Anonymous Multi-Attribute Encryption with Range Query and Conditional Decryption*

John Bethencourt **T-H. Hubert Chan**
Adrian Perrig **Elaine Shi** **Dawn Song**

May 2006
CMU-CS-06-135

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

*This material is based upon work supported by the National Science Foundation under Grant No. 0448452. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Keywords: Range query, conditional decryption, anonymous identity-based encryption, multi-dimensional lattice.

Abstract

We introduce the concept of Anonymous Multi-Attribute Encryption with Range Query and Conditional Decryption (AMERQCD). In AMERQCD, a plaintext is encrypted under a point in multidimensional space. To a computationally bounded adversary, the ciphertext hides both the plaintext and the point under which it is encrypted. In a range query, a master key owner releases the decryption key for an arbitrary hyper-rectangle in space, thus allowing decryption of ciphertexts previously encrypted under any point within the hyper-rectangle. However, a computationally bounded adversary cannot learn any information on ciphertexts outside the range covered by the decryption key (except the fact that they do not lie within this range). We give an efficient construction based on the Decision Bilinear Diffie-Hellman (D-BDH) and Decision Linear (D-Linear) assumption.

1 Introduction

Searching on encrypted data is an important technique to provide both functionality and privacy in database applications, and has recently captured a considerable amount of attention in the community. Previously, researchers have proposed encryption schemes that allow keyword-based searches [27, 8, 1], and specific types of comparison-based search [11, 10]. In this paper, we propose a more powerful technique that allows range query on encrypted data.

Specifically, we consider the problem of designing an encryption scheme in which data entries consist of a pair $(\mathbf{X}, \mathrm{Msg})$, where the vector $\mathbf{X}$ is a point in some multi-dimensional lattice $\mathcal{U}_\Delta$ and the message Msg is an arbitrary string. Our scheme encrypts the data entries and achieves the following properties.

1. **Range Query & Conditional Decryption.** Upon the request on a region $\mathbf{B} \subseteq \mathcal{U}_\Delta$, the master key owner releases a decryption key $\mathbf{DK}_\mathbf{B}$ such that given a ciphertext $\mathbf{C}$ for some entry $(\mathbf{X}, \mathrm{Msg})$, using the key $\mathbf{DK}_\mathbf{B}$, it is possible to decide whether $\mathbf{X} \in \mathbf{B}$ and it is possible to recover Msg from $\mathbf{C}$ *iff* $\mathbf{X} \in \mathbf{B}$.

2. **Confidentiality.** Given ciphertext $\mathbf{C}$ for entry $(\mathrm{Msg}, \mathbf{X})$, a computationally bounded adversary cannot learn Msg from $\mathbf{C}$, provided that the adversary has not queried the decryption key for a region containing $\mathbf{X}$.

3. **Anonymity.** Given ciphertext $\mathbf{C}$ for entry $(\mathrm{Msg}, \mathbf{X})$, suppose that a computationally bounded adversary has queried for regions $\mathbf{B}_1$, $\mathbf{B}_2$, ..., $\mathbf{B}_q$ all of which do not contain $\mathbf{X}$, the adversary cannot learn anything more about $\mathbf{X}$ from $\mathbf{C}$, apart from the fact that $\mathbf{X}$ does not fall in $\mathbf{B}_1$, $\mathbf{B}_2$, ..., $\mathbf{B}_q$[1].

We refer to this problem as Anonymous Multi-Attribute Encryption with Range Query and Conditional Decryption(AMERQCD).

1.1 Applications

AMERQCD has important applications in database privacy, where data is stored on an untrusted storage server. All database entries are stored encrypted. Under certain circumstances, a master key owner can issue capabilities for a storage server to decrypt entries that satisfy certain conditions. However, the storage server must not be able to learn any information on entries that do not meet the conditions (except that they do not satisfy the conditions). We give a number of concrete examples in this flavor.

Medical privacy. The AMERQCD problem is motivated by medical-privacy applications. When Alice goes around in her daily life, a PDA or smart-phone she carries automatically deposits encrypted crumbs of her trajectory at a storage server. Assume that each crumb is of the form

[1]Note that a potentially stronger security definition could be useful in certain applications, where even if $\mathbf{X}$ lies within a queried region $\mathbf{B}$, nothing is revealed about $\mathbf{X}$ except that it lies within $\mathbf{B}$.

$((x, y, t), ct)$, where (x, y) represents the location, t represents time, and ct is Alice's contact information. During an outbreak of an epidemic, Alice wishes to be alerted if she happened to be present at a site borne with the disease during an incubation period, i.e., if (x, y, t) is in some alert region B. However, she is also concerned with privacy, and she does not wish to leak her trajectory if she has not been to a site borne with the disease.

Untrusted remote storage. Individual users may wish to store emails and files on a remote server, but because the storage server is untrusted, the content must be encrypted before it is stored at the remote server. Emails and files can be classified with multi-dimensional attributes. Users may wish to perform range queries and retrieve only data that satisfy the queries.

Using biometrics in anonymous IBE. The AMERQCD scheme can also be used in biometric-based Anonymous Identity-Based Encryption (AIBE). Using biometrics in identity-based encryption first appeared in [25]. In this application, a person's biometric features such as finger-prints, blood-type, year of birth, eye color, etc., are encoded as a point $\mathbf{X}$ in a multi-dimensional lattice. Personal data is encrypted using the owner's biometric features as the identity, and the encryption protects both the secrecy of the personal data and the owner's biometric identity. Due to potential noise each time a person's biometric features are sampled, a user holding the private key for biometric identity $\mathbf{X}$ should be allowed to decrypt data encrypted under $\mathbf{X}'$, *iff* $\mathbf{X}'$ lies within a certain *Hamming distance* from $\mathbf{X}$. In particular, the decryption region is a hyper-cube if the same weight is assigned to each dimension, and a hyper-rectangle if different weights are assigned to different dimensions.

1.2 Our Result and Contributions

To the best of our knowledge, we are among the earliest to study the problem of point encryption, range query, and conditional decryption of matching entries. Concurrently, Boneh and Waters also proposed ways to perform complex queries over encrypted data [12]. We formally define the AMERQCD problem in Section 3. AMERQCD differs from previously proposed searchable encryption schemes [27, 8, 1] in that 1) we consider range query while previous work addresses keyword-based search; 2) in addition to deciding whether an entry is covered by the specified range we also provide conditional decryption of matching entries.

Table 1 compares the cost of our new construction and some straightforward extensions of related work. We observe that naively representing the lattice using a quadtree [14] or Hilbert curve [21, 24] and using Anonymous Identity-Based Encryption (AIBE) [1, 13] on top results in a decryption key size and decryption cost of $\widetilde{O}(T^{D-1})$ (as described in Section 4.2). Meanwhile, a straightforward extension from a one-dimensional AMERQCD scheme results in $O((\log T)^D)$ encryption cost and ciphertext size (as described in Section 4.2).

We propose a new AMERQCD construction with $O(D \cdot \log T)$ public key size, encryption cost, ciphertext size and decryption key size; and with $O((\log T)^D)$ decryption cost (as described in Section 5). We formally define and prove the security of our AMERQCD scheme in the face of computationally bounded adversaries. Since we borrow techniques from the AHIBE scheme [13],

Scheme	Encrypt. Cost	CT Size	Decrypt. Key Size	Decrypt. Cost
QuadTree/Hilbert	$O(D \cdot \log T)$	$O(D \cdot \log T)$	$O(T^{D-1})$	$\widetilde{O}(T^{D-1})$
Naive ext. AMERQCD (1).	$O((\log T)^D)$	$O((\log T)^D)$	$O((\log T)^D)$	$O((\log T)^D)$
AMERQCD (D)	$O(D \cdot \log T)$	$O(D \cdot \log T)$	$O(D \cdot \log T)$	$O((\log T)^D)$

Table 1: Cost of various approaches. *The number of dimensions of the lattice $\mathcal{U}_\Delta$ is denoted D, and the number of points along each dimension is denoted T. QuadTree/Hilbert, Naive ext. AMERQCD (1) are potential schemes as explained in Section 4.2. AMERQCD (D) is our new construction where the number in the parenthesis stands for the number of dimensions. The notation $\widetilde{O}$ suppresses logarithmic terms.*

the security of our scheme can be likewise reduced to the hardness of the Decision Bilinear Diffie-Hellman problem and the Decision Linear problem. We describe applications of AMERQCD to closely related problems in Appendix A.

2 Related Work

IBE, Anonymous IBE. The notion of IBE was introduced by Shamir [26] two decades ago. Several IBE [17, 9, 5, 4, 16, 28, 23] and HIBE [19, 18, 6] schemes have emerged since then. In particular, the HIBE scheme proposed by Boneh Boyen and Goh [6] can be extended to multiple dimensions efficiently and in a collusion-resistant manner. The resulting M-HIBE scheme can be adapted to solve a related problem which we call Multi-Attribute Encryption with Range Query **(MERCD)** (See Section A.1). M-HIBE is also closely related to Forward Secure HIBE (fs-HIBE) [15, 30], where the number of dimensions is two, one of them being the identity, and the other the time.

Recently researchers have proposed anonymous IBE (AIBE) and Anonymous Hierarchical Identity-Based Encryption (AHIBE) schemes [13, 1]. The most closely related work is the AHIBE [13] scheme recently proposed by Boyen and Waters. Like the HIBE scheme [6] mentioned above, the AHIBE scheme can be extended into multiple dimensions in a collusion-resistant manner, resulting in a Multi-dimension Anonymous Hierarchical Identity-Based Encryption scheme (M-AHIBE). Our work borrows techniques from the AHIBE [13] scheme. In particular, we prevent collusion attacks in the same way that an M-AHIBE scheme does. However, directly applying M-AHIBE to AMERQCD has the following problems: 1) Because the encryption is anonymous and hides the attributes used as the public key, at time of decryption one needs to try all possible decryption keys on a given ciphertext. This incurs $O(T^D)$ decryption cost on a single ciphertext, where $[1, T]$ is the range along one dimension, and D is the number of dimensions. 2) The AMERQCD problem does not require the delegation property necessary to traditional HIBE schemes. By removing the delegation property, we can gain savings on the ciphertext size and encryption cost. Although our AMERQCD scheme borrows techniques from the AHIBE [13] paper, we differ from M-AHIBE in that we address the above two requirements posed by the AMERQCD problem.

Search on encrypted data. Song et al. [27] propose one of the first schemes for searching on encrypted data (SoE). The SoE scheme leverages symmetric key techniques and allows a party that encrypted the data to generate keyword search capabilities. Boneh et al. [8] proposed Public Key Encryption with Keyword Search (PEKS), where any party possessing the public key can encrypt and the owner of the corresponding private key can generate keyword search capabilities. Abdalla et al. [1] formalized the notion of anonymous IBE and its relationship to PEKS. Both SoE and PEKS can be trivially extended to support one-dimensional range queries. The extension is similar to the AMERQCD (1) scheme described in Section 4.1: one builds an interval tree in one dimension, and encrypts the plaintext $O(\log T)$ times along the the path to the root. Because any interval can be represented by $O(\log T)$ nodes in the tree, a master key owner may issue $O(\log T)$ capabilities corresponding to the $O(\log T)$ nodes, to allow the query on any particular range. However, such extensions from SoE or PEKS support only range query (i.e., given a ciphertext $\mathbf{C}$ from $(\mathbf{Msg}, \mathbf{X})$, decide whether $\mathbf{X}$ falls within a certain region), but not the decryption of matching entries.

In recent work by Boneh, Sahai and Waters [10], and by Boneh and Waters [11], they propose how to do a specific type comparison query on encrypted data. Given a ciphertext $\mathbf{C}$ over point $\mathbf{X} = (x_1, x_2, \ldots, x_D)$ in space, a master key owner can issue tokens that allow us to decide whether $x_d \leq x'$ for all $1 \leq d \leq D$ with $O(\sqrt{T})$ ciphertext size and token size.

Concurrent to our work, Boneh and Waters [12] propose how to perform complex queries over encrypted data. In particular, they define a more general and stronger security game for complex queries over encrypted data, and propose a primitive called Hidden Vector System (HVE) that enables several types of queries. Since we have a slightly different security goal, and optimize specifically for range queries, our scheme is more efficient for range queries. In addition, our construction builds on a different complexity assumption from theirs.

Spatial data structure and space-filling curves. Spatial data structures such as quadtrees [14] and KD-trees [2] have been intensively studied by the database community for efficient search and indexing based on spatial locations. A typical spatial data structure encodes a multi-dimensional lattice in a single tree, where each leaf node represents a point and non-leaf nodes represent axis-parallel hyper-rectangles in the lattice. Alternatively, to encode a multi-dimensional lattice with a single tree, one can leverage space-filling curves (e.g., the Hilbert space-filling curve), first introduced by Peano [21, 24]. Space-filling curves embed a multi-dimensional space into a curve of length $O(T^D)$ in a single dimension. The resulting curve can then be encoded using a simple interval tree.

One potential approach to tackling the AMERQCD problem is to apply an AIBE scheme on top of spatial data structures or space-filling curves as described in Section 4.2. However, we observe that using a quadtree or Hilbert curve for our purpose can result in $\widetilde{O}(T^{D-1})$ decryption key size and decryption cost for an arbitrary hyper-rectangle, due to the fact that representing an arbitrary hyper-rectangle can take $\widetilde{O}(T^{D-1})$ nodes in the resulting tree [3, 22].

3 Definitions

We use the notation $[T]$ to denote integers from 1 to T, i.e., $[T] = \{1, 2, \ldots, T\}$. Let $S \leq T$ be integers, we use $[S, T]$ to denote integers from S to T inclusive, i.e., $[S, T] = \{S, S+1, \ldots, T\}$.

Definition 1 (D-dimensional lattice, point, hyper-rectangle)**.** *Let $\Delta = \{T_1, T_2, \ldots, T_D\}$. $\mathcal{U}_\Delta = [T_1] \times [T_2] \times \ldots \times [T_D]$ defines a **D-dimensional lattice**. A D-tuple $\mathbf{X} = (x_1, x_2, \ldots, x_D)$ defines a **point** in $\mathcal{U}_\Delta$, where $x_d \in [T_d] (\forall d \in [D])$. A **hyper-rectangle** $\mathbf{B}$ in $\mathcal{U}_\Delta$ is defined as $\mathbf{B}(s_1, t_1, s_2, t_2, \ldots, s_D, t_D) = \{(x_1, x_2, \ldots, x_D) \big| \forall d \in [D], x_d \in [s_d, t_d]\}$ ($\forall d \in [D], 1 \leq s_d \leq t_d \leq T_d$).*

Throughout this paper, we assume that all T_d's are powers of 2, and denote $\log_2$ as simply $\log$.

Definition 2 (AMERQCD (D))**.** *An Anonymous Multi-Attribute Encryption with Range Query and Conditional Decryption in D-dimensions (AMERQCD (D)) scheme consists of the following polynomial-time randomized algorithms.*

1. $\mathbf{Setup}(\mathbf{\Sigma}, \mathcal{U}_{\mathbf{\Delta}})$: Takes a security parameter Σ and D-dimensional lattice $\mathcal{U}_\Delta$ and outputs public key $\mathbf{PK}$ and master private key $\mathbf{MK}$.

2. $\mathbf{Encrypt}(\mathbf{PK}, \mathbf{X}, \mathbf{Msg})$: Takes a public key $\mathbf{PK}$, a point $\mathbf{X}$, and a message $\mathbf{Msg}$ from the message space $\mathcal{M}$ and outputs a ciphertext $\mathbf{C}$.

3. $\mathbf{DeriveKey}(\mathbf{PK}, \mathbf{MK}, \mathbf{B})$: Takes a public key $\mathbf{PK}$, a master private key $\mathbf{MK}$, and a hyper-rectangle $\mathbf{B}$ and outputs decryption key for hyper-rectangle $\mathbf{B}$.

4. $\mathbf{QueryDecrypt}(\mathbf{PK}, \mathbf{DK}, \mathbf{C})$: Takes a public key $\mathbf{PK}$, a decryption key $\mathbf{DK}$, and a ciphertext $\mathbf{C}$ and outputs either a plaintext $\mathbf{Msg}$ or $\perp$, signaling decryption failure.

For each message $\mathbf{Msg} \in \mathcal{M}$, hyper-rectangle $\mathbf{B} \subseteq \mathcal{U}_\Delta$, and point $\mathbf{X} \in \mathcal{U}_\Delta$, the above algorithms must satisfy the following consistency constraints:

$$\mathbf{QueryDecrypt}(\mathbf{PK}, \mathbf{DK}, \mathbf{C}) = \begin{cases} \mathbf{Msg} & \text{if } \mathbf{X} \in \mathbf{B} \\ \perp & \text{w.h.p., if } \mathbf{X} \notin \mathbf{B} \end{cases} \tag{1}$$

where $\mathbf{C} = \mathbf{Encrypt}(\mathbf{PK}, \mathbf{X}, \mathbf{Msg})$ and $\mathbf{DK} = \mathbf{DeriveKey}(\mathbf{PK}, \mathbf{MK}, \mathbf{B})$.

3.1 Security Definitions

For security, we would like the property that for a ciphertext $\mathbf{C} = \mathbf{Encrypt}(\mathbf{PK}, \mathbf{X}, \mathbf{Msg})$, a computationally bounded adversary cannot learn $\mathbf{X}$ and $\mathbf{Msg}$ from $\mathbf{C}$, given that it has not queried the decryption key for any hyper-rectangle containing $\mathbf{X}$.

Since our scheme builds on top of AIBE, we adopt the terminology from IBE schemes, and define the security for AMERQCD in the sense of selective-ID semantic security and anonymity. In Section 7.1, we explain the relationship between selective-ID and standard notions of security (i.e., adaptive-ID security), and show how our scheme can be made secure in the adaptive-ID sense.

Definition 3 (AMERQCD selective-ID confidentiality game)**.** *The AMERQCD selective-ID confidentiality game is defined as below.*

- **Init**: The adversary $\mathcal{A}$ outputs a point $\mathbf{X}^*$ where it wishes to be challenged.
- **Setup**: The challenger runs the $\mathtt{Setup}(\Sigma, \mathcal{U}_{\boldsymbol{\Delta}})$ algorithm to generate $\mathbf{PK}$, $\mathbf{MK}$. It gives $\mathbf{PK}$ to the adversary, but does not divulge $\mathbf{MK}$.
- **Phase 1**: The adversary is allowed to issue decryption key queries for hyper-rectangles that do not contain $\mathbf{X}^*$.
- **Challenge**: The adversary submits two equal length messages $\mathbf{Msg}_0$ and $\mathbf{Msg}_1$. The challenger flips a random coin, b, and encrypts $\mathbf{Msg}_b$ under $\mathbf{X}^*$. The ciphertext is passed to the adversary.
- **Phase 2**: Phase 1 is repeated.
- **Guess**: The adversary outputs a guess b' of b.

Definition 4 (AMERQCD selective-ID anonymity game)**.** *The AMERQCD selective-ID anonymity game is defined as below.*

- **Init**: The adversary $\mathcal{A}$ outputs two points $\mathbf{X}_0$ and $\mathbf{X}_1$, where it wishes to be challenged.
- **Setup**: The challenger runs the $\mathtt{Setup}(\Sigma, \mathcal{U}_{\boldsymbol{\Delta}})$ algorithm to generate $\mathbf{PK}$, $\mathbf{MK}$. It gives $\mathbf{PK}$ to the adversary, but does not divulge $\mathbf{MK}$.
- **Phase 1**: The adversary is allowed to issue decryption key queries for hyper-rectangles that do not contain $\mathbf{X}_0$ and $\mathbf{X}_1$.
- **Challenge**: The adversary submits a message $\mathbf{Msg}$. The challenger first flips a random coin b, and then encrypts $\mathbf{Msg}$ under $\mathbf{X}_b$. The ciphertext is passed to the adversary.
- **Phase 2**: Phase 1 is repeated.
- **Guess**: The adversary outputs a guess b' of b.

In either game, we define the adversary $\mathcal{A}$'s advantage as

$$\mathbf{Adv}_{\mathcal{A}}(\Sigma) = \left| \Pr[b = b'] - \frac{1}{2} \right|$$

Definition 5 (IND-sID-CPA)**.** *An AMERQCD scheme is IND-sID-CPA secure if all polynomial-time adversaries have at most a negligible advantage in the confidentiality game.*

Definition 6 (ANON-sID-CPA)**.** *An AMERQCD scheme is ANON-sID-CPA secure if all polynomial-time adversaries have at most a negligible advantage in the anonymity game.*

Definition 7 (AMERQCD security)**.** *An AMERQCD scheme is selective-ID secure if it is both IND-sID-CPA secure and ANON-sID-CPA secure.*

4 The AMERQCD (1) Scheme and Extensions

In this section, we propose an efficient AMERQCD (1) scheme based on Anonymous IBE (AIBE). Then we show that straightforward extensions of the AMERQCD (1) scheme are expensive in multiple dimensions. In Section 5, we will propose our construction of an efficient AMERQCD (D) scheme.

4.1 AMERQCD (1)

Intuition. Figure 1 illustrates the intuition behind the AMERQCD (1) construction. We build an interval tree over integers from 1 to T. In the interval tree, each leaf node represents a unique integer in $[1, T]$, and an internal node represents a range covered by all leaf nodes in the subtree rooted at that node. To encrypt a message $\mathbf{Msg}$ and a value x, we produce a portion of ciphertext for each node on the path from the root of the tree to the leaf node representing x. Hence, the ciphertext is $O(\log T)$ in length. To issue decryption keys for a range $[s, t] \subseteq [1, T]$, let $\Lambda(s, t)$ denote a set of nodes covering $[s, t]$, we issue a portion of decryption key for each node in $\Lambda(s, t)$. Since any range $[s, t] \subseteq [1, T]$ can be represented by a collection of $O(\log T)$ nodes in the tree, the decryption key has length $O(\log T)$. We now give formal definitions and then describe how to derive an AMERQCD (1) scheme from AIBE.

Definition 8 (Interval tree). *Let $\Gamma(T)$ denote a binary interval tree, where the leaf nodes correspond to integers from 1 to T, and an internal node represent the range covered by all the leaf nodes in the subtree. Naturally $\Gamma(T)$ has height $\log T$. Let $L = \log T + 1$.*

For convenience, we assume that each node in $\Gamma(T)$ has a pre-assigned unique ID. Let $\mathcal{ID}(\Gamma(T))$ represent the set of all node IDs in $\Gamma(T)$. For $ID \in \mathcal{ID}(\Gamma(T))$, if the leaf node in $\Gamma(T)$ representing $x \in [T]$ is in the subtree rooted at node ID, then we say that node ID covers x. Let $\mathsf{cv}(ID)$ denote the set of integers covered by node ID.

Definition 9 (Range as a collection of nodes). *For $1 \leq s \leq t \leq T$, let $\Lambda(s, t) \subseteq \mathcal{ID}(\Gamma(T))$ denote the smallest set of nodes in $\Gamma(T)$ such that*

$$\bigcup_{ID \in \Lambda(s,t)} \mathsf{cv}(ID) = [s, t].$$

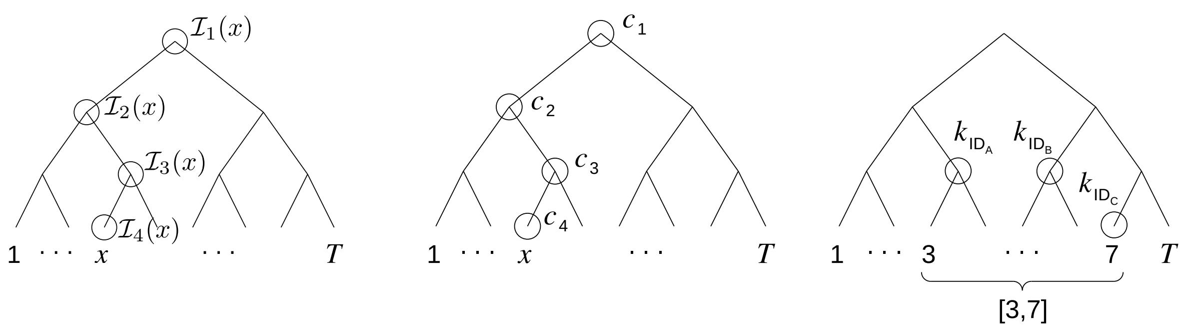
It is trivially true that $\Lambda(s, t)$ is uniquely defined and for any range $[s, t] \subseteq [1, T]$ ($1 \leq s \leq t \leq T$); moreover, $|\Lambda(s, t)| = O(\log T)$.

Definition 10 (Path to root). *Let $\mathcal{P}(x) = (\mathcal{I}_1(x), \mathcal{I}_2(x), \ldots, \mathcal{I}_L(x))$ denote the path from the root of $\Gamma(T)$ to the leaf node representing $x \in [T]$. $\mathcal{I}_l(x)$ ($1 \leq l \leq L$) represents the node on $\mathcal{P}(x)$ at depth l of the tree. By convention, we assume that the root is at depth 1.*

Proposition 1. *Given an IND-sID-CPA and ANON-sID-CPA secure Anonymous IBE (AIBE) scheme*

$$[\mathbf{Setup}^*(\Sigma), \mathbf{DeriveKey}^*(\mathbf{PK}, \mathbf{MK}, ID), \mathbf{Encrypt}^*(\mathbf{PK}, ID, \mathbf{Msg}), \mathbf{Decrypt}^*(\mathbf{PK}, \mathbf{DK}, \mathbf{C})],$$

one can implement an IND-sID-CPA and ANON-sID-CPA AMERQCD (1) scheme, with $O(\log T)$ encryption cost, ciphertext size, decryption key size and decryption cost.



(a) Path to root. (b) Ciphertext and decryption keys in AMERQCD (1).

Figure 1: An AMERQCD (1) scheme. ***(a)*** *Path from the leaf node representing* $x \in [T]$ *to the root.* $\mathcal{P}(x) = (\mathcal{I}_1(x), \mathcal{I}_2(x), \mathcal{I}_3(x), \mathcal{I}_4(x))$. ***(b)*** *Encryption to the point* $x = 3$ *and the keys released for the range* $[3, 7]$.

The construction of AMERQCD (1) from an AIBE scheme is straightforward as shown in Figure 1. To encrypt a message $\mathbf{Msg}$ under a point x, one encrypts $\mathbf{Msg}$ under all nodes along the path $\mathcal{P}(x)$ from x to the root. To release decryption keys for a range $[s, t]$, one releases the decryption keys for all node IDs in $\Lambda(s, t)$.

To be able to check whether a decryption is valid, prior to encryption, we append a message $\mathbf{Msg} \in \{0, 1\}^m$ with a series of trailing 0s, $0^{m'}$.

- $\mathbf{Setup}(\mathbf{\Sigma}, \mathbf{T})$ calls $\mathbf{Setup}^*(\Sigma)$ and outputs $\mathbf{PK}$ and $\mathbf{MK}$.
- $\mathbf{Encrypt}(\mathbf{PK}, \mathbf{x}, \mathbf{Msg})$ yields $(c_1, c_2, \ldots, c_L)$, where $c_l = \mathbf{Encrypt}^*(\mathbf{PK}, \mathcal{I}_l(x), \mathbf{Msg}||0^{m'})$ $(1 \leq l \leq L)$.
- $\mathbf{DeriveKey}(\mathbf{PK}, \mathbf{MK}, [s, t])$ yields a set $\{(l_{ID}, k_{ID}) \,|ID \in \Lambda(s, t)\}$, where

  $$k_{ID} = \mathbf{DeriveKey}^*(\mathbf{PK}, \mathbf{MK}, ID)$$

  and l_{ID} is the depth of node ID in $\Gamma(T)$. $\mathbf{DK}$ has size $O(\log T)$.
- $\mathbf{QueryDecrypt}(\mathbf{PK}, \mathbf{DK}, \mathbf{C})$, where $\mathbf{C} = (c_1, c_2, \ldots, c_L)$, is defined as below:
  1. For each $(l, k_{ID}) \in \mathbf{DK}$, compute

     $$V \leftarrow \mathbf{Decrypt}^*(\mathbf{PK}, k_{ID}, c_l).$$

     If V is of the form $\widehat{MSG}||0^{m'}$, then output $\widehat{MSG}$ as the decrypted message and exit.
  2. If for all $(l, k_{ID}) \in \mathbf{DK}$, the previous step fails to produce a plaintext, then output $\perp$.

4.2 Extension to Multi-Dimension

We explore several potential approaches to extend the AMERQCD (1) scheme to multiple dimensions.

Spatial data structures and space-filling curves. The first approach is to encode a multi-dimensional lattice using a single tree, where leaf nodes correspond to points in the lattice and internal nodes represent sets of points. There are two potential ways to achieve this: 1) use spatial data structures like a quad-tree or KD-tree; 2) use a space-filling curve to embed the D-dimension lattice into a single dimension, and then encode the single dimension using a binary interval tree.

Once we encode a D-dimensional lattice in a single tree, we can apply the same technique we used for AMERQCD (1), and construct a AMERQCD (D) scheme from an AIBE scheme. The resulting AMERQCD (D) scheme has $O(D \log T)$ encryption cost and ciphertext size. The decryption key size and decryption cost depends on how many nodes in the tree it takes to represent an arbitrary hyper-rectangle in $\mathcal{U}_\Delta$. It has been previously shown that representing an arbitrary hyper-rectangle in space can take $O(T^{D-1})$ nodes in a quad-tree (for $D \geq 2$) in the worst case [3]. For space-filling curves such as the Hilbert curve, representing an arbitrary hyper-rectangle in space can take $O(T^{D-1})$ intervals on the curve [22]. Hence, applying AIBE on top of a quad-tree or a Hilbert curve results in $\widetilde{O}(T^{D-1})$ decryption key size and decryption cost.

Naive extension of AMERQCD (1). One seemingly plausible way to extend the AMERQCD (1) scheme into D dimensions is to build D interval trees, each representing one of the D dimensions. We randomly split the plaintext into D shares, and encrypt each share in one dimension using the above-mentioned technique. To decrypt a hyper-rectangle in the lattice, one projects the hyper-rectangle onto each separate dimension, and releases the decryption keys under each dimension. This results in ciphertext size of $O(D \log T)$ and decryption key size $O(D \log T)$. Unfortunately, the security of such a scheme is flawed. As shown in Figure 2, when given the decryption keys for region R_1 and R_4, one can decrypt R_2 and R_3 as well. This attack is in spirit the collusion attack when taken in the context of Multi-dimensional Hierarchical Identity-Based Encryption(M-HIBE) schemes. A straightforward approach to securing against the collusion attack is to encrypt the ciphertext under the cross-product of D paths to the root, thus incurring $O((\log T)^D)$ ciphertext size.

We now formally describe the resulting AMERQCD (D) scheme. We assume that each tree node has a globally unique ID, and define function $\Phi(ID) := (d, l)$ to return the dimension and depth of node ID. In particular, $\Phi_1(ID) = d$ outputs the dimension of ID, and $\Phi_2(ID) = l$ outputs the depth of ID. For convenience, in the remainder of the paper, we assume that all dimensions have the same size, i.e., $T_1 = T_2 = \ldots = T_D = T$. Let $L = \log T + 1$.

Definition 11 (Hyper-rectangle as a collection of simple hyper-rectangles)**.** *For* $1 \leq s_d \leq t_d \leq T$ *(*$1 \leq d \leq D$*), suppose we have a hyper-rectangle* $\mathbf{B}(s_1, t_1, \ldots, s_D, t_D) = \{(x_1, x_2, \ldots, x_D) \big| s_d \leq x_d \leq t_d, 1 \leq d \leq D\} \subseteq \mathcal{U}_\Delta$*. Denote* $\Lambda_d(\mathbf{B}) := \Lambda(s_d, t_d)$*.*

Then, $\mathbf{B}$ *gives a collection* $\Lambda^\times(\mathbf{B})$ *of simple hyper-rectangles:*

$$\Lambda^\times(\mathbf{B}) = \Lambda_1(\mathbf{B}) \times \Lambda_2(\mathbf{B}) \times \ldots \times \Lambda_D(\mathbf{B})$$

We call a hyper-rectangle $\mathbf{B}_0 \subseteq \mathcal{U}_\Delta$ a *simple hyper-rectangle* if and only if $\forall 1 \leq d \leq D, |\Lambda_d(\mathbf{B}_0)| = 1$. In other words, a simple hyper-rectangle $\mathbf{B}_0$ can be represented by a D-tuple,

$$\Lambda^\times(\mathbf{B}_0) = \{(ID_1, ID_2, \ldots, ID_D)\},$$

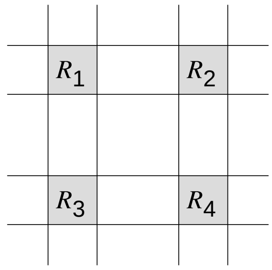


Figure 2: The collusion attack. *With decryption keys for regions R_1 and R_4, R_2 and R_3 are compromised.*

where ID_d represents the ID of a tree node in the d^{th} dimension. In particular, any point $\mathbf{X}$ in the lattice is a simple hyper-rectangle. Let $L_d = \log T_d + 1$, then a point $\mathbf{X}$ can be represented by a D-tuple $\Lambda^\times(\mathbf{X}) = \{(ID_1, ID_2, \ldots, ID_D)\}$. Because $\Lambda^\times(\mathbf{B}_0)$ is a set containing only one element, for convenience, we overload $\Lambda^\times(\mathbf{B}_0)$ to denote a D-tuple $(ID_1, ID_2, \ldots, ID_D)$ without risk of ambiguity.

It is easy to see that for any hyper-rectangle $\mathbf{B} \subseteq \mathcal{U}_\Delta$, $|\Lambda^\times(\mathbf{B})| = O((\log T)^D)$.

Definition 12 (Path to root in the d^{th} dimension)**.** *Let $\Lambda^\times(\mathbf{B}_0) = \{(ID_1, ID_2, \ldots, ID_D)\}$ represent a simple hyper-rectangle $\mathbf{B}_0$, and let l_d be the depth of node ID_d in the d^{th} tree. Then, for $1 \leq d \leq D$,*

$$\mathcal{P}_d(\mathbf{B}_0) = \left(\mathcal{I}_{(d,1)}(\mathbf{B}_0), \mathcal{I}_{(d,2)}(\mathbf{B}_0), \ldots, \mathcal{I}_{(d,l_d)}(\mathbf{B}_0)\right)$$

represents the path from node ID_d to the root in the tree $\Gamma(T_d)$ of the d^{th} dimension, where $\mathcal{I}_{(d,l)}(\mathbf{B}_0)$ ($1 \leq d \leq D$, $1 \leq l \leq l_d$) represents the node at depth l on $\mathcal{P}_d(\mathbf{B}_0)$.

Proposition 2. *Given an IND-sID-CPA and ANON-sID-CPA secure Anonymous IBE (AIBE) scheme*

$$[\mathbf{Setup}^*(\Sigma), \mathbf{DeriveKey}^*(\mathbf{PK}, \mathbf{MK}, ID), \mathbf{Encrypt}^*(\mathbf{PK}, ID, \mathbf{Msg}), \mathbf{Decrypt}^*(\mathbf{PK}, \mathbf{DK}, \mathbf{C})],$$

one can implement an IND-sID-CPA and ANON-sID-CPA AMERQCD (D) scheme, with $O((\log T)^D)$ encryption cost, ciphertext size, decryption key size and decryption cost.

The construction of AMERQCD (D) from AIBE is straightforward: To encrypt message $\mathbf{Msg}$ under point $\mathbf{X}$, one encrypts $\mathbf{Msg}$ under the cross-product of D paths to the root. To release the decryption key for a hyper-rectangle $\mathbf{B}$, one releases a decryption key for each simple hyper-rectangle in $\Lambda^\times(\mathbf{B})$.

- $\mathbf{Setup}(\mathbf{\Sigma}, \mathbf{T})$ calls $\mathbf{Setup}^*(\Sigma)$ and outputs $\mathbf{PK}$ and $\mathbf{MK}$.
- $\mathbf{Encrypt}(\mathbf{PK}, \mathbf{X}, \mathbf{Msg})$ yields ciphertext $\mathbf{C}$ of length L^D, $\mathbf{C} = \left[c_\mathbf{v} \middle| \mathbf{v} = (l_1, l_2, \ldots, l_D) \in [L]^D\right]$. Let $\mathbf{id}\left((l_1, l_2, \ldots, l_D), \mathbf{X}\right) = \left(\mathcal{I}_{(1,l_1)}(\mathbf{X}), \mathcal{I}_{(2,l_2)}(\mathbf{X}), \ldots, \mathcal{I}_{(D,l_D)}(\mathbf{X})\right)$, which is used as the identity for encryption in the underlying AIBE scheme. $c_\mathbf{v}$ is then computed as below:

$$c_\mathbf{v} = \mathbf{Encrypt}^*(\mathbf{PK}, \mathbf{id}(\mathbf{v}, \mathbf{X}), \mathbf{Msg}||0^{m'})$$

- $\mathbf{DeriveKey}(\mathbf{PK}, \mathbf{MK}, \mathbf{B})$ yields the following set of size L^D:

$$\left[[\Phi_2(ID_d)]_{d\in[L]}, k_{\mathbf{B}_0}\right]_{\Lambda^\times(\mathbf{B}_0)=\{(ID_1,ID_2,\ldots,ID_d)\}\subseteq\Lambda^\times(\mathbf{B})},$$

where $\Phi_2(ID)$ returns the depth of node ID, and $k_{\mathbf{B}_0} = \mathbf{DeriveKey}^*(\mathbf{PK}, \mathbf{MK}, \Lambda^\times(\mathbf{B}_0))$.

- $\mathbf{QueryDecrypt}(\mathbf{PK}, \mathbf{DK}, \mathbf{C})$, where $\mathbf{C} = \left[c_{\mathbf{v}} \middle| \mathbf{v} = (l_1, l_2, \ldots, l_D) \in [L]^D\right]$, is define as below:

  1. For each $(\mathbf{v}, k_{\mathbf{B}_0}) \in \mathbf{DK}$, where $\mathbf{v} = (l_1, l_2, \ldots, l_D)$, compute

  $$V \leftarrow \mathbf{Decrypt}^*(\mathbf{PK}, k_{\mathbf{B}_0}, c_{\mathbf{v}}).$$

  If V is of the form $\widehat{MSG}||0^{m'}$, then output $\widehat{MSG}$ as the decrypted message and exit.

  2. If for all $(\mathbf{v}, k_{\mathbf{B}_0}) \in \mathbf{DK}$, the previous step fails to produce a plaintext, then output $\perp$.

5 The AMERQCD (D) Scheme

In Section 4, we show that straightforward extensions of an AMERQCD (1) scheme into multiple dimensions are expensive in at least one of the following: ciphertext size, encryption cost, decryption key size and decryption cost. In this section, we detail our cryptographic construction of an efficient AMERQCD (D) scheme.

5.1 Preliminary: Bilinear Groups

A pairing is an efficiently computable, non-degenerate function, $\mathbf{e} : \mathcal{G} \times \widehat{\mathcal{G}} \rightarrow \mathcal{G}'$, satisfying the bilinear property that $\mathbf{e}(g^r, \widehat{g}^s) = \mathbf{e}(g, \widehat{g})^{rs}$. $\mathcal{G}$, $\widehat{\mathcal{G}}$ and $\mathcal{G}'$ are all groups of prime order. g, $\widehat{g}$ and $\mathbf{e}(g, \widehat{g})$ are generators of $\mathcal{G}$, $\widehat{\mathcal{G}}$ and $\mathcal{G}'$ respectively. Although our AMERQCD scheme can be constructed using asymmetric pairing, for simplicity, we describe our scheme using symmetric pairing in the remainder of the paper, i.e., $\mathcal{G} = \widehat{\mathcal{G}}$.

We name a tuple $\mathbf{G} = [p, \mathcal{G}, \mathcal{G}', g, \mathbf{e}]$ a bilinear instance, where $\mathcal{G}$ and $\mathcal{G}'$ are two cyclic groups of prime order p. We assume an efficient generation algorithm that on input a security parameter Σ, outputs $\mathbf{G} \overset{R}{\leftarrow} \text{Gen}(\Sigma)$ where $\log_2 p = \Theta(\Sigma)$.

We rely on the following complexity assumptions:

Decision BDH Assumption: The Decision Bilinear DH assumption, first used by Joux [20], later used by IBE systems [9], posits the hardness of the following problem:

Given $[g, g^{z_1}, g^{z_2}, g^{z_3}, Z] \in \mathcal{G}^4 \times \mathcal{G}'$, where exponents z_1, z_2, z_3 are picked at random from $\mathcal{Z}_p$, decide whether $Z = \mathbf{e}(g, g)^{z_1 z_2 z_3}$.

Decision Linear Assumption: The Decision Linear assumption, first proposed by Boneh, Boyen and Shacham for group signatures [7], posits the hardness of the following problem:

Given $[g, g^{z_1}, g^{z_2}, g^{z_1 z_3}, g^{z_2 z_4}, Z] \in \mathcal{G}^6$, where z_1, z_2, z_3, z_4 are picked at random from $\mathcal{Z}_p$, decide whether $Z = g^{z_3 + z_4}$.

5.2 The AMERQCD (D) Scheme

A D-dimensional lattice $\mathcal{U}_\Delta$ can be represented with a set of D trees. Assume each tree node has a globally unique ID. For convenience, we assume that an ID of a tree node has two components, $ID = [ID_j]_{j\in[2]}$. By convention, the first component $ID_1 \in \mathcal{Z}_p^*$, and is globally unique for each tree node; and the second component ID_2 is always fixed to 1. Hence, from now on, the reader should keep in mind that the notation ID and $\mathcal{I}$ have two components. In particular, when the subscript j is used in combination with ID or I, it is always used to index into the two components. This avoids ambiguity when multiple subscripts exist. We append a message $\mathbf{Msg} \in \{0,1\}^m$ with a series of trailing zeros, $0^{m'}$, prior to encryption. Assume that $\{0,1\}^{m+m'} \subseteq \mathcal{G}'$.

Intuition. To encrypt message $\mathbf{Msg}$ under point $\mathbf{X}$, for each dimension $d \in [D]$, we encrypt $\mathbf{Msg}$ along the path $\mathcal{P}_d(\mathbf{X})$ to the root. This results in $O(D\log T)$ ciphertext size. To compute decryption keys for a hyper-rectangle $\mathbf{B} \subseteq \mathcal{U}_\Delta$, we represent $\mathbf{B}$ using $\Lambda^\cup(\mathbf{B})$, a collection of nodes corresponding to disjoint intervals over different dimensions (See Definition 13). We compute a decryption key $\mathbf{DK}(ID)$ for each node with $ID \in \Lambda^\cup(\mathbf{B})$, resulting in a decryption key size of $O(D\log T)$. To prevent the collusion attack, we rerandomize decryption keys for each query, such that the decryption keys for different dimensions are tied to each other, and cannot be used in combination with decryption keys issued in another query. At the time of decryption, we can use $[\mathbf{DK}(ID)]_{ID\in\Lambda^\cup(\mathbf{B})}$ to generate a separate decryption key for each simple hyper-rectangle $\Lambda^\times(\mathbf{B}_0) \subseteq \Lambda^\times(\mathbf{B})$. We then try these decryption keys on the ciphertext to see if any one of them yields a valid decryption. To ensure the security of the scheme, we need to introduce extra degrees of freedom as represented by the subscripts $n \in \{1,2\}$ and $j \in \{1,2\}$ in the following scheme description. On one hand, our construction borrows techniques from the AHIBE construction [13] by Boyen and Waters; on the other hand, the technique we use to prevent the collusion attack can also be used to extend their AHIBE construction into multiple dimensions, resulting in a Multi-Dimension AHIBE (M-AHIBE) or a Forward Secure AHIBE (fs-AHIBE) scheme.

Definition 13 (Hyper-rectangle as a collection of nodes)**.** *A hyper-rectangle* $\mathbf{B} \subseteq \mathcal{U}_\Delta$ *gives a collection of nodes corresponding to disjoint intervals over different dimensions:*

$$\Lambda^\cup(\mathbf{B}) = \Lambda_1(\mathbf{B}) \cup \Lambda_2(\mathbf{B}) \cup \ldots \cup \Lambda_D(\mathbf{B})$$

Note that for all hyper-rectangle $\mathbf{B} \subseteq \mathcal{U}_\Delta$, $|\Lambda^\cup(\mathbf{B})| = O(DL)$.

$\mathbf{Setup}(\mathbf{\Sigma}, \mathcal{U}_\mathbf{\Delta})$ To generate public parameters and the master private key, the setup algorithm first generates a bilinear instance $\mathbf{G} = [p, \mathcal{G}, \mathcal{G}', g, \mathbf{e}] \xleftarrow{R} \text{Gen}(\Sigma)$. Then, the setup algorithm does the following.

1. Select at random the following parameters.

$$\begin{bmatrix} \omega, \\ [\alpha_{\varphi,n}, \beta_{\varphi,n}]_{\varphi=(d,l)\in[D]\times[L], n\in[2]} \\ [\theta_{\varphi,\delta}]_{\varphi=(d,l)\in[D]\times[L], \delta=(n,j)\in[2]\times[2]} \end{bmatrix} \quad \in_R \mathcal{Z}_p^* \times \mathcal{Z}_p^{*4DL} \times \mathcal{Z}_p{}^{4DL}$$

At this point, we give a brief explanation of our notation. The variable φ is used to index a tuple $(d,l) \in [D] \times [L]$, where d denotes the dimension and l denote the depth of a node in the corresponding tree. Some variables have four versions, indexed by $(n,j) \in [2] \times [2]$. At first sight, they seem to be redundant, but are in fact needed to prove the security of the scheme. In particular, we need j to prove confidentiality, and both n and j to prove anonymity.

2. Publish $\mathbf{G}$ and the following public parameters:

$$\mathbf{PK} \leftarrow \begin{bmatrix} \Omega \leftarrow \mathbf{e}(g,g)^{\omega}, \\ \left[a_{\varphi,\delta} \leftarrow g^{\alpha_{\varphi,n}\theta_{\varphi,\delta}}, \quad b_{\varphi,\delta} \leftarrow g^{\beta_{\varphi,n}\theta_{\varphi,\delta}}\right]_{\varphi=(d,l)\in[D]\times[L],\delta=(n,j)\in[2]\times[2]} \end{bmatrix} \quad \in \mathcal{G}' \times \mathcal{G}^{8DL}$$

3. Retain a master private key comprising the following elements:

$$\mathbf{MK} \leftarrow \begin{bmatrix} \widehat{\omega} \leftarrow g^{\omega}, \\ \left[\mathsf{a}_{\varphi,n} \leftarrow g^{\alpha_{\varphi,n}}, \quad \mathsf{b}_{\varphi,n} \leftarrow g^{\beta_{\varphi,n}}\right]_{\varphi=(d,l)\in[D]\times[L],n\in[2]} \\ \left[y_{\varphi,\delta} \leftarrow g^{\alpha_{\varphi,n}\beta_{\varphi,n}\theta_{\varphi,\delta}}\right]_{\varphi=(d,l)\in[D]\times[L],\delta=(n,j)\in[2]\times[2]} \end{bmatrix} \quad \in \mathcal{G}^{8DL+1}$$

$\mathbf{DeriveKey}(\mathbf{PK}, \mathbf{MK}, \mathbf{B})$ The following steps compute the decryption key for hyper-rectangle $\mathbf{B}$, given public key $\mathbf{PK}$ and master private key $\mathbf{MK}$.

1. Pick $O(D \cdot L)$ random integers,

$$\begin{bmatrix} \left[\widehat{\mu}_d\right]_{d\in[D]}, & \left[\rho_n^{(ID)}\right]_{ID\in\Lambda^{\cup}(\mathbf{B}),n\in[2]} \end{bmatrix} \in_R \mathcal{Z}_p^{2|\Lambda^{\cup}(\mathbf{B})|+D}$$

such that $\prod_{d\in[D]} \widehat{\mu}_d = \widehat{\omega}$.

Observe that we release a portion of the decryption key for each node in $\Lambda^{\cup}(\mathcal{B})$, as opposed to for each hyper-rectangle in $\Lambda^{\times}(\mathcal{B})$. In this way, the size of the private key is $O(DL)$, instead of $O(L^D)$.

2. Compute and release the following decryption key:

$$\mathbf{DK} \leftarrow \left[\ \Phi(ID), \mathbf{DK}(ID)\ \right]_{ID\in\Lambda^{\cup}(\mathbf{B})} \quad \in \left([D] \times [L] \times \mathcal{G}^5\right)^{|\Lambda^{\cup}(\mathbf{B})|}$$

where $\Phi(ID)$ extracts the dimension and depth of node ID, and $\mathbf{DK}(ID)$ is defined as below:

$$\mathbf{DK}(ID) \leftarrow \begin{bmatrix} \widehat{\mu}_d \cdot \prod_{\delta=(n,j)\in[2]\times[2]} \left({y_{\varphi,\delta}}^{ID_j}\right)^{\rho_n^{(ID)}}, \\ \left[{\mathsf{a}_{\varphi,n}}^{-\rho_n^{(ID)}}, {\mathsf{b}_{\varphi,n}}^{-\rho_n^{(ID)}}\right]_{n\in[2]} \end{bmatrix}$$

where $\varphi = (d,l) = \Phi(ID)$.

Encrypt$(\mathbf{PK}, \mathbf{X}, \mathbf{Msg})$ The following algorithm encrypts a message $\mathbf{Msg}$ under point $\mathbf{X}$.

1. Select $2DL + 1$ random integers $r \in \mathcal{Z}_p$, $[r_{\varphi,n}]_{\varphi=(d,l)\in[D]\times[L],n\in[2]} \in \mathcal{Z}_p^{2DL}$.
2. Compute and output the following ciphertext:

$$\left[\begin{array}{l} (\mathbf{Msg}||0^{m'}) \cdot \Omega^{-r}, \quad g^r, \\ \left[\prod_{\substack{j\in[2],\\ \delta=(n,j)}} \left(b_{\varphi,\delta}{}^{\mathcal{I}_{\varphi,j}}\right)^{r_{\varphi,n}}, \prod_{\substack{j\in[2],\\ \delta=(n,j)}} \left(a_{\varphi,\delta}{}^{\mathcal{I}_{\varphi,j}}\right)^{r-r_{\varphi,n}} \right]_{\varphi=(d,l)\in[D]\times[L],n\in[2]} \end{array}\right] \in \mathcal{G}' \times \mathcal{G}^{4DL+1}$$

   where $\mathcal{I}_\varphi = \mathcal{I}_\varphi(\mathbf{X})$.

   For each dimension d and depth l, we create a portion of the ciphertext corresponding to the node in the d^{th} tree at depth l, on the path $\mathcal{P}_d(\mathbf{X})$ to the root.

QueryDecrypt$(\mathbf{PK}, \mathbf{DK}, \mathbf{C})$ Given ciphertext $\mathbf{C} = \left(c, c_0, \left[c^{(\mathrm{I})}_{\varphi,n}, c^{(\mathrm{II})}_{\varphi,n}\right]_{\varphi=(d,l)\in[D]\times[L],n\in[2]}\right)$ and decryption key $\mathbf{DK}$ for hyper-rectangle $\mathbf{B}$, the following algorithm iterates through the simple hyper-rectangles in $\Lambda^\times(\mathbf{B})$ and attempts to decrypt the ciphertext under each simple hyper-rectangle.

1. For each simple hyper-rectangle $\Lambda^\times(\mathbf{B}_0) = \{(ID_1, ID_2, \ldots, ID_D)\} \subseteq \Lambda^\times(\mathbf{B})$,
   (1) Let $\mathbf{DK}(ID_d) = \left(\mathsf{k}^{(\mathrm{O})}_{ID_d}, [\mathsf{k}^{(\mathrm{I})}_{ID_d,1}, \mathsf{k}^{(\mathrm{II})}_{ID_d,1}], [\mathsf{k}^{(\mathrm{I})}_{ID_d,2}, \mathsf{k}^{(\mathrm{II})}_{ID_d,2}]\right)$ represent the element in $\mathbf{DK}$ for ID_d, where $d \in [D]$.
   (2) Compute the following:

$$\mathsf{k}_0 \leftarrow \prod_{d\in[D]} \mathsf{k}^{(\mathrm{O})}_{ID_d};$$
$$V \leftarrow c \cdot \mathbf{e}(c_0, \mathsf{k}_0) \cdot \prod_{\substack{d\in[D],n\in[2],\\ \varphi_d=\Phi(ID_d)}} \left(\mathbf{e}(c^{(\mathrm{I})}_{\varphi_d,n}, \mathsf{k}^{(\mathrm{I})}_{ID_d,n}) \cdot \mathbf{e}(c^{(\mathrm{II})}_{\varphi_d,n}, \mathsf{k}^{(\mathrm{II})}_{ID_d,n})\right)$$

   If V is of the form $\widehat{\mathbf{Msg}}||0^{m'}$, then output $\widehat{\mathbf{Msg}}$ as the decrypted plaintext and exit.

2. If for all simple hyper-rectangles in $\Lambda^\times(\mathbf{B})$, the previous step fails to produce the plaintext, then output $\perp$.

When done naively, the above **QueryDecrpty** algorithm takes $O(D(\log T)^D)$ time. However, if one saves intermediate results, it can be done in $O((\log T)^D)$ time with $O(D \log T)$ storage.

6 Consistency, Security

In this section, we formally state the consistency and security of the above AMERQCD (D) construction. We give a high level intuition about the security proof, and provide the detailed proof in Appendix C.

Theorem 6.1 (Internal consistency)**.** *The above defined AMERQCD scheme satisfies the consistency requirement posed by Equation (1).*

We say that an AMERQCD scheme is (τ, q, ϵ) secure if any adversary making q range queries for decryption keys, cannot have more than ϵ advantage within time τ.

Theorem 6.2 (Confidentiality)**.** *Suppose* $\mathbf{G}$ *satisfies the* (τ, ϵ) *D-BDH assumption, then the above defined AMERQCD scheme is* (τ', q, ϵ) *IND-sID-CPA secure, where* $\tau' < \tau - \Theta(qD \log T)$.

Theorem 6.3 (Anonymity)**.** *Suppose* $\mathbf{G}$ *satisfies the* (τ, ϵ) *D-Linear assumption, then the above defined AMERQCD scheme is* (τ', q, ϵ') *ANON-sID-CPA secure, where* $\tau' < \tau - \Theta(qD \log T)$*, and* $\epsilon' = (2D \log T + 1)(\epsilon + 1/p)$.

In particular, $\Theta(qD \log T)$ comes from the fact that the simulator needs $O(D \log T)$ time to compute the decryption key for each hyper-rectangle queried. The $2D \log T + 1$ loss factor in ϵ' comes from the hybrid argument we use to prove anonymity, and additive $1/p$ comes from the probability that bad events happen in the simulation so that the simulator has to abort.

Our proof techniques are similar to that presented in the AHIBE paper [13]. However, since we do not require the delegation property of HIBE/AHIBE schemes, we do not need as many degrees of freedom as required by the AHIBE scheme. In the proof, our simulator inherits parameters specified by the D-BDH/D-Linear instance, hence, it has incomplete information about the master private key. In order to simulate the key query process, the simulator leverages a dimension d along which $\mathbf{X}^*$ does not overlap with the queried hyper-rectangle. The simulator uses randomness in the d^{th} dimension to cancel out the unknown parameters inherited from the D-BDH/D-Linear instance. In comparison, the anonymity proof is more complicated than the confidentiality proof, because it involves a hybrid argument containing $2DL$ steps. In step (d_1, l_1, n_1) of the hybrid argument, $y_{\varphi_1,(n_1,j)}$ ($\varphi_1 = (d_1, l_1), j \in [2]$) in the master key contains unknown parameters inherited from the D-Linear instance. Therefore, we need to condition on the relative position between $\mathbf{X}^*$ and the (d_1, l_1) in question. We provide detailed proof of Theorems 6.1, 6.2 and 6.3 in Appendix C.1, C.2 and C.3 respectively.

7 Extensions

7.1 Adaptive-ID Security

Our scheme is provably secure in the selective-ID model. A stronger notion of security is adaptive-ID security (also known as *full* security), i.e., the adversary does not have to commit ahead of time which point in the lattice to attack. Specifically, in both the confidentiality and anonymity games, the adversary $\mathcal{A}$ does not commit to any particular point $\mathbf{X}^*$ in the **Init** phase. **Phase 1** is the same as before except that there is no restriction on the hyper-rectangles the adversary can query. In the **Challenge** stage, the adversary picks a point $\mathbf{X}^*$, which is not contained in any of the hyper-rectangles queried in **Phase 1**. In **Phase 2**, the adversary can query hyper-rectangles that do not contain $\mathbf{X}^*$. After this point, the game proceeds in the same way as before. We present the formal definition for AMERQCD adaptive-ID security in Appendix B.

Previous research has shown that IBE schemes secure in the selective-ID sense can be converted to schemes fully secure [4, 16, 28, 23] with some loss in security. In particular, Boneh and Boyen prove the following theorem:

Theorem 7.1 ([4]). *A (t, q, ϵ)-selective identity secure IBE system (*IND-sID-CPA*) that admits N distinct identities is also a $(t, q, N\epsilon)$-fully secure IBE (*IND-ID-CPA*).*

This technique can be applied to our case to achieve full confidentiality and anonymity. In our case, the scheme admits $N = T^D$ identities and hence that would be the loss factor in security.

7.2 Applications to Closely Related Problems

Our AMERQCD scheme can be applied to two closely related problems which we call Multiple-Attribute Encryption with Ranged Conditional Decryption (MERCD) and Anonymous Multi-Attribute Encryption with Range Query (AMERQ) respectively. In Appendix A, we define these two problems, describe scenarios where each might be useful, and explain their relationship with AMERQCD.

8 Conclusion

We define the concept of Anonymous Multi-Attribute Encryption with Range Query and Conditional Decryption (AMERQCD), and give a cryptographic construction. When compared with other potential approaches, our scheme is efficient in terms of encryption cost, ciphertext size, decryption key size, and decryption cost. We reduce the security of our AMERQCD scheme to the intractability of the D-BDH and D-Linear problem in bilinear groups.

References

[1] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In *Advances in Cryptology - Proceedings of CRYPTO '05*, pages 205–222. Springer-Verlag, August 2005.

[2] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[3] Paul E. Black. Algorithms and theory of computation handbook, CRC Press LLC, “quadtree complexity theorem”. Dictionary of Algorithms and Data Structures [online], ed., U.S. National Institute of Standards and Technology. Available from: http://www.nist.gov/dads/HTML/quadtreecplx.html.

[4] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT*, pages 223–238, 2004.

[5] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *CRYPTO*, pages 443–459, 2004.

[6] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *Proceedings of Eurocrypt 2005*, LNCS. Springer, 2005.

[7] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO*, pages 41–55, 2004.

[8] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *EUROCRYPT*, pages 506–522, 2004.

[9] Dan Boneh and Matthew Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[10] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *EUROCRYPT*, 2006.

[11] Dan Boneh and Brent Waters. A collusion resistant broadcast, trace and revoke system. `http://crypto.stanford.edu/~dabo/abstracts/tr.html`.

[12] Dan Boneh and Brent Waters. Complex queries on encrypted data. 2006.

[13] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *CRYPTO*, 2006.

[14] Alexandre Brandwajn. A model of a time sharing virtual memory system solved using equivalence and decomposition methods. *Acta Inf.*, 4:11–47, 1974.

[15] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT*, pages 255–271, 2003.

[16] Sanjit Chatterjee and Palash Sarkar. Trading time for space: Towards an efficient IBE scheme with short(er) public parameters in the standard model. In *Proceedings of ICISC*, 2004.

[17] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, pages 360–363, 2001.

[18] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In *ASIACRYPT '02: Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security*, pages 548–566, London, UK, 2002. Springer-Verlag.

[19] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In *EUROCRYPT '02: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pages 466–481, London, UK, 2002. Springer-Verlag.

[20] Antoine Joux. A one round protocol for tripartite Diffie-Hellman. In *ANTS-IV: Proceedings of the 4th International Symposium on Algorithmic Number Theory*, pages 385–394, London, UK, 2000. Springer-Verlag.

[21] B. B. Mandelbrot. Harnessing the Peano monster curves. The Fractal Geometry of Nature, Ch. 7, 1982.

[22] Bongki Moon, H. v. Jagadish, Christos Faloutsos, and Joel H. Saltz. Analysis of the clustering properties of the Hilbert space-filling curve. *IEEE Transactions on Knowledge and Data Engineering*, 13(1):124–141, 2001.

[23] David Naccache. Secure and *practical* identity-based encryption. Cryptology ePrint Archive, Report 2005/369, 2005. `http://eprint.iacr.org/`.

[24] Hans Sagan. Space-filling curves. Springer-Verlag, 1994.

[25] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.

[26] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 47–53, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[27] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *SP '00: Proceedings of the 2000 IEEE Symposium on Security and Privacy*, page 44, Washington, DC, USA, 2000. IEEE Computer Society.

[28] Brent Waters. Efficient identity-based encryption without random oracles. In *Proceedings of Eurocrypt*, 2005.

[29] Brent R. Waters, Dirk Balfanz, Glenn Durfee, and D. K. Smetters. Building an encrypted and searchable audit log. In *Proceedings of Network and Distributed System Security Symposium 2004 (NDSS'04)*, San Diego, CA, February 2004.

[30] Danfeng Yao, Nelly Fazio, Yevgeniy Dodis, and Anna Lysyanskaya. ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 354–363, New York, NY, USA, 2004. ACM Press.

A Applications to Closely Related Problems

In this section, we describe two closely related problems, MERCD and AMERQ. We describe specific scenarios where each may be useful. It is easy to show that AMERQCD implies both MERCD and AMERQ.

A.1 Multiple-Attribute Encryption with Ranged Conditional Decryption

In a *partially encrypted database*, each data entry consists of both *public attributes* and *private attributes*. The public attributes are stored in clear-text, and the private attributes are stored in encrypted format. Multiple-Attribute Encryption with Ranged Conditional Decryption (MERCD) allows a master key owner to issue decryption keys for a hyper-rectangle such that it is possible to decrypt the private attributes of an entry *iff* the public attributes of that entry fall within the hyper-rectangle.

Partially encrypted database are useful in various scenarios. In trace-driven network intrusion detection, network traces are often partially encrypted before they are released. For instance, due to privacy concerns, source/destination addresses and payload are encrypted, but port numbers, packet length and other fields in the packet header are in clear-text. One can apply statistical methods to find anomalies in the network traces. When an anomaly is found, one wishes to obtain capabilities from an authority to decrypt the source/destination addresses and payload of suspicious packets whose port numbers or payload length fall within a certain range. Waters et al. study a related problem of searching encrypted audit logs [29].

We can formally define MERCD problem in a way similar to the AMERQCD problem. The differences between MERCD and AMERQCD are: 1) In MERCD, by looking at the public attributes of a database entry, one learns under which point in space a message is encrypted. Hence, at decryption, one does not have to iterate through all potential points and try with the corresponding decryption key; 2) The security definition for MERCD has only the confidentiality game. Anonymity is not required, since the point under which a message is encrypted is stated by the public attributes of a database entry.

It is easy to see that AMERQCD implies MERCD. Meanwhile, MERCD can be solved using an M-HIBE scheme. In particular, the HIBE scheme proposed by Boneh Boyen and Goh [6] can be extended to multiple dimensions in a collusion-resistant manner. Using the resulting M-HIBE scheme for MERCD results in $O(D)$ ciphertext size, $O(D \log T)$ encryption cost, $O(D \log T)$ decryption key size, and $O(D \log T)$ decryption cost. Since MERCD does not require the delegation property needed by M-HIBE, it is possible to derive a more efficient MERCD scheme by removing key delegation from the M-HIBE scheme.

A.2 Anonymous Multi-Attribute Encryption with Range Query

AMERQCD allows the untrusted database to perform range query on encrypted data entries, and decrypt only ciphertexts matching the range query. By contrast, in some scenarios, a data owner may want to retrieve from the database encrypted entries satisfying certain range queries, without having the database decrypt these matching entries. Instead, the data owner performs the final decryption after retrieving the entries. We call this problem AMERQ.

The AMERQ problem can be formally defined in a similar way as the AMERQCD problem. The differences between AMERQ and AMERQCD are: 1) In AMERQ, $\mathtt{Setup}$ returns a public key $\mathbf{PK}$, a master private key $\mathbf{MK}$ for generating query capabilities, and a decryption private key $\mathbf{DK}$ for decryption; 2) Instead of generating a decryption key, $\mathbf{DeriveKey}$ generates a query key $\mathbf{QK}$; 3) The $\mathbf{QueryDecrypt}(\mathbf{PK}, \mathbf{DK}, \mathbf{C})$ algorithm is replaced by a separate $\mathbf{Query}(\mathbf{PK}, \mathbf{QK}, \mathbf{C})$ algorithm and a $\mathbf{Decrypt}(\mathbf{PK}, \mathbf{DK}, \mathbf{C})$ algorithm. For any message $\mathrm{Msg} \in \mathcal{M}$, hyper-rectangle $\mathbf{B} \subseteq \mathcal{U}_\Delta$, point $\mathbf{X} \in \mathcal{U}_\Delta$, let $\mathbf{C} = \mathbf{Encrypt}(\mathbf{PK}, \mathbf{X}, \mathbf{Msg})$, $\mathbf{QK} = \mathbf{DeriveKey}(\mathbf{PK}, \mathbf{MK}, \mathbf{B})$, then

- $\mathbf{Query}(\mathbf{PK}, \mathbf{QK}, \mathbf{C})$ returns 1 if $\mathbf{X} \in \mathbf{B}$; otherwise, it returns 0 with high probability.
- $\mathbf{Decrypt}(\mathbf{PK}, \mathbf{DK}, \mathbf{C})$ returns the plaintext Msg.

Corollary 1. *Let AMERQCD* $= (\mathbf{Setup}^*, \mathbf{Encrypt}^*, \mathbf{DeriveKey}^*, \mathbf{QueryDecrypt}^*)$ *be an IND-sID-CPA and ANON-sID-CPA secure AMERQCD scheme, let* $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *be an IND-CPA secure public key encryption scheme independent of the AMERQCD scheme.* $\mathcal{K}, \mathcal{E}, \mathcal{D}$ *represent the key generation, encryption and decryption algorithms respectively. Given AMERQCD and* $\mathcal{AE}$*, an IND-sID-CPA and ANON-sID-CPA secure AMERQ scheme* $(\mathbf{Setup}, \mathbf{Encrypt}, \mathbf{DeriveKey}, \mathbf{Query}, \mathbf{Decrypt})$ *can be constructed as below:*

- $\mathbf{Setup}$ runs $(pk_0, dk_0) \leftarrow \mathcal{K}$ and $(pk_1, mk) \leftarrow \mathbf{Setup}^*$; and outputs $\mathbf{PK} \leftarrow (pk_0, pk_1)$, $\mathbf{DK} \leftarrow dk_0$ and $\mathbf{MK} \leftarrow mk$.
- $\mathbf{Encrypt}((pk_0, pk_1), \mathbf{X}, \mathbf{Msg}) := \left[\mathcal{E}(pk_0, \mathbf{Msg}), \mathbf{Encrypt}^*(pk_1, \mathbf{X}, 0^{m'})\right]$;
- $\mathbf{DeriveKey}((pk_0, pk_1), \mathbf{MK}, \mathbf{B}) := \mathbf{DeriveKey}^*(pk_1, \mathbf{MK}, \mathbf{B})$;
- $\mathbf{Query}((pk_0, pk_1), \mathbf{QK}, (c_0, c_1))$ calls $\mathbf{QueryDecrypt}^*(pk_1, \mathbf{QK}, c_1)$, and outputs 1 if $\mathbf{QueryDecrypt}^*$ decrypts to $0^{m'}$, 0 otherwise.
- $\mathbf{Decrypt}((pk_0, pk_1), \mathbf{DK}, (c_0, c_1)) := \mathcal{D}(pk_0, \mathbf{DK}, c_0)$.

In particular, in the above construction, $\mathbf{Encrypt}^*(pk_1, \mathbf{X}, 0^{m'})$ provides a searchable label which enables range query. The security of the AMERQ scheme relies on the IND-CPA security of $\mathcal{AE}$ and the ANON-sID-CPA security of the underlying AMERQCD scheme. The searchable label idea is similar to that adopted by Song et al. [27] and Boneh et al. [8] in their encryption schemes with keyword-based search. We can take the idea one step further, and label arbitrary content with a searchable label. The label encrypts multiple integer attributes and can be used to perform range query on the encrypted attributes.

B Definition of Adaptive-ID Security

Definition 14 (AMERQCD adaptive-ID confidentiality game)**.** *The AMERQCD adaptive-ID confidentiality game is defined as below.*

- **Setup**: The challenger runs the $\mathtt{Setup}(\Sigma, \mathcal{U}_{\boldsymbol{\Delta}})$ algorithm to generate $\mathbf{PK}$, $\mathbf{MK}$. It gives $\mathbf{PK}$ to the adversary, but does not divulge $\mathbf{MK}$.
- **Phase 1**: The adversary is allowed to issue decryption key queries for hyper-rectangles in the D-dimensional lattice.
- **Challenge**: The adversary submits two equal length messages $\mathbf{Msg}_0$ and $\mathbf{Msg}_1$, and a point $\mathbf{X}^*$. $\mathbf{X}^*$ must not be contained in any hyper-rectangle queried in **Phase 1**. The challenger flips a random coin, b, and encrypts $\mathbf{Msg}_b$ under $\mathbf{X}^*$. The ciphertext is passed to the adversary.
- **Phase 2**: Phase 1 is repeated, in addition, any hyper-rectangle queried in **Phase 2** must not contain $\mathbf{X}^*$ as specified in the **Challenge** phase.
- **Guess**: The adversary outputs a guess b' of b.

Definition 15 (AMERQCD adaptive-ID anonymity game)**.** *The AMERQCD adaptive-ID anonymity game is defined as below.*

- **Setup**: The challenger runs the $\mathtt{Setup}(\Sigma, \mathcal{U}_{\boldsymbol{\Delta}})$ algorithm to generate $\mathbf{PK}$, $\mathbf{MK}$. It gives $\mathbf{PK}$ to the adversary, but does not divulge $\mathbf{MK}$.
- **Phase 1**: The adversary is allowed to issue decryption key queries for hyper-rectangles in the D-dimensional lattice.
- **Challenge**: The adversary submits two points $\mathbf{X}_0$ and $\mathbf{X}_1$, and a message $\mathbf{Msg}$. $\mathbf{X}_0$ and $\mathbf{X}_1$ must not be contained in any hyper-rectangle queried in **Phase 1**. The challenger flips a random coin, b, and encrypts $\mathbf{Msg}$ under $\mathbf{X}_b$. The ciphertext is passed to the adversary.
- **Phase 2**: Phase 1 is repeated, in addition, any hyper-rectangle queried in **Phase 2** must not contain $\mathbf{X}_0$ and $\mathbf{X}_1$ specified in the **Challenge** phase.
- **Guess**: The adversary outputs a guess b' of b.

C Proof of Consistency and Security

C.1 Proof: Consistency

Proof of Theorem 6.1:

Let $\mathbf{C} = \left(c, c_0, \left[c^{(\mathrm{I})}_{\varphi,n}, c^{(\mathrm{II})}_{\varphi,n}\right]_{\varphi=(d,l)\in[D]\times[L], n\in[2]}\right)$ be the encryption of $\mathbf{Msg}$ on point $\mathbf{X}$. Let $\Lambda^{\times}(\mathbf{B}_0) = \{(ID_1, ID_2, \ldots, ID_D)\} \subseteq \Lambda^{\times}(\mathbf{B})$ be the current simple hyper-rectangle under decryption. Let $\varphi_d = \Phi(ID_d)$ $(d \in [D])$.

If $\mathbf{X} \in \mathbf{B}_0$, then for all $d \in [D]$, $\mathcal{I}_{\varphi_d}(\mathbf{X}) = \mathcal{I}_{\varphi_d}(\mathbf{B_0}) = ID_d$. For simplicity, let $\lambda(x) = \mathbf{e}(g, g)^x$. Now decryption for $\mathbf{B_0}$ proceeds as follows:

$$\begin{aligned}
\mathsf{k}_0 &= \prod_{d\in[D]} \mathsf{k}^{(\mathrm{O})}_{ID_d} \\
&= \prod_{d\in[D]} \left(\widehat{\mu}_d \cdot \prod_{\delta=(n,j)\in[2]\times[2]} \left(y_{\varphi_d,\delta}{}^{ID_{d,j}}\right)^{\rho_n^{(ID_d)}}\right) \\
&= \widehat{\omega} \prod_{\substack{d\in[D],\\ \delta=(n,j)\in[2]\times[2]}} \left(y_{\varphi_d,\delta}{}^{ID_{d,j}}\right)^{\rho_n^{(ID_d)}}
\end{aligned}$$

$$
\begin{aligned}
V =&(\mathbf{Msg}||0^{m'}) \cdot \Omega^{-r} \cdot \mathbf{e}\left(g^r, \widehat{\omega} \prod_{\substack{d\in[D],\\ \delta=(n,j)\in[2]\times[2]}} \left(y_{\varphi_d,\delta}{}^{ID_{d,j}}\right)^{\rho_n^{(ID_d)}}\right)\\
&\cdot \prod_{\substack{d\in[D],\\ n\in[2]}} \left(\mathbf{e}\left(\mathsf{a}_{\varphi_d,n}{}^{-\rho_n^{(ID_d)}}, \prod_{\substack{j\in[2],\\ \delta=(n,j)}} \left(b_{\varphi_d,\delta}{}^{ID_{d,j}}\right)^{r_{\varphi_d,n}}\right) \cdot \mathbf{e}\left(\mathsf{b}_{\varphi_d,n}{}^{-\rho_n^{(ID_d)}}, \prod_{\substack{j\in[2],\\ \delta=(n,j)}} \left(a_{\varphi_d,\delta}{}^{ID_{d,j}}\right)^{r-r_{\varphi_d,n}}\right)\right)\\
=&(\mathbf{Msg}||0^{m'}) \cdot \Omega^{-r} \cdot \mathbf{e}\left(g^r, \widehat{\omega}\right) \cdot \lambda\left(r \cdot \sum_{\substack{d\in[D],\\ \delta=(n,j)\in[2]\times[2]}} \alpha_{\varphi_d,n}\beta_{\varphi_d,n}\theta_{\varphi_d,\delta}\rho_n^{(ID_d)} ID_{d,j}\right)\\
&\cdot \lambda\left(\sum_{\substack{d\in[D],\\ \delta=(n,j)\in[2]\times[2]}} \alpha_{\varphi_d,n}(-\rho_n^{(ID_d)})\beta_{\varphi_d,n}\theta_{\varphi_d,\delta} ID_{d,j} r_{\varphi_d,n}\right)\\
&\cdot \lambda\left(\sum_{\substack{d\in[D],\\ \delta=(n,j)\in[2]\times[2]}} \beta_{\varphi_d,n}(-\rho_n^{(ID_d)})\alpha_{\varphi_d,n}\theta_{\varphi_d,\delta} ID_{d,j}(r - r_{\varphi_d,n})\right)\\
=&(\mathbf{Msg}||0^{m'}) \cdot \Omega^{-r} \cdot \mathbf{e}(g^r, \widehat{\omega}) \cdot \lambda\left(r \cdot \sum_{\substack{d\in[D],\\ \delta=(n,j)\in[2]\times[2]}} \alpha_{\varphi_d,n}\beta_{\varphi_d,n}\theta_{\varphi_d,\delta}\rho_n^{(ID_d)} ID_{d,j}\right)\\
&\cdot \lambda\left(r \cdot \sum_{\substack{d\in[D],\\ \delta=(n,j)\in[2]\times[2]}} \alpha_{\varphi_d,n}\beta_{\varphi_d,n}\theta_{\varphi_d,\delta}(-\rho_n^{(ID_d)}) ID_{d,j}\right)\\
=&\mathbf{Msg}||0^{m'}.
\end{aligned}
$$

Else if $\mathbf{X} \notin \mathbf{B}_0$, $\mathcal{I}_{\varphi_d}(\mathbf{X}) \neq \mathcal{I}_{\varphi_d}(\mathbf{B_0}) = ID_d$, $d \in [D]$. Hence decryption yields

$$
\begin{aligned}
V &= (\mathbf{Msg}||0^{m'}) \cdot \frac{\lambda\left(r \cdot \sum_{\substack{d\in[D],\\ \delta=(n,j)\in[2]\times[2]}} \alpha_{\varphi_d,n}\beta_{\varphi_d,n}\theta_{\varphi_d,\delta}\rho_n^{(ID_d)} ID_{d,j}\right)}{\lambda\left(r \cdot \sum_{\substack{d\in[D],\\ \delta=(n,j)\in[2]\times[2]}} \alpha_{\varphi_d,n}\beta_{\varphi_d,n}\theta_{\varphi_d,\delta}\rho_n^{(ID_d)} \mathcal{I}_{\varphi,j}(\mathbf{X})\right)}\\
&= (\mathbf{Msg}||0^{m'}) \cdot Q^r
\end{aligned}
$$

where

$$Q = \lambda \left(\sum_{\substack{d\in[D],\\ \delta=(n,j)\in[2]\times[2]}} \alpha_{\varphi_d,n}\beta_{\varphi_d,n}\theta_{\varphi_d,\delta}\rho_n^{(ID_d)} ID_{d,j} - \sum_{\substack{d\in[D],\\ \delta=(n,j)\in[2]\times[2]}} \alpha_{\varphi_d,n}\beta_{\varphi_d,n}\theta_{\varphi_d,\delta}\rho_n^{(ID_d)} \mathcal{I}_{\varphi,j}(\mathbf{X}) \right)$$

With probability $1-1/p$, $Q \neq 1$, and the ciphertext is distributed uniformly at random in $\mathcal{G}'$. Hence the probability that V is of the form $\widehat{\mathbf{Msg}}||0^{m'}$ is less than $\frac{1}{p} + \frac{1}{2^{m'}}$.

■

C.2 Proof: Confidentiality

Proof of Theorem 6.2:

We reduce the semantic security of AMERQCD to the hardness of the D-BDH problem. Let $[g, g_1, g_2, g_3, Z]$ denote the D-BDH instance supplied to the simulator, $\mathcal{B}$, where $g_1 = g^{z_1}$, $g_2 = g^{z_2}$, $g_3 = g^{z_3}$, the simulator's task is to decide whether or not $Z = \mathbf{e}(g,g)^{z_1z_2z_3}$. And to do this, the simulator leverages an AMERQCD IND-sID-CPA adversary, $\mathcal{A}$.

We describe a reduction such that if $Z = \mathbf{e}(g,g)^{z_1z_2z_3}$, the simulator produces a valid ciphertext; otherwise, the first term c in the ciphertext is random. Hence, if the adversary could break the confidentiality of the scheme, the simulator would be able to solve the D-BDH problem.

Init: The adversary selects a point $\mathbf{X}^* \in \mathcal{U}_\Delta$ that it wishes to attack. For $\varphi \in [D] \times [L]$, define $\mathcal{I}^*_\varphi = \mathcal{I}_\varphi(\mathbf{X}^*)$.

Setup: To create public and private parameters, the simulator does the following:

1. Pick at random

$$\begin{bmatrix} [\alpha_{\varphi,n}, \beta_{\varphi,n}]_{\varphi=(d,l)\in[D]\times[L],n\in[2]} \\ [\theta_{\varphi,\delta}]_{\varphi=(d,l)\in[D]\times[L],\delta=(n,j)\in[2]\times[2]} \\ \left[\bar{\theta}_{\varphi,\delta}\right]_{\varphi=(d,l)\in[D]\times[L],\delta=(n,j)\in[2]\times[2]} \end{bmatrix} \in_R \mathcal{Z}_p^{*4DL} \times \mathcal{Z}_p^{4DL} \times \mathcal{Z}_p^{*4DL}$$

   subject to the constraint that

$$\left[\sum_{j\in[2],\delta=(n,j)} \bar{\theta}_{\varphi,\delta}\mathcal{I}^*_{\varphi,j} = 0 \right]_{\varphi=(d,l)\in[D]\times[L],n\in[2]}$$

   where $\mathcal{I}^*_\varphi = \mathcal{I}_\varphi(\mathbf{X}^*)$.

2. Release the following public parameters to the adversary.

$$\begin{bmatrix} \Omega \leftarrow \mathbf{e}(g_1, g_2), \\ \left[a_{\varphi,\delta} \leftarrow \left(g^{\theta_{\varphi,\delta}} g_1^{\bar{\theta}_{\varphi,\delta}}\right)^{\alpha_{\varphi,n}}, \quad b_{\varphi,\delta} \leftarrow \left(g^{\theta_{\varphi,\delta}} g_1^{\bar{\theta}_{\varphi,\delta}}\right)^{\beta_{\varphi,n}} \right]_{\varphi=(d,l)\in[D]\times[L],\delta=(n,j)\in[2]\times[2]} \end{bmatrix}$$

   Note that this posits that $\omega = z_1 z_2$; in addition, both ω and $\widehat{\omega}$ are both unknown to the simulator.

3. Compute what it can of the master key.

$$\begin{bmatrix} \left[\mathsf{a}_{\varphi,n} \leftarrow g^{\alpha_{\varphi,n}}, \quad \mathsf{b}_{\varphi,n} \leftarrow g^{\beta_{\varphi,n}}\right]_{\varphi=(d,l)\in[D]\times[L],n\in[2]} \\ \left[y_{\varphi,\delta} \leftarrow \left(g^{\theta_{\varphi,\delta}} {g_1}^{\bar{\theta}_{\varphi,\delta}}\right)^{\alpha_{\varphi,n}\beta_{\varphi,n}},\right]_{\varphi=(d,l)\in[D]\times[L],\delta=(n,j)\in[2]\times[2]} \end{bmatrix}$$

Portion $\widehat{\omega}$ of the master key is unknown to the simulator.

Phase 1: Suppose the adversary makes a decryption key query for the hyper-rectangle $\mathbf{B}(s_1, t_1, s_2, t_2, \ldots, s_D, t_D)$. Since $\mathbf{B}$ does not contain $\mathbf{X}^*$, there exists a dimension $d_0 \in [D]$ such that $x^*_{d_0} \notin [s_{d_0}, t_{d_0}]$, where $x^*_{d_0}$ is $\mathbf{X}^*$ projected onto the d_0^{th} dimension. Hence, there exists a dimension $d_0 \in [D]$, such that for all $ID \in \Lambda_{d_0}(\mathbf{B})$, $ID \neq \mathcal{I}^*_{\varphi}$, where $\varphi = (d_0, l) = \Phi(ID)$. We say that $\mathbf{X}^*$ does not overlap with $\mathbf{B}$ in dimension d_0. The simulator now does the following:

1. Pick d_0 such that $\mathbf{X}^*$ does not overlap with $\mathbf{B}$ in dimension d_0. Let $n_0 = 1$.

2. Pick the following numbers at random:

$$\begin{bmatrix} [\mu_d]_{d\in[D]} \\ \left[\widetilde{\rho}_{n_0}^{(ID)}\right]_{ID\in\Lambda_{d_0}(\mathbf{B})} \\ \left[\rho_n^{(ID)}\right]_{ID\in\Lambda_{d_0}(\mathbf{B}),n\neq n_0} \\ \left[\rho_n^{(ID)}\right]_{ID\in\Lambda^{\cup}(\mathbf{B})-\Lambda_{d_0}(\mathbf{B}),n\in[2]} \end{bmatrix} \in_R \mathcal{Z}_p^D \times \mathcal{Z}_p^{2|\Lambda^{\cup}(\mathbf{B})|}$$

   subject to the constraint that $\sum_{d\in[D]} \mu_d = 0$.

3. For all $ID \in \Lambda^{\cup}(\mathbf{B}) - \Lambda_{d_0}(\mathbf{B})$, let $\mathbf{DK}(ID) = \left(\mathsf{k}_{ID}^{(\mathrm{O})}, [\mathsf{k}_{ID,1}^{(\mathrm{I})}, \mathsf{k}_{ID,1}^{(\mathrm{II})}], [\mathsf{k}_{ID,2}^{(\mathrm{I})}, \mathsf{k}_{ID,2}^{(\mathrm{II})}]\right)$ represent the element in $\mathbf{DK}$ for ID, let $\varphi = (d, l) = \Phi(ID)$ where $d \neq d_0$, compute and release $\mathbf{DK}(ID)$ as below:

$$\begin{bmatrix} \mathsf{k}_{ID}^{(\mathrm{O})} \leftarrow g^{\mu_d} \cdot \prod_{\delta=(n,j)\in[2]\times[2]} \left({y_{\varphi,\delta}}^{ID_j}\right)^{\rho_n^{(ID)}}, \\ \left[\mathsf{k}_{ID,n}^{(\mathrm{I})} \leftarrow {\mathsf{a}_{\varphi,n}}^{-\rho_n^{(ID)}}, \mathsf{k}_{ID,n}^{(\mathrm{II})} \leftarrow {\mathsf{b}_{\varphi,n}}^{-\rho_n^{(ID)}}\right]_{n\in[2]} \end{bmatrix}$$

4. For all $ID \in \Lambda_{d_0}(\mathbf{B})$, let $\varphi_0 = (d_0, l) = \Phi(ID)$, compute and release $\mathbf{DK}(ID)$ as below:

$$\begin{bmatrix} \mathsf{k}_{ID}^{(\mathrm{O})} \leftarrow \widehat{\omega} g^{\mu_{d_0}} \cdot \prod_{\delta=(n,j)\in[2]\times[2]} \left({y_{\varphi,\delta}}^{ID_j}\right)^{\rho_n^{(ID)}}, \\ \left[\mathsf{k}_{ID,n}^{(\mathrm{I})} \leftarrow {\mathsf{a}_{\varphi_0,n}}^{-\rho_n^{(ID)}}, \mathsf{k}_{ID,n}^{(\mathrm{II})} \leftarrow {\mathsf{b}_{\varphi_0,n}}^{-\rho_n^{(ID)}}\right]_{n\in[2]} \end{bmatrix}$$

   where

$$\rho_{n_0}^{(ID)} = \widetilde{\rho}_{n_0}^{(ID)} - \frac{z_2}{\alpha_{\varphi_0,n_0}\beta_{\varphi_0,n_0}\bar{\Theta}_{\varphi_0,n_0}} \tag{2}$$

$$\bar{\Theta}_{\varphi_0,n_0} = \sum_{j\in[2],\delta_0=(n_0,j)} \bar{\theta}_{\varphi_0,\delta_0} ID_j \neq 0.$$

This ensures that $\rho_{n_0}^{(ID)}$ is distributed uniformly at random in $\mathcal{Z}_p$. And since $\sum_{j\in[2],\delta_0=(n_0,j)} \bar{\theta}_{\varphi_0,\delta_0}\mathcal{I}^*_{\varphi_0,j} = 0$; moreover, the simulator has picked d_0 such that $ID \neq \mathcal{I}^*_{\varphi_0}$, we then have $\bar{\Theta}_{\varphi_0,n_0} \neq 0$. Although the simulator does not know $\rho_{n_0}^{(ID)}$ (since it does not know z_2), it can compute ${\mathsf{a}_{\varphi_0,n_0}}^{-\rho_{n_0}^{(ID)}}$ and ${\mathsf{b}_{\varphi_0,n_0}}^{-\rho_{n_0}^{(ID)}}$ given g^{z_2}. Since the simulator does not know $\widehat{\omega}$, we now explain how to compute $\mathsf{k}_{ID}^{(\mathrm{O})}$. The simulator rewrites the equation for $\mathsf{k}_{ID}^{(\mathrm{O})}$ as

$$\begin{aligned}
\mathsf{k}_{ID}^{(\mathrm{O})} =& \widehat{\omega} g^{\mu_{d_0}} \cdot \prod_{\delta=(n,j)\in[2]\times[2]} \left({y_{\varphi_0,\delta}}^{ID_j}\right)^{\rho_n^{(ID)}} \\
=& \left[g^{\mu_{d_0}} \cdot \prod_{\delta=(2,j)\in\{2\}\times[2]} \left({y_{\varphi_0,\delta}}^{ID_j}\right)^{\rho_2^{(ID)}} \right] \cdot \widehat{\omega} \cdot \prod_{\delta=(1,j)\in\{1\}\times[2]} \left({y_{\varphi_0,\delta}}^{ID_j}\right)^{\rho_1^{(ID)}}
\end{aligned}$$

Let

$$\Psi = g^{\mu_{d_0}} \cdot \prod_{\delta=(2,j)\in\{2\}\times[2]} \left({y_{\varphi_0,\delta}}^{ID_j}\right)^{\rho_2^{(ID)}}$$

Then

$$\begin{aligned}
\mathsf{k}_{ID}^{(\mathrm{O})} &= \Psi \cdot \widehat{\omega} \cdot \prod_{j\in[2],\delta_0=(n_0,j)} \left({y_{\varphi_0,\delta_0}}^{ID_j}\right)^{\rho_{n_0}^{(ID)}} \\
&= \Psi \cdot g^{z_1 z_2} \cdot \prod_{j\in[2],\delta_0=(n_0,j)} \left(g^{\alpha_{\varphi_0,n_0}\beta_{\varphi_0,n_0}(\theta_{\varphi_0,\delta_0}+z_1\bar{\theta}_{\varphi_0,\delta_0})}\right)^{ID_j \rho_{n_0}^{(ID)}}
\end{aligned}$$

The simulator can compute part Ψ because it possesses all necessary parameters required to compute it.

Although the simulator cannot directly compute the value of $\rho_{n_0}^{(ID)}$ (since it does not know z_2), it is capable of computing $\mathsf{k}_{ID}^{(\mathrm{O})}$ given g^{z_1} and g^{z_2}; since if we rewrite $\mathsf{k}_{ID}^{(\mathrm{O})}$ as below, we can see that the exponent only contains z_1 and z_2 to the first degree.

$$\begin{aligned}
\mathsf{k}_{ID}^{(\mathrm{O})} =& \Psi \cdot g^{z_1 z_2} \cdot \prod_{j\in[2],\delta_0=(n_0,j)} g^{-z_1 z_2 \rho_{n_0}^{(ID)} \bar{\theta}_{\varphi_0,\delta_0} ID_j / \bar{\Theta}_{\varphi_0,n_0}} \\
& \cdot \prod_{j\in[2],\delta_0=(n_0,j)} \left[g^{\alpha_{\varphi_0,n_0}\beta_{\varphi_0,n_0}\theta_{\varphi_0,\delta_0} ID_j \rho_{n_0}^{ID}} \cdot g^{\alpha_{\varphi_0,n_0}\beta_{\varphi_0,n_0} z_1 \bar{\theta}_{\varphi_0,\delta_0} ID_j \widetilde{\rho}_{n_0}^{(ID)}} \right] \\
=& \Psi \cdot \prod_{j\in[2],\delta_0=(n_0,j)} \left[g^{\alpha_{\varphi_0,n_0}\beta_{\varphi_0,n_0}\theta_{\varphi_0,\delta_0} ID_j \rho_{n_0}^{ID}} \cdot g^{\alpha_{\varphi_0,n_0}\beta_{\varphi_0,n_0} z_1 \bar{\theta}_{\varphi_0,\delta_0} ID_j \widetilde{\rho}_{n_0}^{(ID)}} \right]
\end{aligned}$$

Challenge: The adversary gives the simulator two messages Msg_0 and Msg_1. The simulator picks a random bit b, and encrypts Msg_b under point $\mathbf{X}^*$ as below:

1. Pick random integers $[r_{\varphi,n}]_{\varphi=(d,l)\in[D]\times[L],n\in[2]} \in \mathcal{Z}_p^{2DL}$.

2. Compute and release the following as the ciphertext.

$$\left[\begin{array}{l} (\mathbf{Msg}_b||0^{m'}) \cdot Z^{-1}, g_3, \\ \left[\prod_{\substack{j\in[2],\\ \delta=(n,j)}} g^{r_{\varphi,n}\beta_{\varphi,n}\theta_{\varphi,\delta}\mathcal{I}^*_{\varphi,j}}, \prod_{\substack{j\in[2],\\ \delta=(n,j)}} (g_3 \cdot g^{-r_{\varphi,n}})^{\alpha_{\varphi,n}\theta_{\varphi,\delta}\mathcal{I}^*_{\varphi,j}} \right]_{\varphi=(d,l)\in[D]\times[L],n\in[2]} \end{array}\right]$$

Note that this implies that $r = z_3$; and if $Z = \mathbf{e}(g,g)^{z_1z_2z_3}$, it is easy to verify that the ciphertext is well-formed, due to the fact that $\left[\sum_{j\in[2],\delta=(n,j)} \bar{\theta}_{\varphi,\delta}\mathcal{I}^*_{\varphi,j} = 0\right]_{\varphi=(d,l)\in[D]\times[L],n\in[2]}$. On the other hand, if Z is a random number, then the first term c in the ciphertext is random and independent of the remaining terms.

Phase 2: Phase 1 is repeated.

Guess: When the adversary outputs a guess b' of b, the simulator outputs 1 if $b' = b$ and 0 otherwise, in answer to the D-BDH instance. ■

C.3 Proof: Anonymity

In Definition 4 of the selective-ID anonymity game, the challenger flips a random coin b in the **Challenge** phase. An equivalent definition is where the challenger flips the coin b in the **Setup** phase before running the $\mathbf{Setup}(\mathbf{\Sigma}, \mathcal{U}_{\mathbf{\Delta}})$ algorithm. This new definition can be further tranlated into a real-or-random version which we will use in the following proof of anonymity. In the real-or-random game, the adversary commits to only one point $\mathbf{X}^*$ in the **Init** phase; any of its subsequent range queries must not contain $\mathbf{X}^*$; in the **Challenge** phase, the challenger either returns a faithful encryption of Msg under $\mathbf{X}^*$ or a completely random ciphertext; and the adversary's job is to distinguish between these two worlds. It is easy to verify that the above real-or-random definition implies the selective-ID anonymity definition as stated in Definition 4 [13].

The proof of anonymity is carried out in $2DL$ steps using a hybrid argument. To do this, we define the following games, where $*$ represents a number distributed uniformly at random from the appropriate group.

$$\begin{aligned}
\mathcal{W}_{real}: &\quad \text{The challenge ciphertext is } \left(c, c_0, [c^{(\mathrm{I})}_{(1,1),1}, c^{(\mathrm{II})}_{(1,1),1}], \ldots, [c^{(\mathrm{I})}_{(D,L),2}, c^{(\mathrm{II})}_{(D,L),2}]\right);\\
\mathcal{W}_0: &\quad \text{The challenge ciphertext is } \left(*, c_0, [c^{(\mathrm{I})}_{(1,1),1}, c^{(\mathrm{II})}_{(1,1),1}], \ldots, [c^{(\mathrm{I})}_{(D,L),2}, c^{(\mathrm{II})}_{(D,L),2}]\right);\\
\mathcal{W}_{1,1,1}: &\quad \text{The challenge ciphertext is } \left(*, c_0, [*,*], [c^{(\mathrm{I})}_{(1,1),2}, c^{(\mathrm{II})}_{(1,1),2}], \ldots, [c^{(\mathrm{I})}_{(D,L),2}, c^{(\mathrm{II})}_{(D,L),2}]\right);\\
\mathcal{W}_{1,1,2}: &\quad \text{The challenge ciphertext is } \left(*, c_0, [*,*], [*,*], [c^{(\mathrm{I})}_{(1,2),1}, c^{(\mathrm{II})}_{(1,2),1}], \ldots, [c^{(\mathrm{I})}_{(D,L),2}, c^{(\mathrm{II})}_{(D,L),2}]\right);\\
&\ldots\\
\mathcal{W}_{D,L,1}: &\quad \text{The challenge ciphertext is } \left(*, c_0, [*,*], [*,*], \ldots, [*,*], [c^{(\mathrm{I})}_{(D,L),2}, c^{(\mathrm{II})}_{(D,L),2}]\right);\\
\mathcal{W}_{D,L,2}: &\quad \text{The challenge ciphertext is } (*, c_0, [*,*], [*,*], \ldots, [*,*], [*,*]).
\end{aligned}$$

In step (d,l,n) of the hybrid argument, we show that $\mathcal{W}_{d,l,n}$ is computationally indistinguishable from the previous world. Note that the transition from $\mathcal{W}_{real}$ to $\mathcal{W}_0$ is the standard concept of semantic security, and has been proved in the previous section. In addition, $\mathcal{W}_{D,L,2}$ is computationally indistinguishable from a completely random ciphertext, hence is anonymous.

We reduce the anonymity of our AMERQCD scheme to the hardness of the D-Linear problem. We rewrite the D-Linear problem as given $[g, g^{z_1}, g^{z_2}, Y, g^{z_2 z_4}, g^{z_3+z_4}] \in \mathcal{G}^6$, where z_1, z_2, z_3, z_4 are picked at random from $\mathcal{Z}_p$, decide whether $Y = g^{z_1+z_3}$. It is easy to show that this is equivalent to the original D-Linear problem. For convenience, let $g_1 = g^{z_1}$, $g_2 = g^{z_2}$, $g_{24}^{\times} = g^{z_2 z_4}$, $g_{34}^{+} = g^{z_3+z_4}$.

Without loss of generality, we show only how to prove step (d_1, l_1, n_1) of the hybrid argument.

Lemma C.1. *Suppose* $\mathbf{G}$ *satisfies the* (τ, ϵ) *D-Linear assumption, then no adversary making* q *decryption key queries, within time* $\tau - \Theta(qD \log T)$*, can distinguish between* $\mathcal{W}_{d_1,l_1,n_1}$ *and the preceding game with more than* $\epsilon + 1/p$ *probability.*

Proof of Lemma C.1: Let $\varphi_1 = (d_1, l_1)$. We describe a reduction such that if $Y = g^{z_1+z_3}$, then the simulator produces a ciphertext in which the block $[c^{(\mathrm{I})}_{(d_1,l_1),n_1}, c^{(\mathrm{II})}_{(d_1,l_1),n_1}]$ is well-formed; otherwise, if Y is picked at random, the block is random as well. Hence, if the adversary can distinguish between the two scenarios, the simulator can solve the D-Linear problem.

Init: The adversary selects a point $\mathbf{X}^*$ in space that it wishes to attack. Define $\mathcal{I}^*_{\varphi,j} = \mathcal{I}_{\varphi,j}(\mathbf{X}^*)$.

Setup: To create public and private parameters, the simulator does the following:

1. Pick the following parameters at random:

$$\begin{bmatrix} \omega, \\ [\alpha_{\varphi,n}, \beta_{\varphi,n}]_{\varphi=(d,l)\in[D]\times[L], n\in[2], (\varphi,n)\neq(\varphi_1,n_1)}, \\ [\theta_{\varphi,\delta}]_{\varphi=(d,l)\in[D]\times[L], \delta=(n,j)\in[2]\times[2]}, \\ [\bar{\theta}_{\varphi,\delta}]_{\varphi=(d,l)\in[D]\times[L], \delta=(n,j)\in[2]\times[2], (\varphi,n)\neq(\varphi_1,n_1)} \end{bmatrix} \in_R \mathcal{Z}_p^* \times \mathcal{Z}_p^{*\,4DL-1} \times \mathcal{Z}_p^{\,4DL} \times \mathcal{Z}_p^{*\,4DL-1}$$

subject to the constraint that

$$\left[\sum_{j\in[2], \delta=(n,j)} \bar{\theta}_{\varphi,\delta} \mathcal{I}^*_{\varphi,j} = 0\right]_{\varphi=(d,l)\in[D]\times[L], n\in[2], (\varphi,n)\neq(\varphi_1,n_1)}$$

where $\mathcal{I}^*_{\varphi,j} = \mathcal{I}_{\varphi,j}(\mathbf{X}^*)$.

In addition, later in Equation (5), we will need that $\sum_{j\in[2]} \mathcal{I}^*_{\varphi_1,j}\theta_{\varphi_1,(n_1,j)} \neq 0$. Hence, the simulator simply aborts if it happens to pick $\theta_{\varphi_1,(n_1,j)}$'s such that $\sum_{j\in[2]} \mathcal{I}^*_{\varphi_1,j}\theta_{\varphi_1,(n_1,j)} = 0$. Note that this happens with probability $1/p$, and this explains why the $1/p$ additive factor exists in the adversary's advantage in Lemma C.1.

2. Compute and release to the adversary the following public parameters:

$$\left[\begin{array}{l} \Omega \leftarrow \mathbf{e}(g,g)^{\omega}, \\ \left[a_{\varphi_1,\delta_1} \leftarrow {g_1}^{\theta_{\varphi_1,\delta_1}}, b_{\varphi_1,\delta_1} \leftarrow {g_2}^{\theta_{\varphi_1,\delta_1}}\right]_{\delta_1=(n_1,j)\in\{n_1\}\times[2]}, \\ \left[a_{\varphi,\delta} \leftarrow \left(g^{\theta_{\varphi,\delta}}{g_1}^{\bar{\theta}_{\varphi,\delta}}\right)^{\alpha_{\varphi,n}}, b_{\varphi,\delta} \leftarrow \left(g^{\theta_{\varphi,\delta}}{g_1}^{\bar{\theta}_{\varphi,\delta}}\right)^{\beta_{\varphi,n}}\right]_{\varphi=(d,l)\in[D]\times[L],\delta=(n,j)\in[2]\times[2],(\varphi,n)\neq(\varphi_1,n_1)} \end{array}\right]$$

This posits that $\alpha_{\varphi_1,n_1} = z_1$, $\beta_{\varphi_1,n_1} = z_2$, both of which are unknown to the simulator.

3. Compute what it can of the private key:

$$\left[\begin{array}{l} \widehat{\omega} \leftarrow g^{\omega}, \mathsf{a}_{\varphi_1,n_1} \leftarrow g_1, \mathsf{b}_{\varphi_1,n_1} \leftarrow g_2, \\ \left[\mathsf{a}_{\varphi,n} \leftarrow g^{\alpha_{\varphi,n}}, \mathsf{b}_{\varphi,n} \leftarrow g^{\beta_{\varphi,n}}\right]_{\varphi=(d,l)\in[D]\times[L],n\in[2],(\varphi,n)\neq(\varphi_1,n_1)}, \\ \left[y_{\varphi,\delta} \leftarrow \left(g^{\theta_{\varphi,\delta}}{g_1}^{\bar{\theta}_{\varphi,\delta}}\right)^{\alpha_{\varphi,n}\beta_{\varphi,n}}\right]_{\varphi=(d,l)\in[D]\times[L],\delta=(n,j)\in[2]\times[2],(\varphi,n)\neq(\varphi_1,n_1)} \end{array}\right]$$

Note that the simulator does not know $y_{\varphi_1,(n_1,j)}$.

The following lemma shows that even if we do not know the parameters z_1, z_2, $y_{\varphi_1,(n_1,j)}$, we can still compute certain terms efficiently.

Lemma C.2. *In step (d_1,l_1,n_1) of the hybrid arguement, let $\varphi_1 = (d_1,l_1)$. Suppose we are given $(d_2,l_2,n_2) \neq (d_1,l_1,n_1)$, and let $\varphi_2 = (d_2,l_2)$. Suppose ID_1 and ID_2 are nodes such that $\Phi(ID_1) = \varphi_1$ and $\Phi(ID_2) = \varphi_2$ and $ID_2 \neq \mathcal{I}^*_{\varphi_2}$. Moreover, suppose we are given $\rho_1 \in \mathcal{Z}_p$. Then, even though the simulator does know know $y_{\varphi_1,(n_1,j)}$, it can efficiently generate the following term, such that the its resulting distribution is the same as when ρ_2 is picked uniformly at random.*

$$\prod_{j\in[2]}\left(y^{ID_{1,j}}_{\varphi_1,(n_1,j)}\right)^{\rho_1} \cdot \prod_{j\in[2]}\left(y^{ID_{2,j}}_{\varphi_2,(n_2,j)}\right)^{\rho_2}, \tag{3}$$

Moreover, the following two terms can also be computed efficiently

$$\mathsf{a}^{-\rho_2}_{\varphi_2,n_2}, \mathsf{b}^{-\rho_2}_{\varphi_2,n_2}. \tag{4}$$

Proof. Let $\alpha = \alpha_{\varphi_2,n_2}$, $\beta = \beta_{\varphi_2,n_2}$. For $i \in [2]$, $j \in [2]$, let ${\theta'}_{i,j} = \theta_{\varphi_i,(n_1,j)}$; $\overline{\theta'}_{2,j} = \overline{\theta}_{\varphi_2,(n_2,j)}$.

Define for $i \in [2]$, $\Theta_i = \sum_{j\in[2]} \theta'_{i,j} ID_{i,j}$ and $\overline{\Theta}_2 = \sum_{j\in[2]} \overline{\theta}'_{2,j} ID_{2,j}$

Recall our convention that the first component (indexed by subscript j) of an ID is globally unique for each tree node, and the second component of an ID is fixed to 1. Since $ID_2 \neq \mathcal{I}^*_{\varphi_2}$, and $\sum_{j\in[2]} \overline{\theta}'_{2,j}\mathcal{I}^*_{\varphi_2,j} = 0$ and $\overline{\theta}'_{2,1} \neq 0$,

$$\overline{\Theta}_2 = \sum_{j\in[2]} \overline{\theta}'_{2,j} ID_{2,j} \neq 0$$

First, the simulator pick ρ uniformly at random and define

$$\rho_2 = \rho - \frac{z_2 \rho_1 \Theta_1}{\alpha\beta\overline{\Theta}_2}.$$

Observe that ρ_2 is distributed uniformly, but we cannot compute ρ_2 efficiently because we do not know z_2. However, since we know g^{z_2}, we can compute g^{ρ_2} efficiently. Hence, it follows that we can compute the two terms in (4) efficiently in the following way.

$$\mathsf{a}_{\varphi_2,n_2}^{-\rho_2} = (g^{\rho_2})^{-\alpha}, \mathsf{b}_{\varphi_2,n_2}^{-\rho_2} = (g^{\rho_2})^{-\beta}.$$

It remains to show how to compute the term in (3). Rewrite (3) as below:

$$\prod_{j\in[2]} (y_{\varphi_1,(n_1,j)}^{ID_{1,j}})^{\rho_1} \cdot \prod_{j\in[2]} (y_{\varphi_2,(n_2,j)}^{ID_{2,j}})^{\rho_2} = \prod_{j\in[2]} g^{z_1 z_2 \theta'_{1,j}\, ID_{1,j}\, \rho_1} \cdot \prod_{j\in[2]} g^{\alpha\beta(\theta'_{2,j}+z_1\overline{\theta'}_{2,j}) ID_{2,j}\, \rho_2}$$
$$= g^{z_1 z_2 \rho_1 \Theta_1 + \alpha\beta(\Theta_2 + z_1\overline{\Theta}_2)(\rho - z_2\rho_1\Theta_1/\alpha\beta\overline{\Theta}_2)} = g^{\alpha\beta\Theta_2\rho} \cdot (g^{z_1})^{\alpha\beta\overline{\Theta}_2\rho} \cdot (g^{z_2})^{-\rho_1\Theta_1\Theta_2/\overline{\Theta}_2},$$

which can be computed efficiently given g^{z_1} and g^{z_2}. ■

Phase 1: Suppose the adversary makes a decryption query for the hyper-rectangle $\mathbf{B}(s_1, t_1, \ldots, s_D, t_D)$. Since $\mathbf{B}$ does not contain $\mathbf{X}^*$, there exists a dimension $d_0 \in [D]$ such that $x_{d_0}^* \notin [s_{d_0}, t_{d_0}]$, where $x_{d_0}^*$ is $\mathbf{X}^*$ projected onto the d_0^{th} dimension. Hence, exactly one of the following cases must be true:

Case 1: For all $ID \in \Lambda_{d_1}(\mathbf{B})$ such that $\Phi(ID) = \varphi_1$, $ID \neq \mathcal{I}_{\varphi_1}(\mathbf{X}^*)$.

Case 2: There exists $ID \in \Lambda_{d_1}(\mathbf{B})$ such that $\Phi(ID) = \varphi_1$ and $ID = \mathcal{I}_{\varphi_1}(\mathbf{X}^*)$. Note that in this case, for all $ID' \in \Lambda_{d_1}(\mathbf{B})$ such that $ID' \neq ID$, $ID' \neq \mathcal{I}_{\varphi'}(\mathbf{X}^*)$, where $\varphi' = \Phi(ID')$; moreover, there exists a dimension d_0, such that for all $ID_0 \in \Lambda_{d_0}(\mathbf{B})$, $ID_0 \neq \mathcal{I}_{\varphi_0}(\mathbf{X}^*)$, where $\varphi_0 = \Phi(ID_0)$.

Figure 3 illustrates the above two cases with a 2-dimensional example. We now explain how the simulator generates the decryption key in each of the above cases.

Case 1: (a) Pick at random $[\widehat{\mu}_d]_{d\in[D]} \in_R \mathcal{G}^D$, such that $\prod_{d\in[D]} \widehat{\mu}_d = \widehat{\omega}$.

(b) For each $ID \in \Lambda^{\cup}(\mathbf{B})$ where $\varphi := \Phi(ID) \neq \varphi_1$, pick at random $\left[\rho_n^{(ID)}\right]_{n\in[2]}$. Let $\mathbf{DK}(ID) = \left(\mathsf{k}_{ID}^{(\mathrm{O})}, [\mathsf{k}_{ID,1}^{(\mathrm{I})}, \mathsf{k}_{ID,1}^{(\mathrm{II})}], [\mathsf{k}_{ID,2}^{(\mathrm{I})}, \mathsf{k}_{ID,2}^{(\mathrm{II})}]\right)$ represent the element in $\mathbf{DK}$ for ID, compute and release $\mathbf{DK}(ID)$ as below:

$$\left[\begin{array}{l} \mathsf{k}_{ID}^{(\mathrm{O})} \leftarrow \widehat{\mu}_d \cdot \prod_{\delta=(n,j)\in[2]\times[2]} \left(y_{\varphi,\delta}{}^{ID_j}\right)^{\rho_n^{(ID)}}, \\ \left[\mathsf{k}_{ID,n}^{(\mathrm{I})} \leftarrow \mathsf{a}_{\varphi,n}{}^{-\rho_n^{(ID)}}, \mathsf{k}_{ID,n}^{(\mathrm{II})} \leftarrow \mathsf{b}_{\varphi,n}{}^{-\rho_n^{(ID)}}\right]_{n\in[2]} \end{array}\right]$$

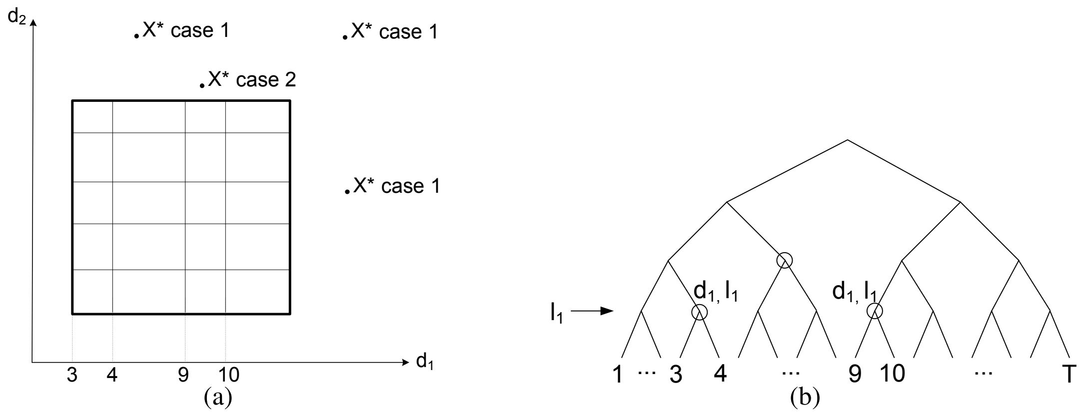


Figure 3: A 2-dimensional example: Relative position between $\mathbf{X}^*$ and the queried hyper-rectangle. ***(a)*** *Each small rectangle shown is a simple rectangle. Along dimension d_1, ranges* $[3, 4]$ *and* $[9, 10]$ *correspond to nodes at level l_1.* ***(b)*** *The interval tree corresponding to dimension d_1.*

(c) For each $ID \in \Lambda^{\cup}(\mathbf{B})$ such that $\Phi(ID) = \varphi_1$, the simulator can compute the following $\mathbf{DK}(ID)$ efficiently:

$$\begin{bmatrix} \mathsf{k}_{ID}^{(\mathrm{O})} \leftarrow \widehat{\mu}_{d_1} \cdot \prod_{\delta=(n,j)\in[2]\times[2]} \left(y_{\varphi_1,\delta}^{ID_j}\right)^{\rho_n^{(ID)}}, \\ \left[\mathsf{k}_{ID,n}^{(\mathrm{I})} \leftarrow \mathsf{a}_{\varphi_1,n}^{-\rho_n^{(ID)}}, \mathsf{k}_{ID,n}^{(\mathrm{II})} \leftarrow \mathsf{b}_{\varphi_1,n}^{-\rho_n^{(ID)}}\right]_{n\in[2]} \end{bmatrix}$$

Since the simulator does not know $y_{\varphi_1,(n_1,j)}$, hence, it needs to use Lemma C.2 to generate $\mathbf{DK}(ID)$. Let $n' \neq n_1$. To apply Lemma C.2, the simulator first picks at random $\rho_{n_1}^{(ID)}$, and rewrites $\mathsf{k}_{ID}^{(\mathrm{O})}$ as

$$\begin{aligned} \mathsf{k}_{ID}^{(\mathrm{O})} &= \widehat{\mu}_{d_1} \cdot \prod_{\delta=(n,j)\in[2]\times[2]} \left(y_{\varphi_1,\delta}^{ID_j}\right)^{\rho_n^{(ID)}} \\ &= \widehat{\mu}_{d_1} \cdot \prod_{\delta=(n_1,j)\in\{n_1\}\times[2]} \left(y_{\varphi_1,\delta}^{ID_j}\right)^{\rho_{n_1}^{(ID)}} \cdot \prod_{\delta=(n',j)\in\{n'\}\times[2]} \left(y_{\varphi_1,\delta}^{ID_j}\right)^{\rho_{n'}^{(ID)}} \end{aligned}$$

since $ID \neq \mathcal{I}_{\varphi_1}(\mathbf{X}^*)$, the simulator can apply Lemma C.2 by substituting (d_2, l_2, n_2) in the lemma with (d_1, l_1, n'), and ρ_1 with $\rho_{n_1}^{(ID)}$; in addition, both ID_1 and ID_2 in the lemma are substituted with ID.

Case 2: (a) Pick at random $[\mu_d]_{d\in[D]} \in_R \mathcal{Z}_p$ such that $\sum_{d\in[D]} \mu_d = \omega$.

(b) For each $ID \in \Lambda^{\cup}(\mathbf{B}) - \Lambda_{d_0}(\mathbf{B}) - \Lambda_{d_1}(\mathbf{B})$ where $\varphi := \Phi(ID) = (d, l)$, $d \neq d_0$ and $d \neq d_1$, pick at random $\left[\rho_n^{(ID)}\right]_{n\in[2]}$. Let $\mathbf{DK}(ID) = \left(\mathsf{k}_{ID}^{(\mathrm{O})}, [\mathsf{k}_{ID,1}^{(\mathrm{I})}, \mathsf{k}_{ID,1}^{(\mathrm{II})}], [\mathsf{k}_{ID,2}^{(\mathrm{I})}, \mathsf{k}_{ID,2}^{(\mathrm{II})}]\right)$

represent the element in $\mathbf{DK}$ for ID, compute and release $\mathbf{DK}(ID)$ as below:

$$\begin{bmatrix} \mathsf{k}_{ID}^{(\mathrm{O})} \leftarrow g^{\mu_d} \cdot \prod_{\delta=(n,j)\in[2]\times[2]} \left(y_{\varphi,\delta}{}^{ID_j}\right)^{\rho_n^{(ID)}}, \\ \left[\mathsf{k}_{ID,n}^{(\mathrm{I})} \leftarrow \mathsf{a}_{\varphi,n}{}^{-\rho_n^{(ID)}}, \mathsf{k}_{ID,n}^{(\mathrm{II})} \leftarrow \mathsf{b}_{\varphi,n}{}^{-\rho_n^{(ID)}}\right]_{n\in[2]} \end{bmatrix}$$

(c) Let $\overline{ID} \in \Lambda_{d_1}(\mathbf{B})$ and $\overline{ID} = \mathcal{I}_{\varphi_1}(\mathbf{X}^*)$. There exists exactly one such $\overline{ID}$. The simulator picks at random $\rho_{n_1}^{(\overline{ID})} \in_R \mathcal{Z}_p$. Define $\Upsilon = \prod_{\delta_1=(n_1,j)\in\{n_1\}\times[2]} y_{\varphi_1,\delta_1}{}^{\overline{ID}_j \rho_{n_1}^{(\overline{ID})}}$.

(d) For each $ID \in \Lambda_{d_0}(\mathbf{B})$ where $\varphi_0 = (d_0, l) := \Phi(ID)$, compute and release $\mathbf{DK}(ID)$:

$$\begin{bmatrix} \mathsf{k}_{ID}^{(\mathrm{O})} \leftarrow g^{\mu_{d_0}} \cdot \Upsilon \cdot \prod_{\delta=(n,j)\in[2]\times[2]} \left(y_{\varphi_0,\delta}{}^{ID_j}\right)^{\rho_n^{(ID)}}, \\ \left[\mathsf{k}_{ID,n}^{(\mathrm{I})} \leftarrow \mathsf{a}_{\varphi_0,n}{}^{-\rho_n^{(ID)}}, \mathsf{k}_{ID,n}^{(\mathrm{II})} \leftarrow \mathsf{b}_{\varphi_0,n}{}^{-\rho_n^{(ID)}}\right]_{n\in[2]} \end{bmatrix}$$

This implies that $\widehat{\mu}_{d_0} = g^{\mu_{d_0}} \cdot \Upsilon$. Note that Υ cannot be computed efficiently, as the simulator does not know $y_{\varphi_1,(n_1,j)}$. However, since $ID \neq \mathcal{I}_{\varphi_0}(\mathbf{X}^*)$, the simulator can apply Lemma C.2 by substituting (d_2, l_2, n_2) in the lemma with $(d_0, l, 1)$, ρ_1 with $\rho_{n_1}^{(\overline{ID})}$, ID_1 with $\overline{ID}$, and ID_2 with ID. The remaining terms in $\mathsf{k}_{ID}^{(\mathrm{O})}$ can be computed efficiently.

(e) For each $ID \in \Lambda_{d_1}(\mathbf{B})$ where $\varphi_1' = (d_1, l) := \Phi(ID) \neq \varphi_1$, compute and release $\mathbf{DK}(ID)$:

$$\begin{bmatrix} \mathsf{k}_{ID}^{(\mathrm{O})} \leftarrow g^{\mu_{d_1}} \cdot \Upsilon^{-1} \cdot \prod_{\delta=(n,j)\in[2]\times[2]} \left(y_{\varphi_1',\delta}{}^{ID_j}\right)^{\rho_n^{(ID)}}, \\ \left[\mathsf{k}_{ID,n}^{(\mathrm{I})} \leftarrow \mathsf{a}_{\varphi_1',n}{}^{-\rho_n^{(ID)}}, \mathsf{k}_{ID,n}^{(\mathrm{II})} \leftarrow \mathsf{b}_{\varphi_1',n}{}^{-\rho_n^{(ID)}}\right]_{n\in[2]} \end{bmatrix}$$

This implies that $\widehat{\mu}_{d_1} = g^{\mu_{d_1}} \cdot \Upsilon^{-1}$. Note that Υ^{-1} cannot be computed efficiently, as the simulator does not know $y_{\varphi_1,(n_1,j)}$. However, since $ID \neq \mathcal{I}_{\varphi_1'}(\mathbf{X}^*)$, the simulator can apply Lemma C.2, by substituting (d_2, l_2, n_2) in the lemma with $(d_1, l, 1)$, ρ_1 with $-\rho_{n_1}^{(\overline{ID})}$, ID_1 with $\overline{ID}$, and ID_2 with ID. The remaining terms in $\mathsf{k}_{ID}^{(\mathrm{O})}$ can be computed efficiently.

(f) For $\overline{ID}$, let $n' \neq n_1$. Pick $\rho_{n'}^{(\overline{ID})}$ at random from $\mathcal{Z}_p$. Then compute and release the following $\mathbf{DK}(\overline{ID})$:

$$\begin{bmatrix} \mathsf{k}_{\overline{ID}}^{(\mathrm{O})} \leftarrow g^{\mu_{d_1}} \cdot \Upsilon^{-1} \cdot \prod_{\delta=(n,j)\in[2]\times[2]} \left(y_{\varphi_1,\delta}{}^{\overline{ID}_j}\right)^{\rho_n^{(\overline{ID})}}, \\ \left[\mathsf{k}_{\overline{ID},n}^{(\mathrm{I})} \leftarrow \mathsf{a}_{\varphi_1,n}{}^{-\rho_n^{(\overline{ID})}}, \mathsf{k}_{\overline{ID},n}^{(\mathrm{II})} \leftarrow \mathsf{b}_{\varphi_1,n}{}^{-\rho_n^{(\overline{ID})}}\right]_{n\in[2]} \end{bmatrix}$$

As before, here $\widehat{\mu}_{d_1} = g^{\mu_{d_1}} \cdot \Upsilon^{-1}$. $\mathsf{k}_{\overline{ID}}^{(\mathrm{O})}$ can be computed because the terms containing $y_{\varphi_1,(n_1,j)}$ cancel out, leaving $\mathsf{k}_{\overline{ID}}^{(\mathrm{O})} = g^{\mu_{d_1}} \cdot \prod_{\delta=(n',j)\in\{n'\}\times[2]} \left(y_{\varphi_1,\delta}{}^{\overline{ID}_j}\right)^{\rho_{n'}^{(\overline{ID})}}$.

(g) For each $ID \in \Lambda_{d_1}(\mathbf{B})$ such that $\Phi(ID) = \varphi_1$ and $ID \neq \overline{ID}$, compute and release $\mathbf{DK}(ID)$:

$$\left[\begin{array}{l} \mathsf{k}^{(\mathrm{O})}_{ID} \leftarrow g^{\mu_{d_1}} \cdot \Upsilon^{-1} \cdot \prod_{\delta=(n,j)\in[2]\times[2]} \left(y_{\varphi_1,\delta}{}^{ID_j}\right)^{\rho_n^{(ID)}}, \\ \left[\mathsf{k}^{(\mathrm{I})}_{ID,n} \leftarrow \mathsf{a}_{\varphi_1,n}{}^{-\rho_n^{(ID)}}, \mathsf{k}^{(\mathrm{II})}_{ID,n} \leftarrow \mathsf{b}_{\varphi_1,n}{}^{-\rho_n^{(ID)}}\right]_{n\in[2]} \end{array}\right]$$

Again, to be able to generate $\mathsf{k}^{(\mathrm{O})}_{ID}$, Lemma C.2 is required. However, in this case, a slight complication is involved, since two terms in $\mathsf{k}^{(\mathrm{O})}_{ID}$ contain $y_{\varphi_1,(n_1,j)}$:

$$\begin{aligned} \mathsf{k}^{(\mathrm{O})}_{ID} &= g^{\mu_{d_1}} \cdot \Upsilon^{-1} \cdot \prod_{\delta=(n,j)\in[2]\times[2]} \left(y_{\varphi_1,\delta}{}^{ID_j}\right)^{\rho_n^{(ID)}} \\ &= g^{\mu_{d_1}} \cdot \prod_{\delta_1=(n_1,j)\in\{n_1\}\times[2]} y_{\varphi_1,\delta_1}{}^{-\overline{ID}_j \rho_{n_1}^{(\overline{ID})}} \cdot \prod_{\delta=(n,j)\in[2]\times[2]} \left(y_{\varphi_1,\delta}{}^{ID_j}\right)^{\rho_n^{(ID)}} \\ &= g^{\mu_{d_1}} \cdot \prod_{\delta_1=(n_1,j)\in\{n_1\}\times[2]} \left(y_{\varphi_1,\delta_1}{}^{-\overline{ID}_j \rho_{n_1}^{(\overline{ID})}} \cdot y_{\varphi_1,\delta_1}{}^{ID_j \rho_{n_1}^{(ID)}}\right) \\ &\quad \cdot \prod_{\delta=(n',j)\in\{n'\}\times[2]} y_{\varphi_1,\delta}{}^{ID_j \rho_{n'}^{(ID)}} \end{aligned}$$

Now the simulator picks $\rho_{n_1}^{(ID)}$ at random from $\mathcal{Z}_p^*$, and computes

$$\widetilde{\rho}_{n_1}^{(ID)} = \rho_{n_1}^{(ID)} \frac{\sum_{j\in[2]} ID_j \theta_{\varphi_1,(n_1,j)}}{\sum_{j\in[2]} \overline{ID}_j \theta_{\varphi_1,(n_1,j)}} - \rho_{n_1}^{(\overline{ID})} \tag{5}$$

Here we require that $\sum_{j\in[2]} \overline{ID}_j \theta_{\varphi_1,(n_1,j)} \neq 0$, and as we explained in the **Setup** stage, the simulator aborts if it happens to pick $\theta_{\varphi_1,(n_1,j)}$'s such that $\sum_{j\in[2]} \overline{ID}_j \theta_{\varphi_1,(n_1,j)} = 0$. Hence,

$$\mathsf{k}^{(\mathrm{O})}_{ID} = g^{\mu_{d_1}} \cdot \prod_{\delta_1=(n_1,j)\in\{n_1\}\times[2]} \left(y_{\varphi_1,\delta_1}{}^{\overline{ID}_j \widetilde{\rho}_{n_1}^{(ID)}}\right) \cdot \prod_{\delta=(n',j)\in\{n'\}\times[2]} y_{\varphi_1,\delta}{}^{ID_j \rho_{n'}^{(ID)}}$$

And now the simulator can apply Lemma C.2 by substituting (d_2, l_2, n_2) in the lemma with (d_1, l_1, n'), ρ_1 with $\widetilde{\rho}_{n_1}^{(ID)}$, ID_1 with $\overline{ID}$, and ID_2 with ID.

Challenge: On receiving a message Msg from the adversary, the simulator does the following:

1. Pick random integers $[r_{\varphi,n}]_{\varphi=(d,l)\in[D]\times[L],n\in[2]} \in \mathcal{Z}_p^{2DL}$.

2. Compute and release the following as the ciphertext.

$$\left[\begin{array}{l} *, g_{34}^{+}, [*, *], \ldots, [*, *], \prod_{\substack{j\in[2],\\ \delta_1=(n_1,j)}} (g_{24}^{\times})^{\theta_{\varphi_1,\delta_1}\mathcal{I}^*_{\varphi_1,j}}, \prod_{\substack{j\in[2],\\ \delta_1=(n_1,j)}} Y^{\theta_{\varphi_1,\delta_1}\mathcal{I}^*_{\varphi_1,j}}, \\ \left[\prod_{\substack{j\in[2],\\ \delta=(n,j)}} (g^{r_{\varphi,n}})^{\beta_{\varphi,n}\theta_{\varphi,\delta}\mathcal{I}^*_{\varphi,j}}, \prod_{\substack{j\in[2],\\ \delta=(n,j)}} (g_{34}^{+}\cdot g^{-r_{\varphi,n}})^{\alpha_{\varphi,n}\theta_{\varphi,\delta}\mathcal{I}^*_{\varphi,j}}, \right]_{(d_1,l_1,n_1)<(d,l,n)<(D,L,2),\varphi=(d,l)} \end{array}\right]$$

where $(d, l, n) < (d', l', n')$ if and only if 1) $d < d'$; or 2) $d = d'$ and $l < l'$; or 3) $(d, l) = (d', l')$ and $n < n'$.

Note that this implies that $r = z_3 + z_4$ and $r_{\varphi_1,n_1} = z_4$. If $Y = g^{z_1 z_3}$, it is easy to verify that the ciphertext is well-formed, due to the fact that

$$\left[\sum_{j\in[2],\delta=(n,j)} \bar{\theta}_{\varphi,\delta}\mathcal{I}^*_{\varphi,j} = 0\right]_{(d,l,n)\neq(d_1,l_1,n_1),\varphi=(d,l)}$$

If Y is a random number, then term $c^{(\mathrm{II})}_{(d_1,l_1),n_1}$ is random and independent of the remaining terms of the ciphertext.

Phase 2: Phase 1 is repeated.

Guess: If the adversary guesses that the ciphertext is an encryption of $\mathbf{Msg}$ under $\mathbf{X}^*$, the simulator guesses that $Y = g^{z_3+z_4}$. Else if the adversary guesses that the ciphertext is the encryption under a random point, then the simulator guesses that Y is picked at random from $\mathcal{G}$. ■

Proof of Theorem 6.3: The theorem follows naturally from Lemma C.1 and the hybrid argument. ■