{BBG}}$ sends pk to $\mathcal{A}_{\mathsf{FE}}$. *Note:* $\mathcal{A}_{\mathsf{BBG}}$ *does not know* master-key

FE Challenge : $\mathcal{A}_{\mathsf{BBG}}$ chooses a random bit $a \in \{0,1\}$ and a 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. execluding 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 $\mathsf{CT} = (m_b\Omega^s, B, C)$:

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

6. Define $\mathsf{Msg}_a = -\sum_{j\in[d]} s_j \cdot \tau_j$, where unknown s_ξ satisfies $\Omega^{s_\xi} = \Omega^s m_b \cdot m_0^{-1}$. Although the value Msg_a is unknown since s_ξ is unknown, $\mathcal{A}_{\mathsf{BBG}}$ can still compute $f_{\rho=1}(\mathsf{Msg}_a)$:

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

7. Set the challenge ciphertext to $\mathsf{CT}_{\mathsf{FE}} = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_d)$, and send $(\mathsf{CT}_{\mathsf{FE}}, f_{\rho=1}(\mathsf{Msg}_0), f_{\rho=1}(\mathsf{Msg}_1))$ to $\mathcal{A}_{\mathsf{FE}}$.

FE Learning Phase :

1. $\mathcal{A}_{\mathsf{FE}}$ issues a delegation key query $(\mathbf{R}, \rho)$, where $\boldsymbol{x}^* \notin \mathbf{R}$: If $\exists \boldsymbol{x} \in \mathbf{R}$, such that $\mathsf{ID}_\xi(\boldsymbol{x}[\xi]) = \mathsf{ID}_\xi(x_\xi)$, then abort and output a random bit $b' \in \{0,1\}$ (Denote this event as $\mathbf{E}_1$). Otherwise, simulate the procedure of $f\mathsf{KeyGen}$:
   (a) Parse the d-dimensional 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 $j \in [d]$. The private key is $sk = (\mathsf{params}, \mathsf{master\text{-}key}, \boldsymbol{\tau} = (\tau_1, \ldots, \tau_d))$, where $\mathcal{A}_{\mathsf{BBG}}$ has only params and $\boldsymbol{\tau}$, and does not know $\mathsf{master\text{-}key}$ (which is kept securely by the BBG challenger $\mathcal{C}$).
   (b) For each $j \in [d]$, generate a set δ_j in this way:
   – For each identity $\mathtt{id} \in \mathsf{IdSet}_j(\mathbf{A}_j)$, issue a private key query with identity $\mathtt{id}$ to BBG challenger $\mathcal{C}$ and get reply $d_{\mathtt{id}}$.
   – For each identity $\mathtt{id} \in \mathsf{IdSet}_j(\mathbf{A}_j)$, parse the key $d_{\mathtt{id}}$ as $(K_0, K_1, \Upsilon_k, \ldots, \Upsilon_\ell)$ and set $d'_{\mathtt{id}} = (K_0^{\rho\tau_j}, K_1^{\rho\tau_j}, \Upsilon_k^{\rho\tau_j}, \ldots, \Upsilon_\ell^{\rho\tau_j})$.
   – Set $\delta_j \leftarrow \{d'_{\mathtt{id}} : \mathtt{id} \in \mathsf{IdSet}_j(\mathbf{A}_j)\}$.
   (c) Send $\boldsymbol{\delta} = (\delta_1, \delta_2, \ldots, \delta_d)$ to $\mathcal{A}_{\mathsf{FE}}$ as the delegation decryption 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 $sk' = (\mathsf{params}, Z, \boldsymbol{\tau})$ (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 $\mathcal{C}$. (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.

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

BBG Guess : If $a = a'$, output $b' = 0$. Otherwise, output $b' = 1$.

The constructed BBG adversary $\mathcal{A}_{\mathsf{BBG}}$ made $O(d\ell)$ private key query and zero decryption query to the BBG challenger $\mathcal{C}$. If $b = 0$, 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 $b = 1$, in the FE scheme simulated by $\mathcal{A}_{\mathsf{BBG}}$, the challenger ciphertext $\mathsf{CT}_{\mathsf{FE}}$ is *independent* on $\mathsf{Msg}_0, \mathsf{Msg}_1$. 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

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

In case of $\neg\mathbf{E}_1$: Suppose event $\mathbf{E}_1$ does not occur. Since $\mathsf{Pr}[a \neq a' \wedge b' = 0] = \mathsf{Pr}[a = a' \wedge b' = 1] = 0$, we have

$$\begin{aligned}
\mathsf{Pr}[b' = 0 | b = 0] &= \mathsf{Pr}[a = a' \wedge b' = 0 | b = 0] + \mathsf{Pr}[a \neq a' \wedge b' = 0 | b = 0] \\
&= \mathsf{Pr}[a = a' \wedge b' = 0 | b = 0] + 0 \\
&= \mathsf{Pr}[a = a' | b = 0]\mathsf{Pr}[b' = 0 | b = 0, a = a'] \\
&= \mathsf{Pr}[a = a' | b = 0]
\end{aligned}$$

Similarly, we can show that $\mathsf{Pr}[b' = 1 | b = 1] = \mathsf{Pr}[a \neq a' | b = 1]$.

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

$$\begin{aligned}\mathsf{Pr}[b=b'] &= \mathsf{Pr}[b=b'=0]+\mathsf{Pr}[b=b'=1]\\ &= \mathsf{Pr}[b=0]\mathsf{Pr}[b'=0|b=0]+\mathsf{Pr}[b=1]\mathsf{Pr}[b'=1|b=1]\\ &= \mathsf{Pr}[b=0]\mathsf{Pr}[a=a'|b=0]+\mathsf{Pr}[b=1]\mathsf{Pr}[a\neq a'|b=1]\\ &= \frac{1}{2}\times\left(\frac{1}{2}+\epsilon\right)+\frac{1}{2}\times\frac{1}{2}\\ &= \frac{1}{2}+\frac{1}{4}\epsilon\end{aligned}$$

As a resulst, by combining two cases, we obtain the advantage of $\mathcal{A}_{\mathsf{BBG}}$ against BBG scheme in the IND-sID-CPA game:

$$\mathsf{Adv}_{\mathsf{BBG}}^{\mathcal{A}_{\mathsf{BBG}}}=\mathsf{Pr}[\mathbf{E}_1]\times\frac{1}{2}+\mathsf{Pr}[\neg\mathbf{E}_1]\times\left(\frac{1}{2}+\frac{1}{4}\epsilon\right)=\frac{1}{2}+\frac{\epsilon}{4}\mathsf{Pr}[\neg\mathbf{E}_1]\geq\frac{1}{2}+\frac{\epsilon}{4d}$$

The adversary $\mathcal{A}_{\mathsf{BBG}}$ wins the game with probability larger than $1/2$ 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 Encrypt, or BBG key generation KeyGen, and N_{aq} (N_{enc}, respectly) is the number of anonymous delegation key queries (encryption key queries, respectly) 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 [20]. 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*

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

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 $W_m=\{(u_i,u_i^\beta)\in\widetilde{\mathbb{G}}^2:1\leq i\leq m\}$ 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* $\mathsf{Adv}^{\mathbf{Lem\ 6}}_{\mathcal{A},\bar{\mathcal{A}}}(1^\kappa) \leq \nu_2(\kappa)$, *where the advantage* $\mathsf{Adv}^{\mathbf{Lem\ 6}}_{\mathcal{A},\bar{\mathcal{A}}}$ *of $\mathcal{A}$ against $\bar{\mathcal{A}}$ w.r.t. scheme $\widetilde{\mathcal{E}} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{CollRes})$ is defined as*

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

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 algorithm $\mathcal{B}$ based on $\mathcal{A}$.

Adversary $\mathcal{B}$ w.r.t. 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 $\mathsf{Exp}^{\widetilde{\mathcal{E}}}_{\mathcal{A}}$.
   (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 and 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 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 $\mathbf{D}_{\mathtt{B}} = \{\mathbf{D}, \mathbf{T} = \{\boldsymbol{t}_i : i \in [N]\}, \mathbf{C} = \{\mathsf{CT}_i : i \in [N]\}, pk\}$ to adversary $\mathcal{A}$.
   (d) Simulate $\mathsf{CollRes}$: 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. $\mathcal{B}$ has the private key sk, and can generate the *Help-Info* in the same way as in $\mathsf{CollRes}$ in Figure 2.
2. $\mathcal{B}$ has no knowledge of β or θ, and consequently cannot perform verification as in $\mathsf{CollRes}$.
3. The view of adversary $\mathcal{A}$ after interacting with simulated scheme is identically distributed with the the view $\mathsf{view}^{\widetilde{\mathcal{E}}}_{\mathcal{A}}$ of $\mathcal{A}$ after interacting with the real scheme in the experiment $\mathsf{Exp}^{\widetilde{\mathcal{E}}}_{\mathcal{A}}$.
   (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, the public encoding $\mathbf{D}_{\mathtt{B}} = \{\mathbf{D}, \mathbf{T}, \mathbf{C}\}$ that $\mathcal{A}$ received is identically distributed as in real experiment.
   (c) $\mathsf{CollRes}$
      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 $\mathsf{Adv}_{\mathcal{B},\bar{\mathcal{B}}}^{\mathbf{Lem\ 5}} \leq \nu_2$. 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[15] 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 $\mathsf{Adv}_{\mathcal{B},\bar{\mathcal{B}}}^{\mathbf{Lem\ 5}} \leq \nu_2$. 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

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

□

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 $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\mathbf{Lem\ 6}}$ and*

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

$\mathsf{Adv}^{\mathbf{Thm\ 7}}_{\mathcal{A},\bar{\mathcal{A}}}$ *are negligible, where the advantage* $\mathsf{Adv}^{\mathbf{Thm\ 7}}_{\mathcal{A},\bar{\mathcal{A}}}$ *of* $\mathcal{A}$ *against* $\bar{\mathcal{A}}$ *w.r.t. scheme* $\widetilde{\mathcal{E}} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{CollRes})$ *is defined as*

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

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).

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

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

where ν_1 is as in Assumption 1 and $\mathsf{Adv}^{\mathsf{weak\text{-}IND\text{-}sID\text{-}CPA}}_{\mathsf{FE},\mathcal{B}}$ is defined in Section 3 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 $\mathsf{Exp}^{\widetilde{\mathcal{E}}}_{\mathcal{A}}$ by invoking the adversary $\mathcal{A}$, where $\mathcal{B}$ taks 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 $\mathsf{Exp}^{\widetilde{\mathcal{E}}}_{\mathcal{A}}$. $\mathcal{B}$ 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 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 $f(\cdot)$.*
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 valus 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, otuput 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.

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

Therefore,

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{IND\text{-}sID\text{-}CPA}}_{\mathsf{FE},\mathcal{B}} &= \left|\mathsf{Pr}\left[b = b', \mathbf{E}_2\right] - \frac{1}{2}\mathsf{Pr}\left[\mathbf{E}_2\right]\right| \qquad (23) \\
&\geq \left|\ \left|\frac{1}{2}\mathsf{Pr}\left[b = b', \mathbf{E}_2 \mid b = 0\right] - \frac{1}{4}\mathsf{Pr}\left[\mathbf{E}_2 \mid b = 0\right]\right| - \left|\frac{1}{2}\mathsf{Pr}\left[b = b', \mathbf{E}_2 \mid b = 1\right] - \frac{1}{4}\mathsf{Pr}\left[\mathbf{E}_2 \mid b = 1\right]\right| \ \right| \qquad (24)
\end{aligned}$$

$\mathsf{Adv}^{\mathsf{weak\text{-}IND\text{-}sID\text{-}CPA}}_{\mathsf{FE},\mathcal{B}}$ 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 γ 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 $\boldsymbol{t}_i = (v_i, v_i^\beta, f(W_i)) \in \widetilde{\mathbb{G}}^3$;
   iii. generate a ciphertext CT_i as in Figure 2.
   (c) For i^*:
   i. generate a tag $\boldsymbol{t}_{i^*} = (v_{i^*}, v_{i^*}^\beta, f(W_{i^*}))$ 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 do not perform the verifications in step A2 of $\mathsf{CollRes}$.
   Note: Since γ is unknown, some verifications can not be done.
3. Invoke the adversary $\mathcal{A}$ and simulate the experiment $\mathsf{Exp}^{\widetilde{\mathcal{E}}}_{\mathcal{A}}$ 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 ρ 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 ϕ 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 $\mathsf{Exp}^{\widetilde{\mathcal{E}}}_{\mathcal{A}}$ simulated by $\mathcal{D}$ as $\mathsf{Exp}_{\mathcal{D}}$; denote the experiment $\mathsf{Exp}^{\widetilde{\mathcal{E}}}_{\mathcal{A}}$ 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 computationlly 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 thoese 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 $\mathsf{Exp}_{\mathcal{D}}$, $\boldsymbol{t}_{i^*} = (v_{i^*},\ v_{i^*}^\beta,\ f(W_{i^*}))$ 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[16] 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[17], 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

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

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

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

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

□

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 [1] 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 $\mathsf{Adv}^{\mathbf{Lem\ 6}}_{\mathcal{A},\bar{\mathcal{A}}}$, $\mathsf{Adv}^{\mathbf{Thm\ 7}}_{\mathcal{A},\bar{\mathcal{A}}}$, and $\mathsf{Adv}^{\mathbf{Thm\ 8}}_{\mathcal{A},\bar{\mathcal{A}}}$ are negligible, where the advantage $\mathsf{Adv}^{\mathbf{Thm\ 8}}_{\mathcal{A},\bar{\mathcal{A}}}$ of $\mathcal{A}$ against $\bar{\mathcal{A}}$ w.r.t. scheme $\mathcal{E} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{ProVer})$ is defined as*

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

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 8).

[16] 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 $\mathcal{A}$.

[17] 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 $\mathsf{Pr}[\mathbf{E}_2|b = 1]$

Idea of proof. For any PPT algorithm $\mathcal{A}$, applying Theorem 7, let $\bar{\mathcal{A}}$ be the PPT algorithm, such that $\mathsf{Adv}^{\mathbf{Lem\ 6}}_{\mathcal{A},\bar{\mathcal{A}}} \leq \epsilon_5$ and $\mathsf{Adv}^{\mathbf{Thm\ 7}}_{\mathcal{A},\bar{\mathcal{A}}} \leq \epsilon_6$ for some negligible functions $\epsilon_5(\cdot)$ and $\epsilon_6(\cdot)$. Using proof of contradiction, assume that $\mathsf{Adv}^{\mathbf{Thm\ 8}}_{\mathcal{A},\bar{\mathcal{A}}} \geq \epsilon_7$ 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 $\zeta = \mathtt{accept} \ \wedge \ \Delta \neq \prod_{i\in[N]} \left(v_i^{\beta}\right)^{\mu_i}$, and with $\mathbf{E}_2$ the event that $\zeta = \mathtt{accept} \ \wedge \ \Delta = \prod_{i\in[N]} \left(v_i^{\beta}\right)^{\mu_i} \ \wedge \ \exists j \in [N], \mu_j \neq 1$. We can split the probability $\mathsf{Adv}^{\mathbf{Thm\ 8}}_{\mathcal{A},\bar{\mathcal{A}}}$ into two parts,

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

Part I: $\mathsf{Pr}[\mathbf{E}_1] \leq (2d+1)\left(\mathsf{Adv}^{\mathbf{Lem\ 6}}_{\mathcal{A},\bar{\mathcal{A}}} + \mathsf{Adv}^{\mathbf{Thm\ 7}}_{\mathcal{A},\bar{\mathcal{A}}}\right)$. Suppose that: (1) The challenging query range is $\mathbf{R}$. (2) Alice partitions $\mathbf{R}^{\complement}$ into $2d$ rectangualr ranges $\mathbf{R}_1, \ldots, \mathbf{R}_{2d}$ and sets $\mathbf{R}_0 = \mathbf{R}$. (3) For $0 \leq \ell \leq 2d$, denote with $(\zeta_\ell, X_\ell, \Psi_2^{(\ell)})$ the reply returned by adversary $\mathcal{A}$ in the excution of $\mathsf{CollRes}$ on range $\mathbf{R}_\ell$. (4) Denote with (ζ, X, Δ) the output of Alice in the execution of ProVer. (5) Recall that Alice keeps the value $\Delta = \prod_{i\in[N]} v_i^{\beta}$.

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

$$\left(\bigwedge_{\ell\in[0,2d]} \zeta_\ell = \mathtt{accept}\right) \ \wedge \ \Delta = \prod_{\ell\in[0,2d]} \Psi_2^{(\ell)} \ \Leftrightarrow \ \zeta = \mathtt{accept} \tag{27}$$

The conjunctions of equation (27) (denoted as statement A) and statements $A_\ell : \zeta_\ell = \mathtt{accept} \Rightarrow \Psi_2^{(\ell)} = \prod_{i\in[N]} v_i^{\beta\mu_i}$, $0 \leq \ell \leq 2d$, and statements $B_\ell : \zeta_\ell = \mathtt{accept} \wedge \Psi_2^{(\ell)} = \prod_{i\in[N]} v_i^{\beta\mu_i} \Rightarrow \forall \boldsymbol{x}_i \in \mathbf{D} \cap \mathbf{R}^{\complement}, \mu_i = 0$, $0 \leq \ell \leq 2d$, directly imply that

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

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

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

Therefore,

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

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

$\mathcal{B}$: **DLP** Adversary

1. The input is $(v, v^a) \in \widetilde{\mathbb{G}}^2$. The goal is to find $a \in \mathbb{Z}_p$.
2. Define function $f_2 : [N] \to \widetilde{\mathbb{G}}$ in this way: For each $i \in [N]$, choose $y_i, z_i \in \mathbb{Z}_p$ at random and set $f(i) = (v^a)^{y_i} v^{z_i} \in \widetilde{\mathbb{G}}$.
3. Invoke scheme $\mathcal{E} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{ProVer})$ with f_2 defined as above. *Note: $\mathcal{B}$ has full information of private key.*
4. Simulate the experiment $\mathsf{Exp}^{\mathcal{E}}_{\mathcal{A}}$, by invoking the adversary $\mathcal{A}$ (playing the role Bob) to interact with Alice in $\mathcal{E}$. Then invoke $\bar{\mathcal{A}}(\mathsf{view}^{\mathcal{E}}_{\mathcal{A}})$ to obtain $\{\mu_i : i \in [N]\}$.
5. With probability equal to $\mathsf{Pr}[\mathbf{E}_2]$, it holds that $\zeta = \mathtt{accept} \bigwedge \Delta = \prod_{i\in[N]} \left(v_i^{\beta}\right)^{\mu_i} \bigwedge \exists j \in [N], \mu_j \neq 1$.
6. 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 $v_j = f(j)^{\gamma^{-1}} = v^{(z_j+a)\gamma^{-1}}$. 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

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

□

H Proof of Main Theorem 3

Theorem 3 (Main Theorem) *Suppose Assumption 1 and Assumption 2 hold, and BBG [1] 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 $\mathcal{PRC}$ w.r.t. function $F(\cdot,\cdot)$ as defined in Section 2.1, 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 [1] 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.

$\mathcal{B}$: Adversary 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 unkonwn $\theta \in \widetilde{\mathbb{G}}$.
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 $S_{m+1} = \{(v_i\theta^{y_i}, v_i^{\beta}, y_i)\}_{i=0}^{m}$ 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 $\mathsf{CollRes}$.
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 2.1.
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 $\mathcal{A}$.

Claim H.02 *Suppose Assumption 1 and Assumption 2 hold, and BBG [1] 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

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

where $\zeta \in \{\texttt{accept}, \texttt{reject}\}$ 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 [1] HIBE scheme is IND-sID-CPA secure, by applying Lemma 6, Theorem 7 and Theorem 8, the following implications hold with o.h.p.,

$$\zeta = \texttt{accept} \ \Rightarrow\ \left(\Delta = \prod_{i\in[N]} \left(v_i^\beta\right)^{\mu_i} \ \wedge\ \forall i \in [N], \mu_i = 1\right);$$
$$\zeta = \texttt{accept} \ \Rightarrow\ \bar{\Psi}_2 = \prod_{\boldsymbol{x}_i \in \mathbf{D}\cap\mathbf{R}} \left(v_i^\beta\right)^{\mu_i}.$$

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. □