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 relies on Generalized Knowledge of Exponent Assumption proposed by Wu and Stinson [1], and exploits a special property of BBG [2] HIBE scheme. 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, Provable Remote Computing

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 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) 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].

Researches in secure outsourced database focus on two major aspects: privacy [4,6,7,8] (i.e. protect the data confidentiality against both the service provider and any third party), and integrity [3,9,10,11,5,12,13,14,15,16,17,18,19,20,21,22,23] (i.e. authenticate the soundness and completeness of query results returned by the service provider). In the latter aspect, a lot of works are done for "identity query" [13], 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. [12,22,23]) are devoted to the authentication of aggregate query.

1.1 Our results

We propose a scheme, which we call MAIA (Multidimensional Aggregate query Integrity Authentication), 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] [24], then $\mathcal{Z} = N$ and $O(d^2 \log^2 \mathcal{Z})$ is sublinear in N and polynomial in d. To the best of our knowledge, this is the first solution with communication overhead sublinear in the number of points in the dataset and polynomial in dimension, without using fully homomorphic encryption scheme [25, 26].

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 3) under certain assumptions. (2) We brief our strategy and key technique used in the main scheme (Section 4) to reduce the communication overhead and computation overhead. (3) We brief 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 multiplicative cyclic 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 $\pi^* = \prod_{x \in \mathbf{D}} t_{x,2}$. Next, Alice sends dataset $\mathbf{D}$ and tag values $\mathbf{T} = \{t_x : x \in \mathbf{D}\}$ to Bob and deletes everything except π^* 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?

$$\pi^* \stackrel{?}{=} \Psi_2 \cdot \overline{\Psi}_2. \tag{1}$$

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

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

- 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 $\prod_{x \in \mathbf{D}} t_{x,2}^{\mu_x} = \pi^* = \prod_{x \in \mathbf{D}} t_{x,2}$ 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 3 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 $\boldsymbol{\delta}$ (called *Help-Info*) 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 [2], 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

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

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 $|S'| = O(\log \mathrm{TreeSize}) = O(\log \mathcal{Z})$; (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. The security of this method can be reduced to the $\mathsf{IND\text{-}sID\text{-}CCA}$ security of the underlying HIBE scheme.

We call the above method as an *expansion scheme*: For any random nonce ρ, Alice can produce a *Help-Info* $\boldsymbol{\delta}$ of small size. From a fixed public information $\{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_N\}$ and the *Help-Info* $\boldsymbol{\delta}$, Bob can produce $\{f(x)^{\rho} : x \in \mathbf{D} \cap \mathbf{R}\}$, which could be much larger than $\boldsymbol{\delta}$.

[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 [27] and Generalized Knowledge of Exponent (**GKEA**) Assumption 2 [1], where both assumptions are over the target group of a bilinear map. The **GKEA** assumption is an extension of **KEA1** [28,29,30,31,32] and **KEA3** [33], and proposed by Wu and Stinson [1]. Roughly, **GKEA** assumption can be described as below:

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

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 [34].

Additionally, the (Decision) ℓ-wBDHI Assumption [2] is required for the IND-sID-CCA security of the underlying BBG HIBE scheme.

Contribution

1. We propose an *expansion scheme* in Section 3. In the setup of this expansion scheme, Alice generates a public parameter $\mathbf{C} = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_N\}$ for a dataset $\mathbf{D} = \{x_i \in [\mathcal{Z}]^d : i \in [N]\}$ w.r.t. a function f. After the setup, in each query session, for any range $\mathbf{R} \subset [\mathcal{Z}]^d$ and any random nonce ρ, Alice can generate a value $\boldsymbol{\delta}$, called as *Help-Info*. From the *Help-Info* $\boldsymbol{\delta}$ and the public parameter $\mathbf{C}$, Bob can compute $\Phi = \{f(x)^{\rho} : x \in \mathbf{D} \cap \mathbf{R}\}$ but cannot generate $f(x)^{\rho}$ for point $x \notin \mathbf{R}$. In this way, Alice can deliver information Φ of size $O(N)$ to Bob by sending only the *Help-Info* $\boldsymbol{\delta}$ of size $O(d \log^2 \mathcal{Z})$.
2. We incorporate the expansion scheme into the preliminary scheme in Section 4. The resulting scheme called as MAIA, 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 $O(dN)$.
3. We prove that MAIA is secure (Theorem 2) under reasonable assumptions (Assumption 1, Assumption 2 and ℓ-wBDHI Assumption [2]). 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. In fact, we start our work with summing. For the simplicity of presentation, we focus on counting in this paper.

1.2 Related work

There are roughly three categories of approaches for outsourced database authentication in the literatures [3,9,10,11,5,12,13,14,15,16,17,18,19,20,21]. (1) (Homomorphic and/or aggregatable) Cryptographic primitives, like collision-resistant hash, digital signature, commitment [5,35,22]. (2) Geometry partition and authenticated data structure [9,14,17,19,15,23]. For example, Merkle Hash Tree (typically for 1D case) and variants, KD-tree with chained signature [12], and R-Tree with chained signature [14]. (3) Inserting and auditing fake tuples [16].

To the best of our knowledge, the existing few works (e.g. [12,22,23]) on authentication of aggregate query either only deal with 1D case, or have communication overhead[6] 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.* [9], Chen *et al.* [36]), 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.

Very recently, Gennaro *et al.* [37] and Chung *et al.* [38] proposed methods to authenticate *any* outsourced (or delegated) function, based on fully homomorphic encryption [25,26,39]. Without considering the efficiency (particularly ciphertext expansion) of fully homomorphic encryption scheme, Gennaro *et al.* [37] has communication cost which is sublinear w.r.t. the number of points in a dataset and polynomial in dimension, to authenticate

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

[6] 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.

aggregate range query. Our work is different in at least these aspects: (1) They authenticate a much more generic class of functions, while our techniques only deal with some aggregate range query (like counting, summing, finding of maximum or minimum or median) and non-aggregate range selection query. (2) Besides integrity authentication, Gennaro *et al.* [37] even provides privacy protection of the outsourced data against the worker (corresponding to Bob in our formulation). (3) Gennaro *et al.* [37] and Chung *et al.* [38] leverage on fully homomorphic encryption scheme [25]. (4) To deal with aggregate range query over outsourced database, both Gennaro *et al.* [37] and Chung *et al.* [38] have to treat the whole database as a single big chunk of data, so the completeness can be easily guaranteed at the cost of efficiency. We adapt a different approach: We bind each data point with an independent random number of special structure, and force Bob to process the dataset along with these random numbers in an inseparable manner. Then we can verify the consistency between the returned query result and the associated randomness. In this way, we can achieve better[7] complexity than the two generic methods. However, our approach requires more serious attention to deal with the completeness issue. As Gennaro *et al.* [37] pointed out, it is meaningful to design more efficient authentication scheme[8] without using fully homomorphic encryption scheme, even at the cost of sacrificing privacy of outsourced data.

Shi *et al.* [40] proposed MRQED (Multi-Dimensional Range Query over Encrypted Data), a public key encryption scheme supporting multidimensional range queries over ciphertexts. Both MRQED [40] and MAIA deal with multidimensional range selection and have to prevent collusion attack across different queries. But they are essentially different in at least these aspects: (1) MRQED dealt with privacy, and MAIA deals with integrity. (2) In MAIA, there is an aggregate operation after multidimensional range selection, and the verification of aggregated value is an additional requirement and not easy to handle when communication cost is concerned. (3) Besides collusion attack, MAIA also faces other challenges, like completeness issue, which have no counterparts in MRQED. It may be possible to construct an expansion scheme based on MRQED with different tradeoff in complexities since it also has the polymorphic property. In the other direction, it is also possible to construct an alternative solution to MRQED problem using techniques in this paper.

Several works $[41, 42, 43, 44, 45, 46]$ 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 $F : F(\mathbf{D}, \mathbf{R}) \overset{\text{def}}{=} |\mathbf{D} \cap \mathbf{R}| \pmod p$.

2.2 Security Model

We formulize the authentication problem described in Section 1. Let us view a (generic) query on a database as the function $F : \mathbb{D} \times \mathbb{R} \to \{0, 1\}^*$, 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 $F : \mathbb{D} \times \mathbb{R} \to \{0, 1\}^*$, between Alice and Bob, consists of a setup phase and a query phase. The setup phase consists of a key generating algorithm* KGen *and data encoding algorithm* DEnc*; the query phase consists of a pair of interactive algorithms, namely the evaluator* Eval *and the extractor* Ext*. These four algorithms* $(\mathsf{KGen}, \mathsf{DEnc}, \langle \mathsf{Eval}, \mathsf{Ext} \rangle)$ *run in the following way:*

[7] Ciphertext expansion is not the only reason of communication overhead and storage overhead for Gennaro *et al.* [37] and Chung *et al.* [38]. Gennaro *et al.* [37] assigns each bit of data with a large random number and encrypts this random number using fully homomorphic encryption. Chung *et al.* [38] has to transmit κ number of ciphertexts of similar queries/results to process a single query. The difference between their solutions and our work may become more clear when authenticating non-aggregate range selection query: Both Gennaro *et al.* [37] and Chung *et al.* [38] will require linear or even superlinear communication overhead, while our solution sill requires $O(d^2 \log^2 \mathcal{Z})$ communication cost.

[8] Although our solution only supports a very small range of functions.

1. *Given security parameter κ, Alice generates a key k: $k \leftarrow \mathsf{KGen}(1^\kappa)$.*
2. *Alice encodes database $\mathbf{D} \in \mathbb{D}$: $(\mathbf{D}_p, \mathbf{D}_s) \leftarrow \mathsf{DEnc}(\mathbf{D}, k)$, then sends $\mathbf{D}_p$ to Bob and keeps $\mathbf{D}_s$.*
3. *Alice selects a query $\mathbf{R} \in \mathbb{R}$.*
4. *Algorithm $\mathsf{Eval}(\mathbf{D}_p)$ on Bob's side, interacts with algorithm $\mathsf{Ext}(\mathbf{D}_s, \mathbf{R}, k)$ on Alice's side, to compute $(\zeta, X, \boldsymbol{\Psi}) \leftarrow \langle \mathsf{Eval}(\mathbf{D}_p), \mathsf{Ext}(\mathbf{D}_s, \mathbf{R}, k)\rangle$, 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 say a $\mathcal{RC}$ protocol is provable, if the following conditions hold: (1) Alice accepts with o.h.p. (overwhelming high probability), when Bob follows the protocol honestly; (2) Alice rejects with o.h.p., 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 $\langle \mathsf{Eval}(\mathbf{D}_p), \mathsf{Ext}(\mathbf{D}_s, r, k)\rangle = (\mathtt{accept},\ F(\mathbf{D},\ \mathbf{R}), \boldsymbol{\Psi})$ for some $\boldsymbol{\Psi}$, where $(\mathbf{D}_p, \mathbf{D}_s) \leftarrow \mathsf{DEnc}(\mathbf{D}, k)$.*
- *soundness: for any PPT (adaptive) adversary $\mathcal{A}$, the advantage $\mathsf{Adv}_{\mathcal{E},\mathcal{A}}(1^\kappa) \le negl(\kappa)$ (asymptotically less or equal).*

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

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

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

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

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

We remark that the security model is inspired by the formulation of $\mathcal{POR}$ (Proof of Retrievability) [47] and *Computationally Sound Proof System* (Definition 4.8.1 in [48]).

2.3 Assumptions

Throughout the whole paper, let p be a κ bits safe prime, and $e : \mathbb{G} \times \mathbb{G} \rightarrow \widetilde{\mathbb{G}}$ 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*

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

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

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

Assumption 2 (Generalized KEA [28, 33, 1]) *Let $\mathcal{A}$ and $\bar{\mathcal{A}}$ be two algorithms. We define the* **GKEA**-*advantage of $\mathcal{A}$ against $\bar{\mathcal{A}}$ as*

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

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 [34] proved **KEA1** and **KEA3** in *generic group model*. Following their techniques, **GKEA** can be proved in generic group model.

Furthermore, the (Decision) ℓ-wBDHI Assumption [2] is required for the IND-sID-CCA security of the underlying BBG HIBE scheme.

3 Expansion Scheme

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

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

where $\widehat{M} = M\Omega^{s(1-z)}$, $\widehat{g} = g^{z^{-1} \bmod p}$, $\widehat{g}_3 = g_3^{z^{-1} \bmod p}$ and $\widehat{h}_i = h_i^{z^{-1} \bmod p}$ for $1 \leq i \leq \ell$. 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) ID maps an integer $x \in [\mathcal{Z}]$ into an identity $\mathsf{ID}(x) \in \left(\mathbb{Z}_p^*\right)^{\ell}$. (2) IdSet 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 $x \in [a, b]$, 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* [40], 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 in the binary interval tree from the root node $\mathtt{v}_1$ to the node $\mathtt{v}_m$. 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 Expansion Scheme based on HIBE

Let (Setup, KeyGen, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$) be the BBG Hierarchical Identity Based Encryption proposed by Boneh *et al.* [2] (We provide the description of this BBG HIBE scheme in Appendix A). Based on this HIBE scheme, we construct an expansion scheme as in Figure 1.

Lemma 1 *The scheme* ($\mathsf{ExpKGen}, \mathsf{ExpSetup}, \mathsf{ExpDelKey}, \mathsf{Expand}$) *constructed in Figure 1 satisfies this property: For any* $(pk, sk) \leftarrow \mathsf{ExpKGen}(1^{\kappa})$, *for any dataset* $\mathbf{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset [\mathcal{Z}]^d$, *range* $\mathbf{R} \subset [\mathcal{Z}]^d$, *nonce* $\rho \in \mathbb{Z}_p^*$, *and for any* $\boldsymbol{x}_j \in \mathbf{D} \cap \mathbf{R}$, *it always holds that*

$$\mathsf{Expand}(\mathbf{R}, \boldsymbol{\delta}, \boldsymbol{c}_j, \boldsymbol{x}_j, j, pk) = f_2(j) \cdot \Omega^{\rho f_1(j)},$$

where $\{\boldsymbol{c}_i : i \in [N]\} \leftarrow \mathsf{ExpSetup}(\mathbf{D}, \langle f_1 \rangle, \langle f_2 \rangle, sk)$, $\boldsymbol{\delta} \leftarrow \mathsf{ExpDelKey}(\mathbf{R}, \rho, sk)$, *function* $f_1 : [N] \to \mathbb{Z}_p^*$, *function* $f_2 : [N] \to \widetilde{\mathbb{G}}$, *and* $\Omega = e(g_1, g_2) \in \widetilde{\mathbb{G}}$ *is part of* pk. *(The proof is in Appendix C.)*

If $\boldsymbol{x}_j \notin \mathbf{D} \cap \mathbf{R}$, it is required that output of $\mathsf{Expand}(\mathbf{R}, \boldsymbol{\delta}, \boldsymbol{c}_j, j, pk)$ is not equal to $f_2(j) \cdot \Omega^{\rho f_1(j)}$ with o.h.p. This requirement is necessary but not sufficient for the implementation of our authentication scheme in Section 4. In this paper, we do not formulize or prove the security of expansion scheme *separately.*

4 The Construction of MAIA

The construction of Maia is in Figure 2.

Remark.

1. Alice chooses function $f_1 : [N] \to \mathbb{Z}_p^*$ in this way: Choose random elements $z_1, z_2, \ldots, z_N$ from $\mathbb{Z}_p^*$ independently, and sets $f_1(i) = z_i, i \in [N]$. Similar for f_2. After the setup phase, Alice will not keep information of f_1 and f_2, and thus cannot evaluate these functions any more.
2. 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 (5), and the fourth component $v^{\gamma} w^{\rho}$ is the output of $\mathsf{Expand}_{pk}(\mathbf{R}, \boldsymbol{\delta}, \boldsymbol{c}_i, \boldsymbol{x}_i, i, pk)$ w.r.t. the random

Fig. 1: Construction of Expansion scheme based on BBG [2] HIBE Scheme (Setup, KeyGen, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$)

Expansion Scheme

$\mathsf{ExpKGen}(1^\kappa)$

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

$\mathsf{ExpSetup}(\mathbf{D}, \langle f_1 \rangle, \langle f_2 \rangle, sk)$: $\langle f_1 \rangle$ `and` $\langle f_2 \rangle$ `are descriptions of functions` f_1 `and` f_2`, respectively`

For each $\boldsymbol{x}_i = (x_{i,1}, \ldots, x_{i,d}) \in \mathbf{D}$: Choose d random elements $s_{i,1}, \ldots, s_{i,d}$ from $\mathbb{Z}_p$ with constraint $f_1(i) = -\sum_{j=1}^d s_{i,j}\tau_j \pmod p$, choose d random elements $\eta_{i,1}, \ldots, \eta_{i,d}$ from $\widetilde{\mathbb{G}}$ with constraint $f_2(i) = \prod_{j=1}^d \eta_{i,j}$, and choose d random elements $\sigma_{i,1}, \ldots, \sigma_{i,d}$ from $\widetilde{\mathbb{G}}$ with constraint $\prod_{j=1}^d \sigma_{i,j} = \Omega^{-\sum_{j=1}^d s_{i,j}}$. For each $j \in [d]$, encrypt $\eta_{i,j} \cdot \sigma_{i,j}$ under identity $\mathsf{ID}_j(x_{i,j})$ with random coin $s_{i,j}$ to obtain ciphertext $c_{i,j}$ as follows

$$c_{i,j} \leftarrow \mathsf{Encrypt}(\mathtt{params},\ \mathsf{ID}_j(x_{i,j}),\ \eta_{i,j} \cdot \sigma_{i,j};\ s_{i,j}), \tag{4}$$

Let $\boldsymbol{c}_i = (c_{i,1}, \ldots, c_{i,d})$. Output $\{\boldsymbol{c}_i : i \in [N]\}$.

$\mathsf{ExpDelKey}(\mathbf{R}, \rho, sk)$

Parse the d-dimensional range $\mathbf{R}$ as $\mathbf{A}_1 \times \mathbf{A}_2 \ldots \times \mathbf{A}_d \subseteq [\mathcal{Z}]^d$ and parse the private key sk as $(\mathsf{params}, \mathsf{master\text{-}key}, \boldsymbol{\tau})$, where $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_d)$. For each $j \in [d]$, generate δ_j in this way: For each identity $\mathtt{id} \in \mathsf{IdSet}_j(\mathbf{A}_j)$, generate the private key $d_{\mathtt{id}}$, using algorithm KeyGen and taking $\mathsf{master\text{-}key}^{\rho\tau_j}$ as the master key. Set $\delta_j \leftarrow \{d_{\mathtt{id}} : \mathtt{id} \in \mathsf{IdSet}_j(\mathbf{A}_j)\}$. Let $\boldsymbol{\delta} = (\delta_1, \delta_2, \ldots, \delta_d)$ and output $\boldsymbol{\delta}$.

$\mathsf{Expand}(\mathbf{R}, \boldsymbol{\delta}, \boldsymbol{c}_i, \boldsymbol{x}_i, i, pk)$

Parse the d-dimensional range $\mathbf{R}$ as $\mathbf{A}_1 \times \mathbf{A}_2 \ldots \times \mathbf{A}_d \subseteq [\mathcal{Z}]^d$ and parse $\boldsymbol{c}_i$ as $(c_{i,1}, \ldots, c_{i,d})$. For each $j \in [d]$, generate $\widetilde{t}_{i,j}$ in this way: If $\boldsymbol{x}_i[j] \notin \mathbf{A}_j$, then output $\perp$ and abort. Otherwise, do the followings: (1) Find the unique identity $\mathtt{id}^* \in \mathsf{IdSet}_j(\mathbf{A}_j)$ such that $\mathtt{id}^*$ is a prefix of identity $\mathsf{ID}_j(\boldsymbol{x}_i[j])$. (2) Parse $\boldsymbol{\delta}$ as $(\delta_1, \ldots, \delta_d)$ and find the private key $d_{\mathtt{id}^*} \in \delta_j = \{d_{\mathtt{id}} : \mathtt{id} \in \mathsf{IdSet}_j(\mathbf{A}_j)\}$ for identity $\mathtt{id}^*$. (3) Generate the private key $d_{i,j}$ for the identity $\mathsf{ID}_j(\boldsymbol{x}_i[j])$ from private key $d_{\mathtt{id}^*}$, using algorithm KeyGen. (4) Decrypt $c_{i,j}$ using algorithm $\mathsf{Decrypt}$ with decryption key $d_{i,j}$, and denote the decrypted message as $\widetilde{t}_{i,j}$. Let $\widetilde{t}_i = \prod_{1 \le j \le d} \widetilde{t}_{i,j}$. Output $\widetilde{t}_i$.

nonce ρ. 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 $\mathcal{K} = (\beta, \gamma, \theta)$ 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.

3. 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 $\mathsf{Expand}_{pk}(\mathbf{R}, \boldsymbol{\delta}, \boldsymbol{c}_i, \boldsymbol{x}_i, i, pk)$ w.r.t. the random nonce ρ. We give up this alternative and introduce a new component in our construction for the sake of proof. A part of our proof strategy 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 expansion scheme (particularly $\mathsf{ExpSetup}$), the simulator has to find functions f_1 and f_2, such that for any ρ, $f_2(i)\Omega^{\rho f_1(i)} = v_i^\rho$ (See Lemma 1), 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 function f_1 freely, set $w_i = \Omega^{f_1(i)}$, and set $f_2(i) = \left(v_i^\beta\right)^{\gamma'}$.

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

Fig. 2: Construction of $\mathsf{MAIA}=(\mathsf{KGen}, \mathsf{DEnc}, \langle \mathsf{Ext}, \mathsf{Eval}\rangle)$, 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 $\mathsf{ExpKGen}(1^\kappa)$ to obtain public/private key pair (pk, sk).

Note: the public key pk contains as part a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \widetilde{\mathbb{G}}$ and an element $\Omega \in \widetilde{\mathbb{G}}$, where 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 $\mathcal{K} = (\beta, \gamma, \theta)$.

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

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

Step 1: Choose function $f_1 : [N] \to \mathbb{Z}_p^*$ at random, and function $f_2 : [N] \to \widetilde{\mathbb{G}}$ at random.

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

$$v_i \leftarrow f_2(i)^{\gamma^{-1}}; \;\; w_i \leftarrow \Omega^{f_1(i)}; \;\; \boldsymbol{t}_i \leftarrow \left(\theta v_i, \; v_i^\beta, \; w_i\right). \tag{5}$$

Step 3: Run the setup algorithm $\mathsf{ExpSetup}$ of the expansion scheme w.r.t. functions f_1, f_2:

$$\{\boldsymbol{c}_i : i \in [N]\} \leftarrow \mathsf{ExpSetup}(\mathbf{D}, \langle f_1\rangle, \langle f_2\rangle, sk). \tag{6}$$

Note: $\langle f_1\rangle$ and $\langle f_2\rangle$ are descriptions of functions f_1 and f_2, respectively.

Step 4: Send $\mathbf{D}_p = (\mathbf{D}, \; \mathbf{T} = \{\boldsymbol{t}_i : i \in [N]\}, \; \mathbf{C} = \{\boldsymbol{c}_i : i \in [N]\}, pk)$ to Bob, and keep key $(\mathcal{K}, pk, sk)$ and $\mathbf{D}_s = \left(N, \pi^* = \prod_{i\in[N]} v_i^\beta\right)$ in local storage.

(Alice, Bob) $\mathsf{ProVer} = \langle \mathsf{Ext}(N, \pi^*; \; \mathbf{R}; \; \mathcal{K}, sk, pk), \mathsf{Eval}(\mathbf{D}, \mathbf{T}, \mathbf{C}, pk)\rangle$:

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 $\zeta = \texttt{accept}$, if the following equalities hold

$$\forall 0 \le \ell \le 2d, \zeta_\ell \overset{?}{=} \texttt{accept}, \qquad \prod_{0\le\ell\le2d} \Psi_2^{(\ell)} \overset{?}{\equiv} \pi^*; \tag{7}$$

otherwise sets $\zeta = \texttt{reject}$. Alice outputs (ζ, X_0, π^*).

(Alice, Bob) $\mathsf{CollRes} = \langle \widetilde{\mathsf{Ext}}(N, \pi^*; \; \mathbf{R}; \; \mathcal{K}, sk, pk), \widetilde{\mathsf{Eval}}(\mathbf{D}, \mathbf{T}, \mathbf{C}, pk)\rangle$:

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

Step 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 $\mathsf{ExpDelKey}$: $\boldsymbol{\delta} \leftarrow \mathsf{ExpDelKey}(\mathbf{R}, \rho, sk)$. 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

$$X \leftarrow |\mathbf{D} \cap \mathbf{R}|; \qquad (\Psi_1, \Psi_2, \Psi_3) \leftarrow \bigotimes_{\boldsymbol{x}_i \in \mathbf{D}\cap\mathbf{R}} \boldsymbol{t}_i; \qquad \Psi_4 \leftarrow \prod_{\boldsymbol{x}_i \in \mathbf{D}\cap\mathbf{R}} \mathsf{Expand}(\mathbf{R}, \boldsymbol{\delta}, \boldsymbol{c}_i, \boldsymbol{x}_i, i, pk). \tag{8}$$

Bob sends $(X, \Psi_1, \Psi_2, \Psi_3, \Psi_4)$ to Alice.

Note: For $\boldsymbol{x}_i \in \mathbf{D}\cap\mathbf{R}$, $\mathsf{Expand}(\mathbf{R}, \boldsymbol{\delta}, \boldsymbol{c}_i, \boldsymbol{x}_i, i, pk)$ is supposed to output $v_i^\gamma w_i^\rho$; 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 $\zeta = \texttt{accept}$, if the following equalities hold

$$\Lambda^\beta \overset{?}{=} \Psi_2, \qquad \Lambda^\gamma \Psi_3^\rho \overset{?}{=} \Psi_4. \tag{9}$$

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

Consequently, $f_2(i)\Omega^{\rho f_1(i)} = v_i^{\beta\gamma'} w_i^{\rho}$ as desired (taking γ as $\beta\gamma'$). Thus the simulation of ExpSetup can be done.

4. To ensure completeness and prevent double counting or undercounting, we need to run CollRes on the complement query range $R^{\complement}$. Due to the limitation of our construction of expansion scheme, we have to divide range $R^{\complement}$ into multiple rectangular ranges, and then run CollRes on each of them.
5. The $(2d+1)$ invocations of CollRes can be executed in parallel.
6. Like [37, 38], 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 2 (Main Theorem) *Suppose Assumption 1 and Assumption 2 hold, and BBG [2] HIBE scheme is IND-sID-CCA secure. Then the $\mathcal{RC}$ protocol $\mathcal{E}$ = (KGen, DEnc, 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.)*

5.2 Overview of Proof

To process a query, our scheme (particularly the algorithm ProVer) invokes $(2d+1)$ instances of protocol CollRes. In each instance of 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 (7)) 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 CollRes and pass the equation (7). Therefore, an adversary against $\mathcal{E}$ = (KGen, DEnc, ProVer) is also an adversary against $\widetilde{\mathcal{E}}$ = (KGen, DEnc, 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, $1 \le i \le N$, such that $\Psi_2 = \prod_{i\in[N]} \left(v_i^{\beta}\right)^{\mu_i}$, where β and v_i's are as in Figure 2.
– Type III adversary: The same as Type II adversary, with additional constraint: $\mu_i = 0$ for $\boldsymbol{x}_i \in \mathbf{D} \cap \mathbf{R}^{\complement}$.
– 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 2, we introduce and prove Lemma 5, Theorem 6, and Theorem 7. Due to the space constraint, we put these lemmas/theorems together with their full proof in appendix: Lemma 5 is in Appendix D.2, Theorem 6 is in Appendix E, and Theorem 7 is in Appendix F. Additionally, the proof of Theorem 2 based on the above lemmas/theorems is in Appendix G. Basically, our proof framework is like this:

– Lemma 5: 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 $\bar{A}$ in GKEA.
– Theorem 6: If there exists a Type II adversary which is not in Type III, then there exists a PPT algorithm to break the IND-sID-CCA security of the underlying BBG HIBE scheme.

[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 $\boldsymbol{x}_i$.

– Theorem 7: 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 2: 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 2 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 5 and Theorem 6 focus on the partial scheme $\widetilde{\mathcal{E}}$ and Theorem 7 and Theorem 2 focus on the whole scheme $\mathcal{E}$.

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

$$\left.\begin{array}{r}\left.\begin{array}{r}\left.\begin{array}{r}\textbf{Assumption } 2 \Rightarrow \textbf{ Lemma } 4 \Rightarrow \textbf{ Lemma } 5\\ \mathsf{BBG}\text{ is }\mathsf{IND\text{-}sID\text{-}CCA}\text{ secure [2]}\\ \textbf{Assumption } 1\end{array}\right\} \Rightarrow \textbf{ Theorem } 6\\ \textbf{Assumption } 1 \Rightarrow \textbf{DLP Assumption}\end{array}\right\} \Rightarrow \textbf{ Theorem } 7\\ \textbf{Assumption } 1\\ \textbf{Lemma } 1\end{array}\right\} \Rightarrow \textbf{ Theorem } 2$$

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 [37,38]: 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 3 *Suppose Assumption 1 and Assumption 2 hold for group G 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, $\mathcal{E}$ is* correct *and* sound *w.r.t. function F.*

Proof (sketch of Theorem 3). 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.* [33]. First, we construct a **GKEA** adversary $\mathcal{A}$ based on the Type I adversary $\mathcal{B}$:

---

1. The input is $\{(u_i, u_i^{\beta}) : 1 \le i \le m\}$.
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 $\mathbf{D} = \{x_1, x_2, \ldots, x_{m+1}\}$ 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.*

[13] This is just the definition of "successful adversary".

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 $\mathcal{A}$'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 $\bar{\mathcal{A}}$, 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 $\bar{\mathcal{A}}$ which just outputs $\{\mu_i : 1 \le i \le m+1\}$, where $\mu_{m+1} = X - \sum_{i=1}^m \mu_i$. 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.

---

1. The input is (v, v^a, u). The goal is to find u^a
2. Define function f: For each $x_i \in \mathbf{D}$, choose z_i at random and set $f(x_i) = v^{z_i}$.
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 above.
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 $(X, \Psi_1, \Psi_2, \Psi_3, \mu_1, \dots, \mu_N)$ 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.

---

1. The input is (v, v^a). The goal is to find a.
2. Define function f: For each $x_i \in \mathbf{D}$, choose z_i at random and set $f(x_i) = v^{z_i} v^a$; otherwise, set $f(x)$ to a random number.
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 $\mathbf{R}^\complement$.

[14] Including the random coin.

5. If the adversary succeeds, we have

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

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 $\mathcal{A}$ 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 $O(dN)$ and the dominant step is ExpSetup in 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 Expand in Step B1 of CollRes in Figure 2; (2) As a 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 ExpDelKey in 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 scheme and techniques can be useful in other applications, and the idea and construction of expansion scheme may have independent interest. We will look into the possibility to support dynamic operations on the dataset, like adding or deleting a point, by exploiting the dynamic property of the identity hierarchy tree.

References

1. Wu, J., Stinson, D.: An Efficient Identification Protocol and the Knowledge-of-Exponent Assumption. Cryptology ePrint Archive, Report 2007/479 (2007) `http://eprint.iacr.org/`.
2. Boneh, D., Boyen, X., Goh, E.J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: EUROCRYPT. (2005) 440–456
3. Devanbu, P.T., Gertz, M., Martel, C.U., Stubblebine, S.G.: Authentic Third-party Data Publication. In: Proceedings of the IFIP TC11/ WG11.3 Fourteenth Annual Working Conference on Database Security, 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. Hacigümüs, H., Iyer, B.R., Mehrotra, S.: Efficient Execution of Aggregation Queries over Encrypted Relational Databases. In: DASFAA. (2004) 125–136
7. 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
8. Ge, T., Zdonik, S.B.: Answering Aggregation Queries in a Secure System Model. In: VLDB. (2007) 519–530
9. Martel, C., Nuckolls, G., Devanbu, P., Gertz, M., Kwong, A., Stubblebine, S.G.: A General Model for Authenticated Data Structures. Algorithmica **39**(1) (2004) 21–41
10. Devanbu, P., Gertz, M., Martel, C., Stubblebine, S.G.: Authentic data publication over the internet. J. Comput. Secur. **11**(3) (2003) 291–314
11. 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
12. Pang, H., Tan, K.L.: Verifying Completeness of Relational Query Answers from Online Servers. ACM Trans. Inf. Syst. Secur. **11**(2) (2008) 1–50
13. Sion, R.: Query Execution Assurance for Outsourced Databases. In: VLDB. (2005) 601–612
14. Cheng, W., Tan, K.L.: Query assurance verification for outsourced multi-dimensional databases. J. Comput. Secur. **17**(1) (2009) 101–126
15. Li, F., Hadjieleftheriou, M., Kollios, G., Reyzin, L.: Dynamic authenticated index structures for outsourced databases. In: SIGMOD Conference. (2006) 121–132
16. 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
17. 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
18. Yang, Y., Papadias, D., Papadopoulos, S., Kalnis, P.: Authenticated join processing in outsourced databases. In: SIGMOD Conference. (2009) 5–18
19. 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
20. 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)
21. Goodrich, M.T., Tamassia, R., Triandopoulos, N.: Super-Efficient Verification of Dynamic Outsourced Databases. In: CT-RSA. (2008) 407–424
22. 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)
23. 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)
24. Preparata, F.P., Shamos, M.I.: Computational geometry: an introduction. Springer-Verlag New York, Inc., New York, NY, USA (1985)
25. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC '09: ACM symposium on Theory of computing. (2009) 169–178
26. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully Homomorphic Encryption over the Integers. In: EUROCRYPT. (2010) 24–43
27. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory **22**(6) (1976) 644 – 654
28. 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
29. 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
30. Bellare, M., Palacio, A.: Towards Plaintext-Aware Public-Key Encryption Without Random Oracles. In: ASIACRYPT. (2004) 48–62
31. Krawczyk, H.: HMQV: A High-Performance Secure Diffie-Hellman Protocol. In: CRYPTO. (2005) 546–566
32. Dent, A.W.: The Cramer-Shoup Encryption Scheme Is Plaintext Aware in the Standard Model. In: EUROCRYPT. (2006) 289–307

33. Bellare, M., Palacio, A.: The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols. In: CRYPTO. (2004) 273–289
34. Abe, M., Fehr, S.: Perfect NIZK with adaptive soundness. In: TCC '07: Theory of Cryptography Conference. (2007)
35. 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.
36. 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
37. Gennaro, R., Gentry, C., Parno, B.: Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In: CRYPTO. (2010) 465–482
38. Chung, K.M., Kalai, Y., Vadhan, S.P.: Improved Delegation of Computation Using Fully Homomorphic Encryption. In: CRYPTO. (2010) 483–501
39. Gentry, C.: Toward Basing Fully Homomorphic Encryption on Worst-Case Hardness. In: CRYPTO. (2010) 116–137
40. 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
41. 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
42. 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
43. 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
44. Bowers, K.D., Juels, A., Oprea, A.: HAIL: A High-Availability and Integrity Layer for Cloud Storage. Cryptology ePrint Archive, Report 2008/489 (2008)
45. Dodis, Y., Vadhan, S., Wichs, D.: Proofs of Retrievability via Hardness Amplification. In: TCC '09: Theory of Cryptography Conference. (2009) 109–127
46. 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
47. 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
48. Goldreich, O.: Foundations of Cryptography: Volume 1. Cambridge University Press, New York, NY, USA (2006)

A HIBE

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

$\mathsf{Setup}(\ell)$

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

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

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

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

$$\mathsf{CT} = \left(\Omega^s \cdot M,\ g^s,\ \left(h_1^{I_1} \ldots h_k^{I_k} \cdot g_3\right)^s\right) \in \widetilde{\mathbb{G}} \times \mathbb{G}^2. \tag{10}$$

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

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

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

For a valid ciphertext, we have

$$\frac{e(K_1, C)}{e(B, K_0)} = \frac{e\left(g^r, \left(h_1^{I_1} \ldots h_k^{I_k} \cdot g_3\right)^s\right)}{e\left(g^s, g_2^\alpha \left(h_1^{I_1} \ldots h_k^{I_k} \cdot g_3\right)^r\right)} = \frac{1}{e(g^s, g_2^\alpha)} = \frac{1}{e(g_1, g_2)^s} = \frac{1}{\Omega^s}. \tag{11}$$

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 $\mathsf{Pr}[A] \le \mathsf{Pr}[B]$.*

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

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

□

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

Proof.

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

□

C Proof of Lemma 1

Lemma 1 *The scheme* $(\mathsf{ExpKGen}, \mathsf{ExpSetup}, \mathsf{ExpDelKey}, \mathsf{Expand})$ *constructed in Figure 1 satisfies this property: For any $(pk, sk) \leftarrow \mathsf{ExpKGen}(1^\kappa)$, for any dataset $\mathbf{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset [\mathcal{Z}]^d$, range $\mathbf{R} \subset [\mathcal{Z}]^d$, nonce $\rho \in \mathbb{Z}_p^*$, and for any $\boldsymbol{x}_j \in \mathbf{D} \cap \mathbf{R}$, it always holds that*

$$\mathsf{Expand}(\mathbf{R}, \boldsymbol{\delta}, \boldsymbol{c}_j, \boldsymbol{x}_j, j, pk) = f_2(j) \cdot \Omega^{\rho f_1(j)},$$

where $\{\boldsymbol{c}_i : i \in [N]\} \leftarrow \mathsf{ExpSetup}(\mathbf{D}, \langle f_1 \rangle, \langle f_2 \rangle, sk)$, $\boldsymbol{\delta} \leftarrow \mathsf{ExpDelKey}(\mathbf{R}, \rho, sk)$, function $f_1 : [N] \to \mathbb{Z}_p$, function $f_2 : [N] \to \widetilde{\mathbb{G}}$, and $\Omega = e(g_1, g_2) \in \widetilde{\mathbb{G}}$ is part of pk. Note $\langle f_1 \rangle$ and $\langle f_2 \rangle$ are descriptions of functions f_1 and f_2, respectively.

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

$$\begin{aligned}
&\mathsf{CT} = \mathsf{Encrypt}(\mathsf{params}, \mathsf{ID}, M; s) = \left(\Omega^s \cdot M,\ g^s,\ \left(h_1^{I_1} \cdots h_k^{I_k} \cdot g_3\right)^s\right) \\
&\quad \text{under key: } \mathtt{params} = (g, g_1, g_2, g_3, h_1, \dots, h_\ell, \Omega = e(g_1, g_2)),\quad \mathtt{master\text{-}key} = g_2^\alpha \\
&\widehat{\mathsf{CT}} = \mathsf{Encrypt}(\widehat{\mathsf{params}}, \mathsf{ID}, \widehat{M}; sz) = \left(\Omega^{sz} \cdot \widehat{M},\ \widehat{g}^{sz},\ \left(\widehat{h}_1^{I_1} \cdots \widehat{h}_k^{I_k} \cdot \widehat{g}_3\right)^{sz}\right), \\
&\quad \text{under key: } \widehat{\mathtt{params}} = (\widehat{g}, g_1, g_2, \widehat{g}_3, \widehat{h}_1, \dots, \widehat{h}_\ell, \Omega = e(g_1, g_2)),\quad \widehat{\mathtt{master\text{-}key}} = g_2^{\alpha z} \qquad (12)
\end{aligned}$$

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

Let $(pk, sk) \leftarrow \mathsf{ExpKGen}(1^\kappa)$. Given $\mathbf{D} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$, $\mathbf{R}$, ρ from proper domains. Let $\{\boldsymbol{c}_i : i \in [N]\} \leftarrow \mathsf{ExpSetup}(\mathbf{D}, \langle f_1 \rangle, \langle f_2 \rangle, sk)$ and $\boldsymbol{\delta} \leftarrow \mathsf{ExpDelKey}(\mathbf{R}, \rho, sk)$.

We consider dimension $j \in [d]$ and apply the polymorphism property of BBG scheme (equation (12)): Take $M = \eta_{i,j}\sigma_{i,j}, s = s_{i,j}$ and $z = \rho\tau_j$. Then $\widehat{M} = M\Omega^{s(1-z)} = \eta_{i,j}\sigma_{i,j}\Omega^{s_{i,j}(1-\tau_j\rho)}$. Therefore, if $\boldsymbol{x}_i \in \mathbf{D} \cap \mathbf{R}$, then the HIBE decryption will succeed in the process of Expand and we have

$$\widetilde{t}_{i,j} = \widehat{M} = \eta_{i,j} w_{i,j} \Omega^{s_{i,j}(1-\tau_j\rho)}, j \in [d].$$

Combining all d dimensions, and applying the three equalities (see algorithm $\mathsf{ExpSetup}$ in Figure 1) $f_1(i) = -\sum_{j=1}^d s_{i,j}\tau_j \mod p$, $f_2(i) = \prod_{j=1}^d \eta_{i,j}$ and $\prod_{j=1}^d \sigma_{i,j} = \Omega^{-\sum_j^{s_{i,j}}}$ we have,

$$\begin{aligned}
\widetilde{t}_i = \prod_{j=1}^d \widetilde{t}_{i,j} &= \prod_{j=1}^d \left(\eta_{i,j}\sigma_{i,j}\Omega^{s_{i,j}(1-\tau_j\rho)}\right) \\
&= \prod_{j=1}^d \eta_{i,j} \cdot \prod_{j=1}^d \sigma_{i,j} \cdot \prod_{j=1}^d \Omega^{s_{i,j}} \cdot \left(\prod_{j=1}^d \Omega^{-s_{i,j}\tau_j}\right)^\rho \\
&= f_2(i) \cdot \Omega^{-\sum_{j=1}^d s_{i,j}} \cdot \prod_{j=1}^d \Omega^{s_{i,j}} \cdot \left(\Omega^{f_1(i)}\right)^\rho \\
&= f_2(i) \cdot \Omega^{\rho f_1(i)}.
\end{aligned}$$

□

D 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 5. We prove Lemma 5 in two steps: first we show that the **GKEA** Assumption 2 implies Lemma 4 (which states that an alternative form of **GKEA** problem is hard); then we derive Lemma 5 from Lemma 4.

We remark that both the proof of Lemma 4 in Appendix D.1 and proof of Lemma 5 in Appendix D.2 follow the proof framework for statement that **KEA3** implies **KEA** in [33]. 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).

Lemma 4 *Suppose Assumption 2 holds. For any PPT algorithm $\mathcal{A}$, there exists a PPT algorithm $\bar{\mathcal{A}}$, such that*

$$\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\mathbf{Lem\ 4}}(\kappa) \overset{\text{def}}{=} \Pr\left[\begin{array}{l}
S_{m+1} \leftarrow \{(\theta v_i, v_i^\beta) : i \in [m+1], v_i \xleftarrow{\$} \widetilde{\mathbb{G}}, \theta \xleftarrow{\$} \widetilde{\mathbb{G}}, \beta \xleftarrow{\$} \mathbb{Z}_p^*\} \\
(\Psi_1, \Psi_2, X) \leftarrow \mathcal{A}(S_{m+1}; r); \\
(\Psi_1, \Psi_2, X, \mu_1, \mu_2, \dots, \mu_m, \mu_{m+1}) \leftarrow \bar{\mathcal{A}}(S_{m+1}; r, \bar{r}) : \\
\qquad \Psi_2 = \left(\frac{\Psi_1}{\theta^X}\right)^\beta \ \wedge\ \Psi_2 \ne \prod_{j=1}^{m+1} (v_j^\beta)^{\mu_j}
\end{array}\right] \le \nu_2(\kappa), \qquad (13)$$

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.

D.1 Proof of Lemma 4

Proof (of Lemma 4). Let adversary $\mathcal{A}$ be as in Lemma 4. We construct a **GKEA** adversary $\mathcal{A}_1$ based on an adversary $\mathcal{A}$.

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

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

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

$$\left(\frac{\Psi_1}{\theta^X}\right)^\beta = \Psi_2 \qquad \Leftrightarrow \qquad \left(\frac{\Psi_1}{R_1^X}\right)^\beta = \frac{\Psi_2}{R_2^X}. \tag{14}$$

According to Assumption 2, there exists an extractor $\bar{\mathcal{A}}_1$ for the adversary $\mathcal{A}_1$, such that $\mathsf{Adv}_{\mathcal{A}_1,\bar{\mathcal{A}}_1}^{\mathbf{GKEA}}$ 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,

$$U_2 = \prod_{i=1}^m u_i^{\beta\mu_i} \qquad \Rightarrow \qquad \prod_{i=1}^{m+1} v_i^{\beta\mu_i} = \Psi_2. \tag{15}$$

If $U_1^\beta = U_2 \;\Rightarrow\; U_2 = \prod_{i=1}^m u_i^{\beta\mu_i}$, combining with equation (14) and equation (15), 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 (14) and equation (15) 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

$$\mathsf{Pr}\left[U_1^{\beta} = U_2 \Rightarrow U_2 = \prod_{i=1}^{m} u_i^{\beta\mu_i}\right] = 1 - \mathsf{Adv}_{\mathcal{A}_1,\bar{\mathcal{A}}_1}^{\mathbf{GKEA}} \le \mathsf{Pr}\left[\left(\frac{\Psi_1}{\theta^X}\right)^{\beta} = \Psi_2 \Rightarrow \Psi_2 = \prod_{i=1}^{m+1} v_i^{\beta\mu_i}\right] = 1 - \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\mathbf{Lem\ 4}}.$$

Hence,

$$\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\mathbf{Lem\ 4}} \le \mathsf{Adv}_{\mathcal{A}_1,\bar{\mathcal{A}}_1}^{\mathbf{GKEA}} \le \nu_2.$$

□

D.2 Proof of Lemma 5

Lemma 5 *Suppose Assumption 2 holds. For any PPT algorithm $\mathcal{A}$, there exists a PPT algorithm $\bar{\mathcal{A}}$, such that* $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\mathbf{Lem\ 5}}(1^{\kappa}) \le \nu_2(\kappa)$*, where the advantage* $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\mathbf{Lem\ 5}}$ *of $\mathcal{A}$ against $\bar{\mathcal{A}}$ w.r.t. scheme* $\widetilde{\mathcal{E}} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{CollRes})$ *is defined as*

$$\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\mathbf{Lem\ 5}}(1^{\kappa}) \stackrel{\text{def}}{=} 1 - \mathsf{Pr}\left[\begin{array}{l}(\zeta, X, \Psi_2, \mathsf{view}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}, \mathbf{D}, \mathbf{R}) \leftarrow \mathsf{Exp}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}(1^{\kappa});\\ (\{\mu_i : i \in [N]\}) \leftarrow \bar{\mathcal{A}}(\mathsf{view}_{\mathcal{A}}^{\mathcal{E}}) :\\ \qquad \zeta = \mathtt{accept} \ \Rightarrow\ \Psi_2 = \prod_{i\in[N]} \left(v_i^{\beta}\right)^{\mu_i}\end{array}\right],$$

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

Adversary $\mathcal{B}$ w.r.t. Lemma 4: Based on $\mathcal{A}$

1. The input is $S_m = \{(\theta v_j, v_j^{\beta}) \in \widetilde{\mathbb{G}}^2 : j \in [m]\}$. The random coin used in this algorithm is r.
2. Simulate Alice in the experiment $\mathsf{Exp}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$.
   (a) Invoke adversary $\mathcal{A}$ with a random coin derived from r. $\mathcal{A}$ chooses a set $\mathbf{D} = \{\boldsymbol{x}_i \in [\mathcal{Z}]^d : i \in [N]\}$ of $N = m$ d-dimensional data points. *Note: If $N > m$, $\mathcal{B}$ can generate more tuples $(\theta v_j, v_j^{\beta})$ for $j = m+1, \ldots, N$ from S_m. For simplicity, we just assume $N = m$.*
   (b) Simulate KGen:
      i. Invoke $\mathsf{ExpKGen}(1^{\kappa})$ 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 a function $f_1 : [N] \to \mathbb{Z}_p^*$ at random and define function $f_2(i) \stackrel{\text{def}}{=} \left(v_i^{\beta}\right)^{\gamma'}$, $i \in [N]$.
      ii. For each $i \in [N]$, let $\boldsymbol{t}_i = \left(\theta v_i, v_i^{\beta}, \Omega^{f_1(i)}\right)$.
      iii. Invoke $\mathsf{ExpSetup}(\mathbf{D}, \langle f_1\rangle, \langle f_2\rangle, sk)$ to obtain $\mathbf{C} = \{\boldsymbol{c}_i : i \in [N]\}$, where $\langle f_1\rangle$ and $\langle f_2\rangle$ are descriptions of functions f_1 and f_2 respectively.
   (d) Simulate $\mathsf{CollRes}$: For each query range $\mathbf{R}_j$ received from $\mathcal{A}$ during $\mathcal{A}$'s learning phase and challenging phase
      i. (Step A1) Choose random nonce $\rho \in \mathbb{Z}_p^*$ and invoke algorithm $\mathsf{ExpDelKey}(\mathbf{R}_j, \rho, sk)$ using the private key sk.
      ii. (Step A2) Does 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 $\mathsf{CollRes}$.
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 $\mathsf{view}_{\mathcal{A}}^{\mathcal{E}}$ of $\mathcal{A}$ after interacting with the real scheme in the experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathcal{E}}$.
   (a) KGen: The simulator $\mathcal{B}$ generates key (pk, sk) in the same way as the real scheme. The unkonw 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 as in real experiment. Functions f_1 and f_2 are truly randomly chosen as in the real experiment. The ciphertexts $\mathbf{C}$ are generated in the same way as in real experiment. As a result, the public encoding $\mathbf{D}_p = \{\mathbf{D}, \mathbf{T}, \mathbf{C}\}$ $\mathcal{A}$ 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) and 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 4, there exists a PPT algorithm $\bar{\mathcal{B}}$ such that $\mathsf{Adv}_{\mathcal{B},\bar{\mathcal{B}}}^{\textbf{Lem 4}} \leq \nu_2$. 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 $\mathsf{Adv}_{\mathcal{B},\bar{\mathcal{B}}}^{\textbf{Lem 4}} \leq \nu_2$. Invoke $\bar{\mathcal{B}}$ on input S_N using random coin $\mathsf{view}_{\mathcal{A}}^{\mathcal{E}}$, 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 $\mathsf{CollRes}$, $\zeta = \texttt{accept}$ implies that the verification is passed and $\Psi_2 = \left(\frac{\Psi_1}{\theta^X}\right)^{\beta}$ (the first equality test in equation (9)). We have

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

Hence, by applying Proposition 1, we have

$$\begin{aligned} \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem 5}} &= \Pr\left[\zeta = \texttt{accept} \ \wedge \ \Psi_2 \neq \prod_{i \in [N]} (v_i^{\beta})^{\mu_i}\right] \\ \leq \mathsf{Adv}_{\mathcal{B},\bar{\mathcal{B}}}^{\textbf{Lem 4}} &= \Pr\left[\Psi_2 = \left(\frac{\Psi_1}{\theta^X}\right)^{\beta} \ \wedge \ \Psi_2 \neq \prod_{i \in [N]} (v_i^{\beta})^{\mu_i}\right] \leq \nu_2. \end{aligned}$$

□

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

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

Theorem 6 *Suppose Assumption 1 and Assumption 2 hold, and BBG [2] HIBE scheme is IND-sID-CCA secure. For any PPT algorithm $\mathcal{A}$, there exists PPT algorithm $\bar{\mathcal{A}}$, such that both $\mathsf{Adv}^{\mathbf{Lem\ 5}}_{\mathcal{A},\bar{\mathcal{A}}}$ and $\mathsf{Adv}^{\mathbf{Thm\ 6}}_{\mathcal{A},\bar{\mathcal{A}}}$ are negligible, where the advantage $\mathsf{Adv}^{\mathbf{Thm\ 6}}_{\mathcal{A},\bar{\mathcal{A}}}$ of $\mathcal{A}$ against $\bar{\mathcal{A}}$ w.r.t. scheme $\widetilde{\mathcal{E}} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{CollRes})$ is defined as*

$$\mathsf{Adv}^{\mathbf{Thm\ 6}}_{\mathcal{A},\bar{\mathcal{A}}}(1^\kappa) \overset{\text{def}}{=} 1 - \Pr\left[\begin{array}{l}(\zeta, X, \Psi_2, \mathsf{view}^{\widetilde{\mathcal{E}}}_{\mathcal{A}}, \mathbf{D}, \mathbf{R}) \leftarrow \mathsf{Exp}^{\widetilde{\mathcal{E}}}_{\mathcal{A}}(1^\kappa);\\ (\{\mu_i : i \in [N]\}) \leftarrow \bar{\mathcal{A}}(\mathsf{view}^{\widetilde{\mathcal{E}}}_{\mathcal{A}}) :\\ \qquad \zeta = \mathtt{accept} \wedge \Psi_2 = \prod_{i\in[N]} \left(v_i^\beta\right)^{\mu_i} \ \Rightarrow\ \forall \boldsymbol{x}_i \in \mathbf{D} \cap \mathbf{R}^{\mathsf{C}}, \mu_i = 0\end{array}\right],$$

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 6).

The idea of proof. For any PPT algorithm $\mathcal{A}$, applying Lemma 5, let $\bar{\mathcal{A}}$ be the PPT algorithm such that $\mathsf{Adv}^{\mathbf{Lem\ 5}}_{\mathcal{A},\bar{\mathcal{A}}}$ is negligible. Using proof of contradiction, assume that $\mathsf{Adv}^{\mathbf{Thm\ 6}}_{\mathcal{A},\bar{\mathcal{A}}}$ is non-negligible. Based on $\mathcal{A}$ and $\bar{\mathcal{A}}$, we construct a PPT algorithm $\mathcal{B}$, such that $\mathcal{B}$ breaks IND-sID-CCA security of BBG HIBE scheme [2] with non-negligible advantage

$$\mathsf{Adv}^{\text{IND-sID-CCA}}_{\mathsf{BBG},\mathcal{B}} \geq \frac{1}{4dN}\mathsf{Adv}^{\mathbf{Thm\ 6}}_{\mathcal{A},\bar{\mathcal{A}}} - \frac{1}{4}\nu_1.$$

where ν_1 is as in Assumption 1 and $\mathsf{Adv}^{\text{IND-sID-CCA}}_{\mathsf{BBG},\mathcal{B}}$ is defined in [2]. The contradiction implies that our hypothesis is wrong, and thus Theorem 6 is proved.

Construct IND-sID-CCA adversary $\mathcal{B}$ against BBG HIBE based on $\bar{\mathcal{A}}$:

IND-sID-CCA adversary $\mathcal{B}$ against BBG HIBE

1. Setup:
   (a) Invoke adversary $\mathcal{A}$: $\mathcal{A}$ chooses a dataset $\mathbf{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset [\mathcal{Z}]^d$ of N d dimensional points.
   (b) Choose $i^* \in [N]$ at random and $\xi \in [1, d]$ at random. Denote with $\mathtt{id}^*$ the identity associate with the ξ-th dimension of $\boldsymbol{x}_{i^*}$, i.e. $\mathtt{id}^* = \mathsf{ID}_\xi(\boldsymbol{x}_{i^*}[\xi])$. Send $\mathtt{id}^*$ to the IND-sID-CCA challenger.
   (c) The IND-sID-CCA challenger runs BBG setup algorithm Setup and generates a pair of key (master-key, params). The challenger gives params to $\mathcal{B}$ and keeps master-key private.
2. Phase 1: Do nothing.
3. Challenge: Choose a function $f_2 : [N] \to \widetilde{\mathbb{G}}$ at random. Let $M_0 = f_2(i^*)$ and choose M_1 from $\widetilde{\mathbb{G}}$ at random. Send (M_0, M_1) to the challenger and receive the challenging ciphertext $\mathsf{CT} = \mathsf{Encrypt}(\mathsf{params}, \mathtt{id}^*, M_b)$, $b \in \{0, 1\}$, from the IND-sID-CCA challenger.
   Simulate Alice in the experiment $\mathsf{Exp}^{\widetilde{\mathcal{E}}}_{\mathcal{A}}$ as follows.
   Simulate KGen:
   – Choose $\beta, \gamma \in \mathbb{Z}_p^*$ and $\theta \in \widetilde{\mathbb{G}}$ at random. Let $\mathcal{K} = (\beta, \gamma, \theta)$.
   – Choose $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_d) \in \left(\mathbb{Z}_p^*\right)^d$ at random. Let $pk = (p, \mathbb{G}, \widetilde{\mathbb{G}}, e)$ and $sk = (pk, \mathsf{params}, \mathsf{master\text{-}key}, \boldsymbol{\tau})$. Take (pk, sk) as the output of $\mathsf{ExpKGen}(1^\kappa)$. *Note: $\mathcal{B}$ knows only params and does not know master-key.*
   Simulate DEnc:
   – Choose a function $f_1 : [N] \to \mathbb{Z}_p$ at random.
   – For $i \in [N]$, generate $\boldsymbol{t}_i$ in the same way as in the Step 2 of algorithm DEnc in Figure 2.
   – $\mathcal{B}$ executes algorithm ExpSetup on dataset $\mathbf{D}$ w.r.t. functions f_1 and f_2, with params and $\boldsymbol{\tau}$, and generates $\mathbf{C} = \{\boldsymbol{c}_i : i \in [N]\}$. *Note: In our construction in Figure 1, the execution of* ExpSetup *requires only the public key params for BBG encryption, and secret value $\boldsymbol{\tau}$ to find numbers $u_{i,j}$'s. Full information of sk (particularly master-key) is not necessary.*
   – Redefine $\boldsymbol{c}_{i^*}$ based on challenging ciphertext CT:
     - Choose $(d-1)$ random elements $s_{i^*,1}, \ldots, s_{i^*,\xi-1}, s_{i^*,\xi+1}, \ldots, s_{i^*,d}$ from $\mathbf{Z}_p^*$. Choose d random elements $\eta_{i^*,1}, \ldots, \eta_{i^*,d}$ from $\widetilde{\mathbb{G}}$ with constraint $f_2(i^*) = \prod_{j=1}^d \eta_{i^*,j}$.
     - Parse the challenging ciphertext CT as (c_1, c_2, c_3) and $c_1 = M_b \Omega^s$ for some unknown s. Let $s_{i^*,\xi}$ be the unknown value, such that $\Omega^{s_{i^*,\xi}} = c_1 M_0^{-1}$. Choose d random elements $\sigma_{i^*,1}, \ldots, \sigma_{i^*,d}$ from $\widetilde{\mathbb{G}}$ with constraint

$$\prod_{j=1}^d \sigma_{i^*,j} = \Omega^{-\sum_{j=1}^d s_{i^*,j}} = \Omega^{-\sum_{\substack{j\in[d]\\ j\neq\xi}} s_{i^*,j}} \cdot c_1^{-1} M_0$$

       Note: In case of $b = 0$, $s_{i^*,\xi} = s$.
     - For each $j \in [d]$ and $j \neq \xi$, generate $\boldsymbol{c}_{i^*,j}$ in the same way as in algorithm ExpSetup: $c_{i_{j^*},j}$ is the ciphertext of $\eta_{i^*,j} w_{i^*,j}$ under identity $\mathsf{ID}_j(\boldsymbol{x}_{i^*}[j])$ with random coin $s_{i^*,j}$.

  - For dimension ξ, $c_{i^*,\xi} = (c_1\sigma_{i^*,\xi}, c_2, c_3)$, where $\mathsf{CT} = (c_1, c_2, c_3)$. Set $\boldsymbol{c}_{i^*} = (c_{i^*,1}, \ldots, c_{i^*,d})$.
- Redefine tag $\boldsymbol{t}_{i^*}$ by setting its third component w_{i^*} as

$$\Omega^{-\sum_{j=1}^{d} s_{i^*,j}\tau_j} = \Omega^{-\sum_{j\in[d], j\neq\xi} s_{i^*,j}\tau_j} \cdot \Omega^{-s_{i^*,\xi}\tau_\xi} = \Omega^{-\sum_{j\in[d], j\neq\xi} s_{i^*,j}\tau_j} \cdot \left(c_1^{-1}M_0\right)^{\tau_\xi}$$

Note: After this redefinitions of $\boldsymbol{c}_{i^*}$ and $\boldsymbol{t}_{i^*}$, the new tags and ciphertexts $(\mathbf{T}, \mathbf{C})$ are consistent w.r.t. to functions f_1' and f_2, where $f_1'(i) = f_1(i)$ if $i \neq i^*$ and $i \in [N]$, and $f_1'(i^*) = -\sum_{j=1}^{d} s_{i^*,j}\tau_j$ is unknown since $s_{i^*,\xi}$ is unknown. However, the value $\Omega^{f_1'(i^*)}$ can be computed since $\Omega^{s_{i^*,\xi}} = c_1 M_0^{-1}$.

4. Phase 2: Simulate $\mathcal{A}$. Choose $Z \in \widetilde{\mathbb{G}}$ at random. For every query $\mathbf{R}_i$ made by $\mathcal{A}$, $\mathcal{B}$ responds in the following way.
   - $\mathcal{A}$ is in learning phase: Taking Z as the master key, execute Step A1 of $\mathsf{CollRes}$ in Figure 2, and do nothing in Step A2.
     Note: There exists an unknown ϖ such that $\mathsf{master\text{-}key}^{\varpi} = Z$. If the simulator chooses random nonce ρ', then the generated Help-Info $\boldsymbol{\delta}$ *is identical to the one generated from* $\mathsf{master\text{-}key}$ *and random nonce $\rho = \rho'\varpi$. Since ϖ is unknown, the simulator is unable to do verifications in Step A2 of* $\mathsf{CollRes}$.
   - $\mathcal{A}$ is in challenging phase: The challenging query range is $\mathbf{R}$.
     - Parse $\mathbf{R}$ as $\mathbf{A}_1 \times \mathbf{A}_2 \times \ldots \mathbf{A}_d$, where $\mathbf{A}_j \subset [\mathcal{Z}], j \in [d]$.
     - If $\boldsymbol{x}_{i^*}[\xi] \notin \mathbf{A}_\xi$, then make a corresponding query to $\mathsf{IND\text{-}sID\text{-}CCA}$ challenger, and forward its response to $\mathcal{A}$ after some conversion.
       * For any $\mathtt{id} \in \mathsf{IdSet}_j(\mathbf{A}_j), j \in [d]$, send a private key query with identity $\mathtt{id}$ to the challenger. Let the $d_{\mathtt{id}} = (a_0, a_1, b_k, \ldots, b_\ell)$ be the response from the challenger. Choose ρ at random from $\mathbb{Z}_p^*$, and compute $d'_{\mathtt{id}} = (a_0^{\rho\tau_j}, a_1^{\rho\tau_j}, b_k^{\rho\tau_j}, \ldots, b_\ell^{\rho\tau_j})$. Send $\boldsymbol{\delta} = \{d'_{\mathtt{id}} : \mathtt{id} \in \mathsf{IdSet}_j(\mathbf{A}_j), j \in [d]\}$ to $\mathcal{A}$ as *Help-Info* for query range $\mathbf{R}$.
       * Received reply from $\mathcal{A}$. Do verifications and obtain output (ζ, X, Ψ) as in Step A2 of $\mathsf{CollRes}$.

       Note: For this query, the simulator can perform the verifications since it knows (β, γ, ρ).
     - If $\boldsymbol{x}_{i^*}[\xi] \in \mathbf{A}_\xi$, then abort and output a random bit $b' \in \{0, 1\}$. Denote this event as $\mathbf{E}_1$.
5. Guess: Eventually, $\mathcal{A}$ outputs $\{\mu_i : i \in [N]\}$.
   - 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 $\mathbf{E}_2$.
   - Otherwise, then abort and output a random bit $b' \in \{0, 1\}$. Denote this event as $\mathbf{E}_3$.

Remarks on algorithm $\mathcal{B}$.

1. The constructed algorithm $\mathcal{B}$ is a PPT $\mathsf{IND\text{-}sID\text{-}CCA}$ adversary against BBG HIBE scheme with $O(d \log \mathcal{Z})$ chosen private key query and no chosen decryption queries.
2. In the learning phase, the simulator takes a random number $Z \in \widetilde{\mathbb{G}}$ as the master key and "forge" the *Help-Info* by itself. As a result, the simulator does not know the value of "real" ranom nonce $\rho = \rho'\varpi$ and thus cannot perform some verification (i.e. the second equality test in equation (9)) in Step A2 of $\mathsf{CollRes}$.
3. In the challenging phase, the simulator sends private key queries to the $\mathsf{IND\text{-}sID\text{-}CCA}$ challenger, and produce the *Help-Info* using the response from the challenger together with a random nonce ρ of its own choice. In this case, the simulator can perform the all verifications in Step A2 of $\mathsf{CollRes}$.
4. In case of $b = 0$: $\Omega^{s_{i^*,\xi}} = c_1 M_0^{-1} = M_b \Omega^s M_0^{-1} = \Omega^s$, and the new value of $c_{i^*,\xi}$ is a BBG encryption of $\eta_{i^*,\xi}\sigma_{i^*,\xi}$ under identity $\mathsf{ID}_\xi(\boldsymbol{x}_{i^*}[\xi])$ with random coin $s_{i^*,\xi} = s$. After redefining the tag $\boldsymbol{t}_{i^*}$ and ciphertext $\boldsymbol{c}_{i^*}$ based on the challenging ciphertext CT, the new $(\mathbf{T}, \mathbf{C})$ is consistent with functions f_1' and f_2, and $\mathsf{Expand}(\mathbf{R}, \boldsymbol{\delta}, \boldsymbol{c}_{i^*}, \boldsymbol{x}_{i^*}, i^*, pk)$ will output $f_2(i^*)\Omega^{f_1'(i^*)}$. The simulated scheme is identical to a real one (with functions f_1' and f_2), from the view of adversary.
5. In the case of $b = 1$, $\Omega^{s_{i^*,\xi}} = c_1 M_0^{-1} = M_b \Omega^s M_0^{-1} = \Omega^s \cdot \left(M_1 M_0^{-1}\right)$, $s_{i^*,\xi}$ is randomly distributed independent on s. the generated ciphertext $c_{i^*,\xi}$ is a valid BBG encryption of $\eta_{i^*,\xi}\sigma_{i^*,\xi}$ under indentity $\mathsf{ID}_\xi(\boldsymbol{x}_{i^*}[\xi])$ with random coin $s \neq s_{i^*,\xi}$. Thus the output of $\mathsf{Expand}(\mathbf{R}, \boldsymbol{\delta}, \boldsymbol{c}_{i^*}, \boldsymbol{x}_{i^*}, i^*, pk)$ is not equal to $f_2(i^*)\Omega^{f_1'(i^*)}$ with o.h.p, and the simulated scheme is (information theoretically) not identical to the real scheme, from the view of adversary (Note adversary has $\mathbf{C} = \{\boldsymbol{c}_i : i \in [N]\}$).

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.

$$\begin{aligned}
\mathsf{Pr}[b = b'] &= \frac{1}{2}\mathsf{Pr}\left[\mathbf{E}_1 \vee \mathbf{E}_3\right] + \mathsf{Pr}\left[b = b', \mathbf{E}_2\right] \\
&= \frac{1}{2}(1 - \mathsf{Pr}\left[\mathbf{E}_2\right]) + \mathsf{Pr}\left[b = b', \mathbf{E}_2\right] \\
&= \frac{1}{2} + \mathsf{Pr}\left[b = b', \mathbf{E}_2\right] - \frac{1}{2}\mathsf{Pr}\left[\mathbf{E}_2\right] \qquad (16)
\end{aligned}$$

Therefore,

$$\mathsf{Adv}^{\mathsf{IND\text{-}sID\text{-}CCA}}_{\mathsf{BBG},\mathcal{B}} = \left|\Pr\left[b=b', \mathbf{E}_2\right] - \frac{1}{2}\Pr\left[\mathbf{E}_2\right]\right| \tag{17}$$

$$\geq \left| \left|\frac{1}{2}\Pr\left[b=b', \mathbf{E}_2 \mid b=0\right] - \frac{1}{4}\Pr\left[\mathbf{E}_2 \mid b=0\right]\right| - \left|\frac{1}{2}\Pr\left[b=b', \mathbf{E}_2 \mid b=1\right] - \frac{1}{4}\Pr\left[\mathbf{E}_2 \mid b=1\right]\right| \right| \tag{18}$$

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

$$\Pr[\mathbf{E}_4 \mid b=0] = \Pr\left[\zeta = \texttt{accept} \wedge \Psi_2 = \prod_{i\in[N]}\left(v_i^{\beta}\right)^{\mu_i} \wedge \exists \boldsymbol{x}_i \in \mathbf{D}\cap\mathbf{R}^{\mathsf{C}}, \mu_i \neq 0 \mid b=0\right] = \mathsf{Adv}^{\textbf{Thm 6}}_{\mathcal{A},\bar{\mathcal{A}}}$$

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 $\mathbf{E}_5$, where $\mathbf{E}_5$ represents the event $i^* \in S_{\#} = \{i \in [N] : \boldsymbol{x}_i \in \mathbf{D}\cap\mathbf{R}^{\mathsf{C}}, \mu_i \neq 0\}$. 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

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

Therefore,

$$\begin{aligned}\Pr\left[\mathbf{E}_2 \mid b=0\right] &= \Pr\left[\neg\mathbf{E}_1 \wedge \mathbf{E}_4 \wedge \mathbf{E}_5 \mid b=0\right] = \Pr\left[\neg\mathbf{E}_1 \mid \mathbf{E}_4 \wedge \mathbf{E}_5 \wedge b=0\right]\Pr\left[\mathbf{E}_4 \wedge \mathbf{E}_5 \mid b=0\right]\\ &\geq \frac{1}{d}\Pr\left[\mathbf{E}_4 \mid b=0\right]\Pr\left[\mathbf{E}_5 \mid \mathbf{E}_4, b=0\right]\\ &= \frac{1}{d}\mathsf{Adv}^{\textbf{Thm 6}}_{\mathcal{A},\bar{\mathcal{A}}} \cdot \frac{1}{|S_{\#}|} \geq \frac{1}{dN}\mathsf{Adv}^{\textbf{Thm 6}}_{\mathcal{A},\bar{\mathcal{A}}} \qquad (\textit{ Event } \mathbf{E}_4 \textit{ implies that } S_{\#} \neq \emptyset)\end{aligned}$$

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

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

$$\mathsf{Adv}^{\mathsf{IND\text{-}sID\text{-}CCA}}_{\mathsf{BBG},\mathcal{B}}|_{b=0} = \left|\Pr\left[\mathbf{E}_2 \mid b=0\right]\left(\Pr\left[b=b' \mid \mathbf{E}_2, b=0\right] - \frac{1}{2}\right)\right| \geq \frac{1}{2dN}\mathsf{Adv}^{\textbf{Thm 6}}_{\mathcal{A},\bar{\mathcal{A}}}. \tag{19}$$

$\mathsf{Adv}^{\mathsf{IND\text{-}sID\text{-}CCA}}_{\mathsf{BBG},\mathcal{B}}$ conditional on $b=1$.
Next we show that $\Pr[\mathbf{E}_2 \mid b=1]$ is negligible under CDH assumption.

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

Proof (of Claim E.01). The proof idea is as below: Given input (v, v^{γ}, u), we simulates the scheme $\widetilde{\mathcal{E}} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{CollRes})$ and the $\mathsf{IND\text{-}sID\text{-}CCA}$ adversary $\mathcal{B}$. When event $\mathbf{E}_2$ occurs, we try to compute u^{γ}.

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

1. Input is $(v, v^a, u) \in \widetilde{\mathbb{G}}$, where $a \in \mathbb{Z}_p^*$ is uniformly randomly distributed. The goal is to output u^a.
2. Simulate the scheme $\widetilde{\mathcal{E}} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{CollRes})$:
   (a) Run algorithm $\mathsf{KGen}, \mathsf{DEnc}, \mathsf{CollRes}$ as normal, except that:
      i. Let γ be the unknown value a.
      ii. Let the set $\mathbf{D}$ with N d-dimensional points be the dataset chosen by adversary $\mathcal{A}$.

iii. Choose $i^* \in [N]$ at random and $\xi \in [d]$ at random.

iv. Choose function f_1 freely and define function f_2 in this way: (1) For each $i \in [N]$ and $i \neq i^*$, choose $z_i \in \mathbb{Z}_p^*$ at random and set $f_2(i) = (v^\gamma)^{z_i}$; (2) Choose $Z \in \widetilde{\mathbb{G}}$ at random and set $f_2(i^*) = Z$. *Although γ is unknown, we have $f_2(i)^{\gamma^{-1}} = v^{z_i}$ for any $i \in [N]$ and $i \neq i^*$.*

v. For point $\boldsymbol{x}_{i^*}$, set $v_{i^*} = u$ and the tag $\boldsymbol{t}_{i^*} = (\theta \cdot v_{i^*}, u^\beta, \Omega^{f_1(i^*)})$.

Note: Since γ is unknown, some verifications can not be done.

3. Simulate the IND-sID-CCA adversary $\mathcal{B}$ based on the above simulated scheme, except that
   (a) Do not redefine $\boldsymbol{c}_{i^*}$ or $\boldsymbol{t}_{i^*}$.
   (b) When the adversary $\mathcal{A}$ is in challenging phase, the verification cannot be done.
4. 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. Compute ϕ as below and output $\phi^{\mu_{i^*}^{-1}}$:

$$\phi \leftarrow \frac{\Psi_4}{\Psi_3^\rho \cdot \prod_{i\in[N]}^{i\neq i^*} (v_i^\gamma)^{\mu_i}}$$

In case of $b = 1$, the algorithm $\mathcal{B}$ is identical to the one simulated by $\mathcal{C}$, to the view of adversary $\mathcal{A}$: except tag $\boldsymbol{t}_{i^*}$ and $\boldsymbol{c}_{i^*}$, all are identical to the experiment $\mathsf{Exp}_{\mathcal{A}}^{\widetilde{\mathcal{E}}}$ with a real scheme as constructed in Figure 2; for point $\boldsymbol{x}_{i^*}$, the output of $\mathsf{Expand}(\mathbf{R}, \boldsymbol{\delta}, \boldsymbol{c}_{i^*}, \boldsymbol{x}_{i^*}, i^*, pk)$ is random and independent on the desired value $u^\gamma w_{i^*}^\rho$.

Suppose $b = 1$ and event $\mathbf{E}_2$ occurs, that is, $\zeta = \texttt{accept}$ 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 $\mathcal{C}$ 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, $\mathsf{Pr}[\mathbf{E}_2 \mid b = 1] = \nu_1$, where $\nu_1(\cdot)$ is some negligible function. As a result, conditional on $b = 0$, the advantage of $\mathcal{B}$ is

$$\mathsf{Adv}_{\mathsf{BBG},\mathcal{B}}^{\mathsf{IND\text{-}sID\text{-}CCA}}|_{b=1} = \left| \mathsf{Pr}\left[\mathbf{E}_2 \mid b = 1\right] \left(\mathsf{Pr}\left[b = b' \mid \mathbf{E}_2, b = 1\right] - \frac{1}{2}\right) \right| \leq \frac{1}{2}\nu_1. \tag{20}$$

$$\mathsf{Adv}_{\mathsf{BBG},\mathcal{B}}^{\mathsf{IND\text{-}sID\text{-}CCA}} \geq \left| \frac{1}{2}\mathsf{Adv}_{\mathsf{BBG},\mathcal{B}}^{\mathsf{IND\text{-}sID\text{-}CCA}}|_{b=0} - \frac{1}{2}\mathsf{Adv}_{\mathsf{BBG},\mathcal{B}}^{\mathsf{IND\text{-}sID\text{-}CCA}}|_{b=1} \right| \geq \frac{1}{4dN}\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\mathbf{Thm\ 6}} - \frac{1}{4}\nu_1.$$

□

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

Theorem 7 *Suppose Assumption 1 and Assumption 2 hold, and BBG [2] HIBE scheme is IND-sID-CCA secure. For any PPT algorithm $\mathcal{A}$, there exists a PPT adversary $\bar{\mathcal{A}}$, such that all of $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\mathbf{Lem\ 5}}$, $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\mathbf{Thm\ 6}}$, and $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\mathbf{Thm\ 7}}$ are negligible, where the advantage $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\mathbf{Thm\ 7}}$ of $\mathcal{A}$ against $\bar{\mathcal{A}}$ w.r.t. scheme $\mathcal{E} = (\mathsf{KGen}, \mathsf{DEnc}, \mathsf{ProVer})$ is defined as*

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

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 7).

Idea of proof. For any PPT algorithm $\mathcal{A}$, applying Theorem 6, let $\bar{\mathcal{A}}$ be the PPT algorithm, such that $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem 5}} \leq \epsilon_5$ and $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 6}} \leq \epsilon_6$ for some negligible functions $\epsilon_5(\cdot)$ and $\epsilon_6(\cdot)$. Using proof of contradiction, assume that $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 7}} \geq \epsilon_7$ 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 $\zeta = \texttt{accept} \bigwedge \pi^* \neq \prod_{i\in[N]} \left(v_i^{\beta}\right)^{\mu_i}$, and with $\mathbf{E}_2$ the event that $\zeta = \texttt{accept} \bigwedge \pi^* = \prod_{i\in[N]} \left(v_i^{\beta}\right)^{\mu_i} \wedge \exists j \in [N], \mu_j \neq 1$. We can split the probability $\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 7}}$ into two parts,

$$\begin{aligned}\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 7}} &= \mathsf{Pr}\left[\begin{array}{l}(\zeta, X, \boldsymbol{\Psi}, \mathsf{view}_{\mathcal{A}}^{\mathcal{E}}, \mathbf{D}, \mathbf{R}) \leftarrow \mathsf{Exp}_{\mathcal{A}}^{\mathcal{E}}(1^{\kappa});\\ (\{\mu_i : i \in [N]\}) \leftarrow \bar{\mathcal{A}}(\mathsf{view}_{\mathcal{A}}^{\mathcal{E}}) :\\ \qquad \zeta = \texttt{accept} \bigwedge \pi^* \neq \prod_{i\in[N]} \left(v_i^{\beta}\right)^{\mu_i}\end{array}\right] + \mathsf{Pr}\left[\begin{array}{l}(\zeta, X, \boldsymbol{\Psi}, \mathsf{view}_{\mathcal{A}}^{\mathcal{E}}, \mathbf{D}, \mathbf{R}) \leftarrow \mathsf{Exp}_{\mathcal{A}}^{\mathcal{E}}(1^{\kappa});\\ (\{\mu_i : i \in [N]\}) \leftarrow \bar{\mathcal{A}}(\mathsf{view}_{\mathcal{A}}^{\mathcal{E}}) :\\ \qquad \zeta = \texttt{accept} \bigwedge \pi^* = \prod_{i\in[N]} \left(v_i^{\beta}\right)^{\mu_i}\\ \wedge \exists j \in [N], \mu_j \neq 1\end{array}\right]\\ &= \mathsf{Pr}[\mathbf{E}_1] + \mathsf{Pr}[\mathbf{E}_2].\end{aligned}$$

Part I: $\mathsf{Pr}[\mathbf{E}_1] \leq (2d+1)\left(\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem 5}} + \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 6}}\right)$. 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, π^*) the output of Alice in the execution of ProVer. (5) Recall that Alice keeps the value $\pi^* = \prod_{i\in[N]} v_i^{\beta}$.

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

$$\left(\bigwedge_{\ell\in[0,2d]} \zeta_\ell = \texttt{accept}\right) \wedge \pi^* = \prod_{\ell\in[0,2d]} \Psi_2^{(\ell)} \Leftrightarrow \zeta = \texttt{accept} \tag{21}$$

The conjunctions of equation (21) (denoted as statement A) and statements $A_\ell : \zeta_\ell = \texttt{accept} \Rightarrow \Psi_2^{(\ell)} = \prod_{i\in[N]} v_i^{\beta\mu_i}$, $0 \leq \ell \leq 2d$, and statements $B_\ell : \zeta_\ell = \texttt{accept} \wedge \Psi_2^{(\ell)} = \prod_{i\in[N]} v_i^{\beta\mu_i} \Rightarrow \forall \boldsymbol{x}_i \in \mathbf{D} \cap \mathbf{R}^{\complement}, \mu_i = 0$, $0 \leq \ell \leq 2d$, directly imply that

$$\zeta = \texttt{accept} \qquad \Rightarrow \qquad \pi^* = \prod_{\boldsymbol{x}_i\in\mathbf{D}\cap\left(\bigcup_{0\leq\ell\leq 2d}\mathbf{R}_\ell\right)} v_i^{\beta\mu_i} = \prod_{\boldsymbol{x}_i\in\mathbf{D}} v_i^{\beta\mu_i}. \tag{22}$$

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

$$\begin{aligned}\mathsf{Pr}\left[\zeta = \texttt{accept} \Rightarrow \pi^* = \prod_{\boldsymbol{x}_i\in\mathbf{D}} v_i^{\beta\mu_i}\right] &\geq \mathsf{Pr}\left[A \wedge A_0 \wedge \ldots \wedge A_{2d} \wedge B_0 \wedge \ldots \wedge B_{2d}\right]\\ &\geq 1 - \mathsf{Pr}[\neg A] - \sum_{\ell=0}^{2d} \mathsf{Pr}\left[\neg A_\ell\right] - \sum_{\ell=0}^{2d} \mathsf{Pr}\left[\neg B_\ell\right]\\ &\geq 1 - 0 - \sum_{\ell=0}^{2d} \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem 5}} - \sum_{\ell=0}^{2d} \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 6}}\\ &= 1 - (2d+1)\left(\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 6}} + \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem 5}}\right).\end{aligned}$$

Therefore,

$$\mathsf{Pr}[\mathbf{E}_1] = 1 - \mathsf{Pr}\left[\zeta = \texttt{accept} \Rightarrow \pi^* = \prod_{\boldsymbol{x}_i\in\mathbf{D}} v_i^{\beta\mu_i}\right] \leq (2d+1)\left(\mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Lem 5}} + \mathsf{Adv}_{\mathcal{A},\bar{\mathcal{A}}}^{\textbf{Thm 6}}\right).$$

Part II: Break Discrete Log Problem. Applying the result in Part I, we have $\mathsf{Pr}[\mathbf{E}_2] = \mathsf{Adv}^{\textbf{Thm 7}}_{\mathcal{A},\bar{\mathcal{A}}} - \mathsf{Pr}[\mathbf{E}_1] \geq \mathsf{Adv}^{\textbf{Thm 7}}_{\mathcal{A},\bar{\mathcal{A}}} - (2d+1)\left(\mathsf{Adv}^{\textbf{Lem 5}}_{\mathcal{A},\bar{\mathcal{A}}} + \mathsf{Adv}^{\textbf{Thm 6}}_{\mathcal{A},\bar{\mathcal{A}}}\right)$. 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 $z_i \in \mathbb{Z}_p$ at random and set $f(i) = v^{z_i} v^a$.
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 $\bar{\mathcal{A}}$ to obtain $\{\mu_i : i \in [N]\}$.
5. With probability equal to $\mathsf{Pr}[\mathbf{E}_2]$, it holds that $\zeta = \texttt{accept} \bigwedge \pi^* = \prod_{i\in[N]} \left(v_i^{\beta}\right)^{\mu_i} \bigwedge \exists j \in [N], \mu_j \neq 1$.
6. According to our scheme in Figure 2 (Step 4 of DEnc), $\pi^* = \prod_{i\in[N]} v_i^{\beta}$. 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

$$\mathsf{Adv}^{\textbf{Thm 7}}_{\mathcal{A},\bar{\mathcal{A}}} \leq (2d+1)\left(\mathsf{Adv}^{\textbf{Lem 5}}_{\mathcal{A},\bar{\mathcal{A}}} + \mathsf{Adv}^{\textbf{Thm 6}}_{\mathcal{A},\bar{\mathcal{A}}}\right) + \mathsf{Adv}^{\textbf{DLP}}_{\mathcal{B}}.$$

□

G Proof of Main Theorem 2

Theorem 2 (Main Theorem) *Suppose Assumption 1 and Assumption 2 hold, and BBG [2] HIBE scheme is IND-sID-CCA 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 2). 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 $\mathcal{E}$:

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

Applying Theorem 7, let $\bar{\mathcal{A}}$ be the extractor for $\mathcal{A}$ such that all of $\mathsf{Adv}^{\textbf{Lem 5}}_{\mathcal{A},\bar{\mathcal{A}}}$, $\mathsf{Adv}^{\textbf{Thm 6}}_{\mathcal{A},\bar{\mathcal{A}}}$, and $\mathsf{Adv}^{\textbf{Thm 7}}_{\mathcal{A},\bar{\mathcal{A}}}$ are negligible.

We intend to construct a PPT algorithm $\mathcal{B}$ based on $\mathcal{A}$ to break Assumption 1 (Computational Diffie-Hellman Problem), and argue that $\mathcal{B}$ succeeds with probability about ϵ_6, with the help of $\bar{\mathcal{A}}$, under Assumption 1, Assumption 2, and the assumption that BBG [2] HIBE is IND-sID-CCA 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 4 in Appendix D.1.
5. From S_{m+1}, simulate the scheme $\mathcal{E}$ just as adversary $\mathcal{B}$ in the proof of Lemma 5 in Appendix D.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 5, the simulated scheme $\mathcal{E}$ is identical to a real one from the view of adversary $\mathcal{A}$.

Claim G.02 *Suppose Assumption 1 and Assumption 2 hold, and* BBG *[2] HIBE scheme is* IND-sID-CCA *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 G.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

$$\left(\frac{\bar{\Psi}_1}{\theta^X}\right)^\beta = \bar{\Psi}_2, \ \zeta = \texttt{accept}. \tag{23}$$

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 [2] HIBE scheme is IND-sID-CCA secure, by applying Lemma 5, Theorem 6 and Theorem 7, the following implications hold with o.h.p.,

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

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

$$\bar{\Psi}_2 = \prod_{\boldsymbol{x}_i \in \mathbf{D} \cap \mathbf{R}} \left(v_i^\beta\right)^{\mu_i} = \prod_{\boldsymbol{x}_i \in \mathbf{D} \cap \mathbf{R}} v_i^\beta. \tag{24}$$

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*).

$$\left(\frac{\Psi_1}{\theta^Y}\right)^\beta = \Psi_2, \text{ where } \Psi_2 = \prod_{\boldsymbol{x}_i \in \mathbf{D} \cap \mathbf{R}} v_i^\beta \quad \text{is computed following the scheme.} \tag{25}$$

Combing equations (23)(24)(25), Claim G.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 G.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 2 is proved. □