Multi-Dimensional Range Query over Encrypted Data [1]

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

May 2006. Updated: March 2007.
CMU-CS-06-135R

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[1]This research was supported in part by CyLab at Carnegie Mellon under grant DAAD19-02-1-0389 and Cyber-TA Research grant No. W911NF-06-1-0316 from the Army Research Office, and grants 0433540 and 0448452 from the National Science Foundation, and a grant from GM. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of ARO, CMU, GM, NSF, or the U.S. Government or any of its agencies.

This report is an updated version of Computer Science Technical Report CMU-CS-06-135 (May 2006). The conference verion of this report will appear in Proceedings of IEEE Symposium on Security and Privacy, May, 2007.

Keywords: applied cryptography, range query, searchable encryption, anonymous identity-based encryption

Abstract

We design an encryption scheme called Multi-dimensional Range Query over Encrypted Data (MRQED), to address the privacy concerns related to the sharing of network audit logs and various other applications. Our scheme allows a network gateway to encrypt summaries of network flows before submitting them to an untrusted repository. When network intrusions are suspected, an authority can release a key to an auditor, allowing the auditor to decrypt flows whose attributes (e.g., source and destination addresses, port numbers, etc.) fall within specific ranges. However, the privacy of all irrelevant flows are still preserved. We formally define the security for MRQED and prove the security of our construction under the decision bilinear Diffie-Hellman and decision linear assumptions in certain bilinear groups. We study the practical performance of our construction in the context of network audit logs. Apart from network audit logs, our scheme also has interesting applications for financial audit logs, medical privacy, untrusted remote storage, etc. In particular, we show that MRQED implies a solution to its dual problem, which enables investors to trade stocks through a broker in a privacy-preserving manner.

1 Introduction

Recently, the network intrusion detection community has made large-scale efforts to collect network audit logs from different sites [25, 35, 24]. In this application, a network gateway or an Internet Service Provider (ISP) can submit network traces to an audit log repository. However, due to the presence of privacy sensitive information in the network traces, the gateway will allow only authorized parties to search their audit logs. We consider the following four types of entities: a *gateway*, an *untrusted repository*, an *authority*, and an *auditor*. We design a cryptographic primitive that allows the gateway to submit encrypted audit logs to the untrusted repository. Normally, no one is able to decrypt these audit logs. However, when malicious behavior is suspected, an auditor may ask the authority for a search capability. With this search capability, the auditor can decrypt entries satisfying certain properties, e.g., network flows whose destination address and port number fall within a certain range. However, the privacy of all other flows should still be preserved. Note that in practice, to avoid a central point of trust, we can have multiple parties to jointly act as the authority. Only when a sufficient number of the parities collaborate, can they generate a valid search capability.

We name our encryption scheme Multi-dimensional Range Query over Encrypted Data (MRQED). In MRQED, we encrypt a message with a set of attributes. For example, in the network audit log application, the attributes are the fields of a network flow, e.g., source and destination addresses, port numbers, time-stamp, protocol number, etc. Among these attributes, suppose that we would like to support queries on the time-stamp t, the source address a and the destination port number p. Our encryption scheme provides the following properties:

- **Range query on attributes.** An authority can issue a decryption key for all flows whose (t, a, p) falls within a certain range: $t \in [t_1, t_2]$ and $a \in [a_1, a_2]$ and $p \in [p_1, p_2]$. Notice that range query implies equality and greater-than (smaller-than) tests, e.g., $t \geq t_1$ and $a = a_1$ and $p \leq p_1$. With this decryption key, all flows whose (t, a, p) tuple falls within the above range can be decrypted.

- **Security requirement.** Normally, no one can learn any information from the ciphertexts. Under special circumstances, however, an auditor may obtain a decryption key from an authority for some range $t \in [t_1, t_2]$ and $a \in [a_1, a_2]$ and $p \in [p_1, p_2]$. For any flow, if at least one attribute among t, a, p lies outside the specified range, the auditor fails to decrypt it. The auditor inevitably learns that the (t, a, p) tuple of this flow does not lie within the given range. However, apart from this information, the auditor cannot learn anything more about the flow. For example, the auditor cannot learn anything about attributes other than t, a, p; in addition, she cannot decide whether $t < t_1$ or $t > t_2$, etc.

Our results and contributions. We are among the earliest to study the problem of point encryption, range query, and conditional decryption of matching entries. We propose a provably secure encryption scheme that allows us to achieve these properties. Table 1 summarizes the asymptotic performance of our scheme in comparison with other approaches. Please refer to Section 2 for a detailed comparison between our scheme MRQED, and the concurrent work BonehWaters06 [13].

Scheme	Pub. Key Size	Encrypt. Cost	CT Size	Decrypt. Key Size	Decrypt. Cost	Security
BonehWaters06 [13]	$O(D \cdot T)$	$O(D \cdot T)$	$O(D \cdot T)$	$O(D)$	$O(D)$	MC
Naive AIBE-based	$O(1)$	$O((\log T)^D)$	$O((\log T)^D)$	$O((\log T)^D)$	$O((\log T)^D)$	MR
Our scheme	$O(D \cdot \log T)$	$O(D \cdot \log T)$	$O(D \cdot \log T)$	$O(D \cdot \log T)$	$O((\log T)^D)$	MR

Table 1: Performance of different approaches. D denotes the number of dimensions and T the number of points in each. The naive AIBE-based scheme is described in Section 4.3. MC and MR refer to the *match-concealing* and *match-revealing* security models respectively as defined in Section 3.

We study the practical performance of MRQED, and show that it makes the encrypted network audit log application feasible. We also study the dual problem to MRQED, where one encrypts under a hyper-range in multi-dimensional space, and decrypts under a point. We show that MRQED implies a solution to its dual problem, which enables investors to trade stocks through a broker in a privacy-preserving manner.

Paper organization. In the remainder of this section, we give more example applications of MRQED. We review related work in Section 2, and formally define the MRQED problem in Section 3. In Section 4, we demonstrate some initial attempts at constructing MRQED; while in Section 5, we describe our novel construction which we consider the main contribution of this paper. We note that the purpose of Section 4 is not only to exhibit straw-man schemes, but also to better motivate our design of MRQED as described in Section 5. In particular, some of the primitives introduced in Section 4 will later be used in Section 5 when we explain our novel construction. Due to limit of space, formal security proofs of security are provided in Appendix C. In the proof, we borrow techniques from the AHIBE scheme of Boyen and Waters [15]. As a result, the security of our construction is likewise based on the hardness of Decision Bilinear Diffie-Hellman problem and the Decision Linear problem. In Section 7, we consider the practical performance of the scheme in the context of network audit logs. We show that MRQED implies a solution to its dual problem in Section 8, and show that the dual problem is of particular interest to investors who would like to trade stocks through a broker in a privacy-preserving manner.

1.1 Application to Network Audit Logs

We briefly mentioned network audit logs at the beginning of this section. Throughout the paper, we will keep using this example to motivate the design of MRQED. To provide context for the remainder of the paper, we now describe this application in greater detail.

Firewalls and network intrusion detection systems (NIDS) such as Snort [43], Emerald [40], and Bro [39] produce logs summarizing detected or blocked activities suspected to be malicious. Log entries typically correspond to either a single packet (perhaps rejected by a firewall) or an established flow deemed suspicious. Each entry normally includes fields such as source and destination IP address and port, date and time, protocol (e.g., TCP, UDP, or ICMP), and, in the case of NIDS, the type of rule causing an alert. Sharing and comparing such logs across organizations is a method for gaining broader information about malicious activities on the Internet so that administrators may better protect their systems. Current large scale efforts to collect and aggregate network audit logs for this purpose include DShield [25], myNetWatchman [35], and Deepsight [24].

However, sharing of network audit logs is hampered by the presence of security and privacy sensitive information. By encrypting each log entry before sending it to another party, the source can allay these concerns. Later, the source may release a decryption key for a carefully specified set of log entries deemed currently relevant. For example, suppose a particular host with IP address a_1 is determined to have been compromised at time t_1 and later involved in scanning other hosts for vulnerabilities on a certain range of ports $[p_1, p_2]$. A trusted authority may then choose to release a key decrypting any entries at time t, with source address a, connecting to port p such that $t \geq t_1$, $a = a_1$, and $p_1 \leq p \leq p_2$. Note that to avoid a central point of trust, we can have multiple parties jointly act as the authority. Using techniques from secure multi-party computation [27], only when a sufficient number of them collaborate, can they generate a valid decryption key. The source would then have precise guarantees about the privacy of their network while providing useful information to other individual organizations or a global monitoring effort. The public key nature of the scheme would allow distributed, encrypted submissions to a central monitoring organization possessing the master private key and giving out decryption keys as necessary. There have been some previous attempts to protect the security of audit logs through encryption or anonymization while allowing limited queries [46, 23, 33], but in no previous scheme has it been possible to issue keys for conjunctions of ranges over multiple attributes while maintaining the secrecy of the attributes. In particular, we are not aware of any previous method supporting queries such as our example of $(t \geq t_1) \wedge (a = a_1) \wedge (p_1 \leq p \leq p_2)$ that does not require either revealing the attribute values or issuing an exponential number of key components.

1.2 Other Applications

Apart from the network audit log application, and the stock-trading application described in Section 8, we mention here some other potentially interesting applications of MRQED.

Financial audit logs. Financial audit logs contain sensitive information about financial transactions. Our MRQED scheme allows financial institutions to release audit logs in encrypted format. When necessary, an authorized auditor can obtain a decryption key from a trusted authority. With this decryption key, the auditor can decrypt certain transactions that may be suspected of fraudulent activities. However, the privacy of all other transactions are preserved.

Medical privacy. Consider a health monitoring program. When Alice moves about 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 $((x, y, t), ct)$, 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 was present at a site borne with the disease during an incubation period, i.e., if (x, y, t) falls within a certain range. 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 MRQED scheme can also be used in biometric-based Anonymous Identity-Based Encryption (AIBE). Using biometrics in identity-based encryption first appeared in the work by Sahai and Waters [41]. 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* $\mathbf{X}'$ and $\mathbf{X}$ have small distance. In particular, the SahaiWaters04 construction [41] considered the *set-overlap* distance (or the *Hamming* distance); and their encryption scheme does not hide the identity of the user. Our construction allows a user with the private key for identity $\mathbf{X}$, to decrypt an entry encrypted under $\mathbf{X}'$, iff $\ell_\infty(\mathbf{X}, \mathbf{X}') \leq \epsilon$. Here ℓ_∞ denotes the ℓ_∞ distance between $\mathbf{X}$ and $\mathbf{X}'$, and is defined as $\max\{|x_1 - x_1'|, \ldots, |x_D - x_D'|\}$. In this case, the decryption region is a hyper-cube in multi-dimensional space. One can also associate a different weight to each dimension, in which case the decryption region becomes a hyper-rectangle.

2 Related Work

Search on encrypted data. The problem of search on encrypted data (SoE) was introduced in the symmetric key setting by Song et al. [44] and has had some recent improvements in security definitions and efficiency [21]. Boneh et al. [10] later proposed Public Key Encryption with Keyword Search (PEKS), in which any party possessing the public key can encrypt and the owner of the corresponding private key can generate keyword search capabilities. Both SoE and PEKS can be trivially extended to support one-dimensional range queries; the extension is similar to the MRQED[1] scheme described in Section 4.2. However, it is not clear that either can be used to construct a scheme supporting range queries over multiple attributes. Recent work on traitor-tracing systems [14, 12] allows a more specialized sort of range query. Given a ciphertext $\mathbf{C}$ with attributes $\mathbf{X} = (x_1, x_2, \ldots, x_D)$, a master key owner can issue a token for some value x' 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. Applications of searchable encryption have been studied by the database community [30, 22, 2]. Other works related to searches on encrypted data include oblivious RAMs [37, 28], and private stream searching [5, 38].

IBE. The notion of Identity-Based Encryption (IBE) was introduced by Shamir [42]. Several IBE schemes [20, 11, 7, 6, 18, 45, 36], hierarchical IBE (HIBE) schemes [31, 26, 8, 47], and applications [41, 29] were proposed since then. In particular, the HIBE scheme proposed by

Boneh, Boyen, and Goh [8] can be extended to multiple dimensions (M-HIBE) efficiently and in a collusion-resistant[1] manner. The resulting scheme can be used to solve a problem similar to MRQED, but lacking the third property in the previous discussion. That is, when using M-HIBE it would not be possible to hide the attribute values associated with a ciphertext.

Anonymous IBE. Recently, researchers have proposed anonymous IBE and HIBE schemes (AIBE, AHIBE) [15, 1]. The notion of anonymity is also related to key privacy [4, 3]. Like the HIBE scheme mentioned above, the AHIBE scheme of Boyen and Waters [15] can be extended to multiple dimensions in a collusion-resistant manner, resulting in a Multi-dimensional AHIBE (M-AHIBE) scheme. An M-AHIBE scheme could be used to implement MRQED (including the third property), but applying it directly would have a serious drawback. Because the encryption is anonymous and hides the attributes used as the public key, at time of decryption one would need to try all possible decryption keys on a given ciphertext. This incurs $O(T^D)$ decryption cost on a single ciphertext, where T is the number of possible values each attribute may assume and may be quite large. Nevertheless, on a technical level, this AHIBE scheme and its extension to M-AHIBE are the most closely related work to ours. In particular, we prevent collusion in the same way the M-AHIBE construction does. Since we do not require the key delegation property of HIBE schemes, however, we are able to improve decryption cost to be logarithmic in T.

Recent developments. Concurrent to our work, Boneh and Waters [13] propose another construction (BonehWaters06 in Table 1) for complex queries over encrypted data. They propose a primitive called Hidden Vector Encryption, and use it in conjunctive range and subset queries. When applied to multi-dimensional range query, their scheme results in $O(DT)$ encryption time, ciphertext size, and public key size, and $O(D)$ decryption key size and decryption cost. As in Table 1, D and T are the number of attributes and the number of discrete values for each attribute. Their scheme is more expensive in terms of public key size, encryption cost and ciphertext size; but saves on decryption key size and decryption cost. In applications with large T and small D (e.g., network audit logs, and the stock trading application mentioned in Section 8), our approach is more appropriate. In particular, for network audit logs, $T = 2^{32}$ for an IP address, and D may range from 2 to 4. In other applications where D is large and T is small, the BonehWaters06 construction is more appropriate. We also would like to note that the BonehWaters06 construction achieves a stronger notion of security. Their construction hides the attribute values, even when the message is successfully decrypted. This stronger security property is a key difference from our construction, in which the attribute values are revealed upon successful decryption. In Section 3, we name these two different security models *match-concealing* security and *match-revealing* security respectively. For applications like encrypted network audit logs, it is acceptable to reveal the attributes of a message when it is successfully decrypted. By relaxing the security definition to allow this possibility, we achieve $O(D \log T)$ encryption time, ciphertext size, and public key size. This makes applications such as the encrypted network audit logs possible. However, one may conceive of other applications where the stronger security notion is necessary.

[1]Collusion-resistance, in this sense, means that two parties who have been issued different decryption keys cannot combine their keys in some way to allow decryption of ciphertexts that neither could decrypt previously.

3 Problem Definition and Preliminary

3.1 Problem Definition

In the network audit log application, a gateway encrypts network flows, and submits them to an untrusted repository. When necessary, an auditor may ask an authority for a key that allows the decryption of all flows whose attributes fall within a certain range; while the privacy of all irrelevant flows are still preserved. There is a geometric interpretation to these multi-attribute range queries. Suppose that we would like to allow queries on these three fields: time-stamp t, source address a, and destination port p. The tuple (t, a, p) can be regarded as a point $\mathbf{X}$ in multi-dimensional space. Now suppose we query for all flows whose t, a, p falls within some range: $t \in [t_1, t_2]$, $a \in [a_1, a_2]$ and $p \in [p_1, p_2]$. Here the "hyper-range" $[t_1, t_2] \times [a_1, a_2] \times [p_1, p_2]$ forms a hyper-rectangle $\mathbf{B}$ in space. The above range query is equivalent to testing whether a point $\mathbf{X}$ falls inside the hyper-rectangle $\mathbf{B}$.

We now formally define these notions mentioned above. Assume that an attribute can be encoded using discrete integer values 1 through T. For example, an IP address can be encoded using integers 1 through 2^{32}. 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., $[S, T] = \{S, S+1, \ldots, T\}$. Throughout this paper, we assume that T is a power of 2, and denote $\log_2$ as simply $\log$. Suppose that we would like to support range queries on D different attributes, each of them can take on values in $[T_1], [T_2], \ldots, [T_D]$ respectively. We formally define a D-dimensional lattice, points and hyper-rectangles below.

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

A MRQED scheme consists of four (randomized) polynomial-time algorithms: $\mathbf{Setup}$, $\mathbf{Encrypt}$, $\mathbf{DeriveKey}$ and $\mathbf{QueryDecrypt}$. In the network audit log example, an authority runs $\mathbf{Setup}$ to generate public parameters and a master private key; a gateway runs the $\mathbf{Encrypt}$ algorithm to encrypt a flow. Encryption is performed on a pair $(\mathbf{Msg}, \mathbf{X})$. The message $\mathbf{Msg}$ is an arbitrary string, and $\mathbf{X}$ is a point in multi-dimensional space, representing the attributes. For example, suppose that we would like to support queries on the following three attributes of a flow: time-stamp t, source address a, and destination port p. The tuple (t, a, p) then becomes the point $\mathbf{X}$, and the entire flow summary forms the message $\mathbf{Msg}$. Whenever necessary, the authority can run the $\mathbf{DeriveKey}$ algorithm, and compute a decryption key allowing the decryption of flows whose attributes fall within a certain range. Given this decryption key, an auditor runs the $\mathbf{QueryDecrypt}$ algorithm over the encrypted data to decrypt the relevant flows. We now formally define MRQED.

Definition 2 (MRQED)**.** *An Multi-dimensional Range Query over Encrypted Data (MRQED) scheme consists of the following polynomial-time randomized algorithms.*

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

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 $\mathbb{M}$ and outputs a ciphertext $\mathbf{C}$.

3. $\mathbf{DeriveKey}(\mathbf{PK}, \mathbf{SK}, \mathbf{B})$: Takes a public key $\mathbf{PK}$, a master private key $\mathbf{SK}$, 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 $\mathbf{Msg} \in \mathbb{M}$, hyper-rectangle $\mathbf{B} \subseteq \mathbb{L}_\Delta$, and point $\mathbf{X} \in \mathbb{L}_\Delta$, 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 $\mathbf{DK} = \mathbf{DeriveKey}(\mathbf{PK}, \mathbf{SK}, \mathbf{B})$.

3.2 Security Definitions

Suppose that during time $[t_1, t_2]$, there is an outbreak of a worm characteristic by the port number p_1. Now the trusted authority issues a key for the range $t \in [t_1, t_2]$ and $p = p_1$ to a research group who has been asked to study the worm behavior. With this key, the research group should be able to decrypt only flows whose time-stamp and port number fall within the given range. The privacy of all other flows should still be preserved. Informally, suppose that a computationally bounded adversary has obtained decryption keys for regions $\mathbf{B}_0, \mathbf{B}_1, \ldots, \mathbf{B}_q$. Now given a ciphertext $\mathbf{C} = \mathbf{Encrypt}(\mathbf{PK}, \mathbf{X}, \mathbf{Msg})$ such that $\mathbf{X} \notin \mathbf{B}_0, \mathbf{B}_1, \ldots, \mathbf{B}_q$, the adversary cannot learn $\mathbf{X}$ or $\mathbf{Msg}$ from $\mathbf{C}$. Of course, since the adversary fails to decrypt $\mathbf{C}$ using keys for regions $\mathbf{B}_0, \mathbf{B}_1, \ldots, \mathbf{B}_q$, the adversary inevitably learns that the point $\mathbf{X}$ encrypted does not fall within these regions. But apart from this fact, the adversary cannot learn more information about $\mathbf{X}$ or $\mathbf{Msg}$.

We now formalize this intuition into a *selective security* game for MRQED. Here, the selective security notion is similar to the selective-ID security for IBE schemes [16, 17, 6]. We prove the security of our construction in the selective model. A stronger security notion is adaptive security, where the adversary does not have to commit to two points in the **Init** stage of the security game defined below. In Appendix D, we give a formal definition for adaptive security, and state how it is related to the selective security model.

Definition 3 (MR-selective security)**.** *An MRQED scheme is* ***selectively secure*** *in the* ***match-revealing*** *(MR) model if all polynomial-time adversaries have at most a negligible advantage in the selective security game defined below.*

- **Init**: The adversary submits two points $\mathbf{X}_0^*, \mathbf{X}_1^* \in \mathbb{L}_\Delta$ where it wishes to be challenged.

- **Setup**: The challenger runs the $\mathbf{Setup}(\mathbf{\Sigma}, \mathbb{L}_{\mathbf{\Delta}})$ algorithm to generate $\mathbf{PK}, \mathbf{SK}$. It gives $\mathbf{PK}$ to the adversary, keeping $\mathbf{SK}$ secret.

- **Phase 1**: The adversary adaptively issues decryption key queries for q_0 hyper-rectangles $\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_{q_0}$. Furthermore, $\mathbf{X}_0^*$ and $\mathbf{X}_1^*$ are not contained in any hyper-rectangles queried in this phase, i.e., for $0 < i \leq q_0$, $\mathbf{X}_0^* \notin \mathbf{B}_i$, and $\mathbf{X}_1^* \notin \mathbf{B}_i$.

- **Challenge**: The adversary submits two equal length messages $\mathbf{Msg}_0, \mathbf{Msg}_1 \in \mathbb{M}$. The challenger flips a random coin, b, and encrypts $\mathbf{Msg}_b$ under $\mathbf{X}_b^*$. The ciphertext is passed to the adversary.

- **Phase 2**: Phase 1 is repeated. The adversary adaptively issues decryption key queries for $q - q_0$ hyper-rectangles $\mathbf{B}_{q_0+1}, \mathbf{B}_{q_0+2}, \ldots, \mathbf{B}_q$. As before, all hyper-rectangles queried in this stage must not contain $\mathbf{X}_0^*$ and $\mathbf{X}_1^*$.

- **Guess**: The adversary outputs a guess b' of b.

An adversary $\mathcal{A}$'s advantage in the above game is defined as $\mathbf{Adv}_{\mathcal{A}}(\Sigma) = \left|\Pr[b = b'] - \frac{1}{2}\right|$.

We would like to note that a stronger notion of security is possible as defined by Boneh and Waters in their concurrent work [13]. We call this stronger security notion *match-concealing (MC) security*, since it requires that the attribute values (i.e., the point $\mathbf{X}$) remain hidden even when an entry matches a query. MC-selective security can be formally defined through the following game between an adversary and a challenger.

Definition 4 (MC-selective security [13]). *An MRQED scheme is* ***selectively secure*** *in the* ***match-concealing*** *(MC) model if all polynomial-time adversaries have at most a negligible advantage in the selective security game defined below.*

- **Init**: The adversary submits two points $\mathbf{X}_0^*, \mathbf{X}_1^* \in \mathbb{L}_\Delta$ where it wishes to be challenged.

- **Setup**: The challenger runs the $\mathbf{Setup}(\mathbf{\Sigma}, \mathbb{L}_{\mathbf{\Delta}})$ algorithm to generate $\mathbf{PK}, \mathbf{SK}$. It gives $\mathbf{PK}$ to the adversary, keeping $\mathbf{SK}$ secret.

- **Phase 1**: The adversary adaptively issues decryption key queries for q_0 hyper-rectangles $\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_{q_0}$, satisfying the condition that for all $0 < i \leq q_0$, either $(\mathbf{X}_0^* \in \mathbf{B}_i) \wedge (\mathbf{X}_1^* \in \mathbf{B}_i)$, or $(\mathbf{X}_0^* \notin \mathbf{B}_i) \wedge (\mathbf{X}_1^* \notin \mathbf{B}_i)$.

- **Challenge**: The adversary submits two equal length messages $\mathbf{Msg}_0, \mathbf{Msg}_1 \in \mathbb{M}$. If in Phase 1, there exists some $0 < i \leq q_0$ such that $(\mathbf{X}_0^* \in \mathbf{B}_i) \wedge (\mathbf{X}_1^* \in \mathbf{B}_i)$, then $\mathbf{Msg}_0 = \mathbf{Msg}_1$. The challenger flips a random coin, b, and encrypts $\mathbf{Msg}_b$ under $\mathbf{X}_b^*$. The ciphertext is passed to the adversary.

- **Phase 2**: Phase 1 is repeated. The adversary adaptively issues decryption key queries for hyper-rectangles $\mathbf{B}_{q_0+1}, \mathbf{B}_{q_0+2}, \ldots, \mathbf{B}_q$, satisfying the condition that for all $q_0 < i \leq q$, either $(\mathbf{X}_0^* \in \mathbf{B}_i) \wedge (\mathbf{X}_1^* \in \mathbf{B}_i)$, or $(\mathbf{X}_0^* \notin \mathbf{B}_i) \wedge (\mathbf{X}_1^* \notin \mathbf{B}_i)$. In addition, if in the Challenge stage, $\mathbf{Msg}_0 \neq \mathbf{Msg}_1$, then for all $q_0 < i \leq q$, $(\mathbf{X}_0^* \notin \mathbf{B}_i) \wedge (\mathbf{X}_1^* \notin \mathbf{B}_i)$.

- **Guess**: The adversary outputs a guess b' of b.

Likewise, an adversary $\mathcal{A}$'s advantage in the above game is defined as $\mathbf{Adv}_{\mathcal{A}}(\Sigma) = \left|\Pr[b = b'] - \frac{1}{2}\right|$.

In this paper, we use the MR security model, i.e., we do not protect the privacy of the attributes if an entry is matched by the query. This security notion suffices for applications such as network audit logs, and the stock-trading application as described in Section 8.

3.3 Preliminary: Bilinear Groups

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

We name a tuple $\mathbf{G} = [p, \mathbb{G}, \mathbb{G}', g, \mathbf{e}]$ a bilinear instance, where $\mathbb{G}$ and $\mathbb{G}'$ are two cyclic groups of prime order p. We assume an efficient generation algorithm that on input of a security parameter Σ, outputs $\mathbf{G} \xleftarrow{R} \text{Gen}(\Sigma)$ 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 [32], later used by IBE systems [11], posits the hardness of the following problem:

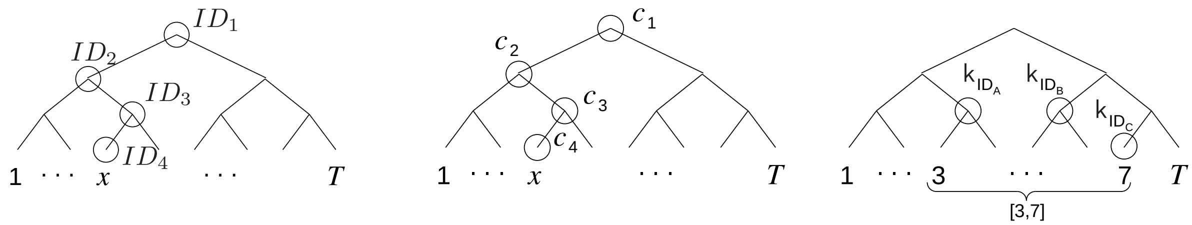
Given $[g, g^{z_1}, g^{z_2}, g^{z_3}, Z] \in \mathbb{G}^4 \times \mathbb{G}'$, where exponents z_1, z_2, z_3 are picked at random from $\mathbb{Z}_p$, decide whether $Z = \mathbf{e}(g, g)^{z_1 z_2 z_3}$.

Decision Linear Assumption : The Decision Linear assumption, first proposed by Boneh, Boyen and Shacham for group signatures [9], posits the hardness of the following problem:

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

4 A First Step towards MRQED

In this section, we first show a trivial construction for MRQED which has $O(T^{2D})$ public key size, $O(T^{2D})$ encryption cost and ciphertext size, $O(1)$ decryption key size and decryption cost. Then in Section 4.2, we show that using AIBE, we can obtain an improved one-dimension MRQED scheme. Henceforth, we refer to a one-dimension MRQED scheme as MRQED^1 and refer to multi-dimension MRQED as MRQED^D. The AIBE-based MRQED^1 construction has $O(1)$ public key size, $O(\log T)$ encryption cost, ciphertext size, decryption key size and decryption cost. While describing the AIBE-based MRQED^1 construction, we introduce some primitives and notations that will later be used in our main construction in Section 5. In Section 4.3, we demonstrate that a straightforward extension of the AIBE-based MRQED^1 scheme into multiple dimensions results in $O\left((\log T)^D\right)$ encryption cost, ciphertext size, decryption key size and decryption cost. The AIBE-based MRQED^1 construction aids the understanding of our main construction in Section 5. By



(a) The path from a leaf to the root.

(b) A ciphertext and a decryption key in MRQED^1.

Figure 1: An MRQED^1 scheme. ***(a)*** *Path from the leaf node representing $x \in [T]$ to the root.* $\mathbb{P}(x) = \{ID_1, ID_2, ID_3, ID_4\}$. ***(b)*** *Encryption under the point $x = 3$ and the keys released for the range* $[3, 7]$.

contrast, details of the AIBE-based MRQED^D scheme are not crucial towards the understanding of our main construction. Therefore, we only highlight a few important definitions and give a sketch of the scheme in Section 4.3. We give the detailed description of the AIBE-based MRQED^D scheme in Appendix F.

4.1 Trivial Construction

We first give a trivial construction for one-dimensional range query over encrypted data. We refer to one-dimensional range query over encrypted data as MRQED^1 where the superscript represents the number of dimensions.

In the trivial MRQED^1 construction, we make use of any secure public key encryption scheme. We first generate $O(T^2)$ public-private key pairs, one for each range $[s, t] \subseteq [1, T]$. To encrypt a message $\mathbf{Msg}$ under a point x, we produce $O(T^2)$ ciphertexts, one for each range $[s, t] \subseteq [1, T]$. In particular, if $x \in [s, t]$, we encrypt $\mathbf{Msg}$ with public key $\mathsf{pk}_{s,t}$; otherwise, we encrypt an invalid message $\perp$ with $\mathsf{pk}_{s,t}$. The decryption key for any range $[s, t]$ is then $\mathsf{sk}_{s,t}$, the private key for $[s, t]$. In Appendix E, we give a formal description of this trivial construction.

One can extend this idea into multiple dimensions. The resulting MRQED^D scheme requires that one encrypt $\delta_{\mathbf{B}}(\mathbf{Msg}, \mathbf{X})$ for all hyper-rectangles $\mathbf{B}$ in space. Therefore, the trivial MRQED^D scheme has $O(T^{2D})$ public key size, $O(T^{2D})$ encryption cost and ciphertext size, $O(1)$ decryption key size and $O(1)$ decryption cost.

4.2 Improved MRQED^1 Construction Based on AIBE

We show an improved MRQED construction based on Anonymous Identity-Based Encryption (AIBE). For clarity, we first explain the construction for one dimension. We call the scheme MRQED^1 where the superscript denotes the number of dimensions. We note that the primitives and notations introduced in this section will be used in our main construction.

4.2.1 Primitives: Efficient Representation of Ranges

To represent ranges efficiently, we build a binary interval tree over integers 1 through T.

Definition 5 (Interval tree). *Let* $\mathsf{tr}(T)$ *denote a binary interval tree over integers from* 1 *to* T*. Each node in the tree has a pre-assigned unique* ID*. For convenience, we define* $\mathsf{tr}(T)$ *to be the set of all node* ID*s in the tree. Each node in* $\mathsf{tr}(T)$ *represents a range. Let* $\mathsf{cv}(ID)$ *denote the range represented by node* $ID \in \mathsf{tr}(T)$*. Define* $\mathsf{cv}(ID)$ *as the following: Let* ID *be the* i^{th} *leaf node, then* $\mathsf{cv}(ID) = i$*. Otherwise, when* ID *is an internal node, let* ID_1 *and* ID_2 *denote its child nodes, then* $\mathsf{cv}(ID) = \mathsf{cv}(ID_1) \cup \mathsf{cv}(ID_2)$*. In other words,* $\mathsf{cv}(ID)$ *is the set of integers that correspond to the leaf descendants of* ID*.*

Given the interval tree $\mathsf{tr}(T)$, we define the $\mathbb{P}(x)$ of IDs covering a point $x \in [1, T]$, and the set $\Lambda(x)$ of IDs representing a range $[s, t] \subseteq [1, T]$.

- **Set of IDs covering a point x.** For a point $x \in [1, T]$ and some node $ID \in \mathsf{tr}(T)$, we say that ID *covers* the point x if $x \in \mathsf{cv}(ID)$. Define $\mathbb{P}(x)$ to be the set of IDs covering point x. Clearly, $\mathbb{P}(x)$ is the collection of nodes on the path from the root to the leaf node representing x. As an example, in Figure 1 (a), $\mathbb{P}(x) = \{ID_1, ID_2, ID_3, ID_4\}$.

- **Range as a collection of IDs.** A range $[s, t] \subseteq [1, T]$ is represented by a collection of nodes: $\Lambda(s, t) \subseteq \mathsf{tr}(T)$. We define $\Lambda(s, t)$ to be the smallest of all subsets $\mathbb{V} \subseteq \mathsf{tr}(T)$ such that $\bigcup_{ID \in \mathbb{V}} \mathsf{cv}(ID) = [s, t]$. It is not hard to see that for any $[s, t] \subseteq [1, T]$, $\Lambda(s, t)$ is uniquely defined, and its size $|\Lambda(s, t)|$ is at most $O(\log T)$.

We will make use of the following properties in our AIBE-based construction: If $x \in [s, t]$, then $\mathbb{P}(x) \cap \Lambda(s, t) \neq \emptyset$; in addition, $\mathbb{P}(x)$ and $\Lambda(s, t)$ intersect at only one node. Otherwise, if $x \notin [s, t]$, then $\mathbb{P}(x) \cap \Lambda(s, t) = \emptyset$.

4.2.2 AIBE-Based MRQED^1 Scheme

AIBE encrypts a message Msg using an identity ID as the public key. Given the private key for ID, one can successfully decrypt all messages encrypted by identity ID. The encryption scheme protects both the secrecy of the message Msg and the identity ID in the following sense: Given ciphertext C, which is an encryption of Msg by identity ID_0, and given decryption keys for identities $ID_1, ID_2, \ldots, ID_q$ but not for ID_0, a computationally bounded adversary cannot learn anything about Msg or about ID_0 from the ciphertext C. Researchers have successfully constructed secure AIBE schemes [15, 1] with $O(1)$ cost in all respects: in public parameter size, encryption cost, ciphertext size, decryption key size and decryption cost.

Given a secure AIBE scheme, we can construct an MRQED^1 scheme based on the following intuition. To encrypt the message Msg under point x, we encrypt Msg under all IDs in $\mathbb{P}(x)$. To release the decryption key for a range $[s, t] \subseteq [1, T]$, we release the keys for all IDs in $\Lambda(s, t)$. Now if $x \in [s, t]$, then $\mathbb{P}(x) \cap \Lambda(s, t) \neq \emptyset$. Suppose $\mathbb{P}(x)$ and $\Lambda(s, t)$ intersect at node ID. Then we can apply the decryption key at ID to the ciphertext encrypted under ID, and obtain the plaintext message Msg. Otherwise, if $x \notin [s, t]$, then $\mathbb{P}(x) \cap \Lambda(s, t) = \emptyset$. In this case, the security of the underlying AIBE scheme ensures that a computationally bounded adversary cannot learn any information about the message Msg or the point x, except for the obvious fact (since decryption fails) that $x \notin [s, t]$.

Example. In Figure 1(b), we show a ciphertext $\mathbf{C}$ encrypted under the point x. Let $L = O(\log T)$ denote the height of the tree, $\mathbf{C}$ is composed of $O(\log T)$ components: $\{c_1, c_2, \ldots, c_L\}$. On the right, we show the decryption keys for the range $[3, 7]$. Since $[3, 7]$ can be represented by the set of nodes $\Lambda(3, 7) = \{ID_A, ID_B, ID_C\}$, the decryption key for $[3, 7]$ consists of three sub-keys, k_{ID_A}, k_{ID_B} and k_{ID_C}.

The AIBE-based construction has $O(1)$ public key size, $O(|\mathbb{P}(x)|)$ encryption cost and ciphertext size, and $O(|\Lambda(s, t)|)$ decryption key size. Since $|\mathbb{P}(x)| = O(\log T)$, and $|\Lambda(s, t)| = O(\log T)$, we get $O(\log T)$ in encryption cost, ciphertext size, and decryption key size. Later, we will show that decryption can be done in $O(\log T)$ time as well.

Stated more formally, given a secure AIBE scheme

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

one can construct a secure MRQED[1] scheme as below:

- $\mathbf{Setup}(\mathbf{\Sigma}, \mathbf{T})$ calls $\mathbf{Setup}^*(\Sigma)$ and outputs $\mathbf{PK}$ and $\mathbf{SK}$.

- $\mathbf{Encrypt}(\mathbf{PK}, \mathbf{x}, \mathbf{Msg})$ encrypts the message $\mathbf{Msg}$ under every $ID \in \mathbb{P}(x)$. In other words, $\mathbf{Encrypt}$ yields $\mathbf{C} = \left\{ c_{ID} \middle| ID \in \mathbb{P}(x) \right\}$, where $c_{ID} = \mathbf{Encrypt}^*(\mathbf{PK}, ID, \mathbf{Msg}||0^{m'})$. To check whether a decryption is valid, prior to encryption, we append m' trailing 0s denoted $0^{m'}$ to message $\mathbf{Msg} \in \{0, 1\}^m$.

- $\mathbf{DeriveKey}(\mathbf{PK}, \mathbf{SK}, [s, t])$ releases a decryption key k_{ID} for each $ID \in \Lambda(s, t)$. k_{ID} is computed as $\mathsf{k}_{ID} = \mathbf{DeriveKey}^*(\mathbf{PK}, \mathbf{SK}, ID)$. The entire decryption key for the range $[s, t]$ is then the set $\mathbf{DK}_{s,t} = \left\{ \mathsf{k}_{ID} \ \middle| \ ID \in \Lambda(s, t) \right\}$.

- $\mathbf{QueryDecrypt}(\mathbf{PK}, \mathbf{DK}, \mathbf{C})$ tries each key $\mathsf{k}_{ID} \in \mathbf{DK}_{s,t}$ on each ciphertext $c_{ID'} \in \mathbf{C}$. If $ID = ID'$, then $\mathbf{Decrypt}^*(\mathbf{PK}, \mathsf{k}_{ID}, c_{ID'})$ yields result of the form $\widehat{\mathbf{Msg}}||0^{m'}$. In this case, we accept the result and exit the $\mathbf{QueryDecrypt}$ algorithm. If all trials fail to yield result of the form $\widehat{\mathbf{Msg}}||0^{m'}$, $\mathbf{QueryDecrypt}$ outputs $\perp$, indicating failure to decrypt.

Note that in the AIBE-based construction, if we simply try all decryption keys over all ciphertexts, then decryption would require $O(|\mathbb{P}(x)| \cdot |\Lambda(s, t)|)$ time; and since $|\mathbb{P}(x)| = O(\log T)$, $|\Lambda(s, t)| = O(\log T)$, decryption would require $O(\log^2 T)$ time. However, observe that it is not necessary to try k_{ID} on $c_{ID'}$, if ID and ID' are at different depth in the tree; since then, ID and ID' cannot be equal. Thus we only need to try k_{ID} on $c_{ID'}$ if ID and ID' are at the same depth in the tree, which requires knowledge of the depth of ID' for ciphertext $c_{ID'}$. Of course, we cannot directly release ID' for ciphertext $c_{ID'}$, since the encryption is meant to hide ID'. However, since each ciphertext $\mathbf{C}$ has a portion at every depth of the tree, we can give out the depth of ID' for each $c_{ID'} \in \mathbf{C}$ without leaking any information about ID'. In this way, we reduce the decryption cost to $O(\log T)$ rather than $O(\log^2 T)$.

We emphasize that using AIBE as the underlying encryption scheme is crucial to ensuring the security of the derived MRQED[1] scheme. In particular, a non-anonymous IBE scheme is not suitable to use as the underlying encryption scheme, since IBE hides only the message $\mathbf{Msg}$ but not the attribute x.

4.3 AIBE-Based MRQED^D Construction

The same idea can be applied to construct an MRQED^D scheme, resulting in $O(1)$ public key size, $O\left((\log T)^D\right)$ encryption cost, ciphertext size, decryption key size, and decryption cost. Since the details of this construction is not crucial to the understanding of our main construction, we only give a sketch here and leave the full description of the scheme to Appendix F. However, we highlight a few important definitions here, including the notion of a simple hyper-rectangle, and the definition of $\Lambda^\times(\mathbf{B})$. These definitions will later be used in our main construction.

We build D binary interval trees, one for each dimension. We assign a globally unique ID to each node in the D trees.

Representing a hyper-rectangle. We represent an arbitrary hyper-rectangle as a collection of *simple hyper-rectangles*. To illustrate this idea, we first give a formal definition of a simple hyper-rectangle, and then state how to represent an arbitrary hyper-rectangle as a collection of simple hyper-rectangles. Simply put, a simple hyper-rectangle is a hyper-rectangle $\mathbf{B}_0$ in space, such that $\mathbf{B}_0$ can be represented by a single node in the tree of every dimension. More specifically, a hyper-rectangle $\mathbf{B}(s_1, t_1, \ldots, s_D, t_D)$ in space is composed of a range along each dimension. If for all $1 \leq d \leq D$, $|\Lambda(s_d, t_d)| = 1$, i.e., $[s_d, t_d]$ is a simple range in the d^{th} dimension, then we say that the hyper-rectangle $\mathbf{B}(s_1, t_1, \ldots, s_D, t_D)$ is a *simple hyper-rectangle*. A simple hyper-rectangle can be defined by a single node from each dimension. We can assign a unique identity to each simple-rectangle $\mathbf{B}_0(s_1, t_1, \ldots, s_D, t_D)$ in space. Define

$$\mathsf{id}_{\mathbf{B}_0} = (ID_1, ID_2, \ldots, ID_D),$$

where $ID_d (1 \leq i \leq D)$ is the node representing $[s_d, t_d]$ in the d^{th} dimension.

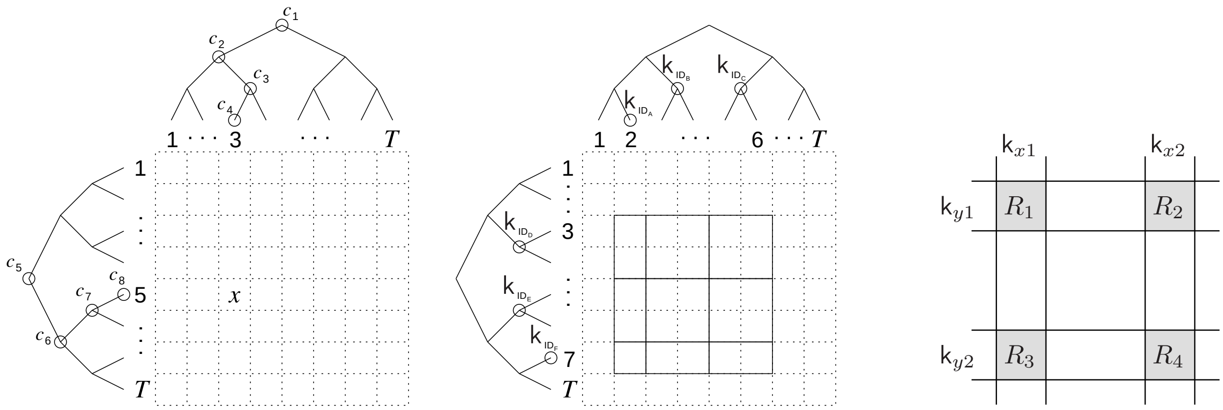
Definition 6 (Hyper-rectangle as a collection of simple hyper-rectangles)**.** *Given an hyper-rectangle* $\mathbf{B}(s_1, t_1, \ldots, s_D, t_D)$*, denote* $\Lambda_d(\mathbf{B}) := \Lambda(s_d, t_d)$ *for* $d \in [D]$*.* $\Lambda(\mathbf{B})$ *is the collection of nodes representing range* $[s_d, t_d]$ *in the* d^{th} *dimension. The hyper-rectangle* $\mathbf{B}$ *can be represented as a collection* $\Lambda^\times(\mathbf{B})$ *of simple hyper-rectangles:*

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

In particular, for every $\mathsf{id} \in \Lambda^\times(\mathbf{B})$*,* id *is a vector of the form* $(ID_1, ID_2, \ldots, ID_D)$*, where* ID_d $(d \in [D])$ *is a node in the tree corresponding to the* d^{th} *dimension. Therefore,* id *uniquely specifies a simple hyper-rectangle* $\mathbf{B}_0$ *in space.*

Clearly, $|\Lambda^\times(\mathbf{B})| = O\left((\log T)^D\right)$; in addition, $\Lambda^\times(\mathbf{B})$ can be efficiently computed. Given the above definitions, we briefly describe the AIBE-based MRQED^D construction. The detailed description is provided in Appendix F.

Encryption. Suppose that now we would like to encrypt a message $\mathbf{Msg}$ and the point $\mathbf{X} = (x_1, x_2, \ldots, x_D)$. We encrypt the message $\mathbf{Msg}$ under all simple hyper-rectangles that contain the point $\mathbf{X} = (x_1, x_2, \ldots, x_D)$. This is equivalent to encrypting $\mathbf{Msg}$ under the cross-product of D different paths to the root. Specifically, for $d \in [D]$, denote $\mathbb{P}_d(\mathbf{X}) := \mathbb{P}(x_d)$. $\mathbb{P}_d(\mathbf{X})$ is the path



(a) A ciphertext and a decryption key in MRQED^2.

(b) Collusion.

Figure 2: An MRQED^2 scheme. ***(a)*** *Encryption under the point $x = (3, 5)$ and the keys released for the range* $[2, 6] \times [3, 7]$. ***(b)*** *With decryption keys $\mathsf{k}_{x1}, \mathsf{k}_{y1}$ for region R_1 and $\mathsf{k}_{x2}, \mathsf{k}_{y2}$ for region R_4, regions R_2 and R_3 are compromised.*

from the root to the leaf node representing x_d in the d^{th} dimension. Define the cross-product of all D different paths to the root:

$$\mathbb{P}^{\times}(\mathbf{X}) = \mathbb{P}_1(\mathbf{X}) \times \mathbb{P}_2(\mathbf{X}) \times \ldots \times \mathbb{P}_D(\mathbf{X}).$$

Then, to encrypt $\mathbf{Msg}$ and $\mathbf{X}$, we use AIBE to encrypt $\mathbf{Msg}$ under every $\mathsf{id} \in \mathbb{P}^{\times}(\mathbf{X})$. Since $|\mathbb{P}^{\times}(\mathbf{X})| = O\left((\log T)^D\right)$, both encryption cost and ciphertext size are $O\left((\log T)^D\right)$.

Key derivation and decryption. To issue decryption keys for a hyper-rectangle $\mathbf{B}$, we issue a key for every $\mathsf{id} \in \Lambda^{\times}(\mathbf{B})$. Since $|\Lambda^{\times}(\mathbf{B})| = O\left((\log T)^D\right)$, the decryption key has size $O\left((\log T)^D\right)$. Now if $\mathbf{X} \in \mathbf{B}$, then $\mathbb{P}^{\times}(\mathbf{X}) \cap \Lambda^{\times}(\mathbf{B}) \neq \emptyset$; in addition, $\mathbb{P}^{\times}(\mathbf{X})$ and $\Lambda^{\times}(\mathbf{B})$ intersect at exactly one simple hyper-rectangle $\mathsf{id}_{\mathbf{B}_0}$, where the keys and the ciphertexts overlap. In this case, we use the key for $\mathsf{id}_{\mathbf{B}_0}$ to decrypt the ciphertext for $\mathsf{id}_{\mathbf{B}_0}$. Otherwise, if $\mathbf{X} \notin \mathbf{B}$, then $\mathbb{P}^{\times}(\mathbf{X}) \cap \Lambda^{\times}(\mathbf{B}) = \emptyset$. In this case, the security of the underlying AIBE schemes ensures the security of the MRQED^D constructions. In Appendix F, we show that the cost of decryption is also $O\left((\log T)^D\right)$.

5 Our MRQED^D Construction

In Section 4, we showed an AIBE-based MRQED^D construction with $O(1)$ public key size, $O\left((\log T)^D\right)$ encryption cost and ciphertext size, $O\left((\log T)^D\right)$ decryption key size and decryption cost. In this section, we propose a new MRQED^D construction with $O(D \log T)$ public key size, $O(D \log T)$ encryption cost and ciphertext size, $O(D \log T)$ decryption key size, and $O\left((\log T)^D\right)$ decryption cost.

5.1 Intuition

We build D interval trees over integers from 1 to T, each representing a separate dimension. Assume each tree node has a globally unique ID. In the previous section, we showed a naive construction for MRQED^D based on AIBE. The naive construction encrypts $\mathbf{Msg}$ under the $O((\log T)^D)$ simple hyper-rectangles that contain the point $\mathbf{X}$; and releases decryption keys for the $O((\log T)^D)$ simple hyper-rectangles that compose a hyper-rectangle $\mathbf{B}$. Our goal is to reduce the ciphertext size and decryption key size to $O(D \log T)$ instead. However, as we will soon explain, naively doing this introduces the *collusion attack* as shown in Figure 2 (b). Our main technical challenge, therefore, is to devise ways to secure against the collusion attack.

Reducing the ciphertext size. In other words, rather than encryption $\mathbf{Msg}$ for each simple hyper-rectangle in $\mathbb{P}^\times(\mathbf{X}) = \mathbb{P}_1(\mathbf{X}) \times \ldots \times \mathbb{P}_D(\mathbf{X})$, we would like to encrypt $\mathbf{Msg}$ for each tree node in the the union of these D different paths:

$$\mathbb{P}^\cup(\mathbf{X}) = \mathbb{P}_1(\mathbf{X}) \cup \ldots \cup \mathbb{P}_D(\mathbf{X}).$$

Reducing the decryption key size. Instead of representing an arbitrary hyper-rectangle using the collection of simple hyper-rectangles, we can represent a simple hyper-rectangle $\mathbf{B}$ as the collection of disjoint intervals over different dimensions:

Definition 7 (Hyper-rectangle as a collection of nodes)**.** *A hyper-rectangle* $\mathbf{B} \subseteq \mathbb{L}_\Delta$ *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 $\mathbf{B} \subseteq \mathbb{L}_\Delta$, $|\Lambda^\cup(\mathbf{B})| = O(D \log T)$; in addition, $\Lambda^\cup(\mathbf{B})$ can be computed efficiently.

Using the above definition, rather than releasing keys for each simple hyper-rectangle in $\Lambda^\times(\mathbf{B}) = \Lambda_1(\mathbf{B}) \times \ldots \times \Lambda_D(\mathbf{B})$, we would like to release keys for each ID in $\Lambda_1(\mathbf{B}) \cup \ldots \cup \Lambda_D(\mathbf{B})$.

Example. Figure 2 (a) is an example in two dimensions. To encrypt under the point $(3, 5)$, we find the path from the leaf node 3 to the root in the first dimension, and the path from the leaf node 5 to the root in the second dimension. We then produce a block in the ciphertext corresponding to each node on the two paths. In the first dimension, we produce blocks c_1, c_2, c_3 and c_4. In the second dimension, we produce blocks c_5, c_6, c_7 and c_8. To release decryption keys for the range $[2, 6] \times [3, 7]$, we find a collection $\Lambda(2, 6)$ of nodes covering the range $[2, 6]$ in the first dimension; and a collection $\Lambda(3, 7)$ of nodes covering $[3, 7]$ in the second dimension. We issue a block in the decryption key corresponding to each node in $\Lambda(2, 6)$ and in $\Lambda(3, 7)$. In the first dimension, we create blocks $\mathsf{k}_{ID_A}, \mathsf{k}_{ID_B}$, and k_{ID_C}; and in the second dimension, we create blocks $\mathsf{k}_{ID_D}, \mathsf{k}_{ID_E}$, and k_{ID_F}.

Preventing the collusion attack. Unfortunately, naively doing the above is equivalent to applying the AIBE-based MRQED^1 scheme independently in each dimension. As we demonstrate in Figure 2 (b), such a scheme is susceptible to the collusion attack. Suppose that Figure 2 (b), every rectangle is a simple rectangle. Now suppose that an adversary were given the decryption keys for region R_1 and R_4, then the adversary would have collected keys $\mathsf{k}_{R1} = \{\mathsf{k}_{x1}, \mathsf{k}_{y1}\}$,

$\mathsf{k}_{R4} = \{\mathsf{k}_{x2}, \mathsf{k}_{y2}\}$. With these, the adversary would be able to reconstruct the keys for R_2 and R_3: $\mathsf{k}_{R2} = \{\mathsf{k}_{x2}, \mathsf{k}_{y1}\}$, $\mathsf{k}_{R3} = \{\mathsf{k}_{x1}, \mathsf{k}_{y2}\}$. Hence, our major challenge is to find a way to secure against the collusion attack without incurring additional cost. We use a *binding* technique to prevent the collusion attack: we use re-randomization to tie together the sub-keys in different dimensions. For example, in Figure 2 (b), when we release the decryption key for region R_1, instead of releasing $\{\mathsf{k}_{x1}, \mathsf{k}_{y1}\}$, we release $\{\widetilde{\mu}_x \mathsf{k}_{x1}, \widetilde{\mu}_y \mathsf{k}_{y1}\}$, where $\widetilde{\mu}_x$ and $\widetilde{\mu}_y$ are random numbers that we pick each time we issue a decryption key. Likewise, when releasing the key for region R_4, we release $\{\widetilde{\mu}'_x \mathsf{k}_{x2}, \widetilde{\mu}'_y \mathsf{k}_{y2}\}$, where $\widetilde{\mu}'_x$ and $\widetilde{\mu}'_y$ are two random numbers picked independently from $\widetilde{\mu}_x$ and $\widetilde{\mu}_y$. Of course, in the real construction, $\widetilde{\mu}_x$ and $\widetilde{\mu}_y$ ($\widetilde{\mu}'_x$ and $\widetilde{\mu}'_y$) also need to satisfy certain algebraic properties (e.g., $\widetilde{\mu}_x\widetilde{\mu}_y = \widetilde{\mu}'_x\widetilde{\mu}'_y =$ some invariant) to preserve the internal consistency of our scheme. In this way, components in the decryption key for R_1 cannot be used in combination with components in the decryption key for R_4.

5.2 The Main Construction

We are now ready to describe our construction. Define $L = O(\log T)$ to represent the height of a tree. Assume that node IDs are picked from $\mathbb{Z}_p^*$. We append a message $\mathbf{Msg} \in \{0,1\}^m$ with a series of trailing zeros, $0^{m'}$, prior to encryption. Assume that $\{0,1\}^{m+m'} \subseteq \mathbb{G}'$.

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

1. Select at random the following parameters from $\mathbb{Z}_p^{8DL+1}$:

$$\omega, \left[\alpha_{\varphi,1}, \alpha_{\varphi,2}, \beta_{\varphi,1}, \beta_{\varphi,2}, \theta_{\varphi,1}, \theta_{\varphi,2}, \theta'_{\varphi,1}, \theta'_{\varphi,2}\right]_{\substack{\varphi=(d,l)\\ \in[D]\times[L]}}$$

   In addition, we require that the α's and the β's be forcibly non-zero. 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.

2. Publish $\mathbf{G}$ and the following public parameters $\mathbf{PK} \in \mathbb{G}' \times \mathbb{G}^{8DL}$:

$$\begin{array}{l} \Omega \leftarrow \mathbf{e}(g,g)^{\omega}, \\ \left[\begin{array}{l} a_{\varphi,1} \leftarrow g^{\alpha_{\varphi,1}\theta_{\varphi,1}},\ a_{\varphi,2} \leftarrow g^{\alpha_{\varphi,2}\theta_{\varphi,2}}, \\ a'_{\varphi,1} \leftarrow g^{\alpha_{\varphi,1}\theta'_{\varphi,1}},\ a'_{\varphi,2} \leftarrow g^{\alpha_{\varphi,2}\theta'_{\varphi,2}}, \\ b_{\varphi,1} \leftarrow g^{\beta_{\varphi,1}\theta_{\varphi,1}},\ b_{\varphi,2} \leftarrow g^{\beta_{\varphi,2}\theta_{\varphi,2}}, \\ b'_{\varphi,1} \leftarrow g^{\beta_{\varphi,1}\theta'_{\varphi,1}},\ b'_{\varphi,2} \leftarrow g^{\beta_{\varphi,2}\theta'_{\varphi,2}}, \end{array}\right]_{\substack{\varphi=(d,l)\in\\ [D]\times[L]}} \end{array}$$

3. Retain a master private key $\mathbf{SK} \in \mathbb{G}^{8DL+1}$ comprising the following elements:

$$\widetilde{\omega} \leftarrow g^{\omega},$$
$$\begin{bmatrix} \mathsf{a}_{\varphi,1} \leftarrow g^{\alpha_{\varphi,1}}, & \mathsf{a}_{\varphi,2} \leftarrow g^{\alpha_{\varphi,2}}, \\ \mathsf{b}_{\varphi,1} \leftarrow g^{\beta_{\varphi,1}}, & \mathsf{b}_{\varphi,2} \leftarrow g^{\beta_{\varphi,2}}, \\ y_{\varphi,1} \leftarrow g^{\alpha_{\varphi,1}\beta_{\varphi,1}\theta_{\varphi,1}}, & y_{\varphi,2} \leftarrow g^{\alpha_{\varphi,2}\beta_{\varphi,2}\theta_{\varphi,2}}, \\ y'_{\varphi,1} \leftarrow g^{\alpha_{\varphi,1}\beta_{\varphi,1}\theta'_{\varphi,1}}, & y'_{\varphi,2} \leftarrow g^{\alpha_{\varphi,2}\beta_{\varphi,2}\theta'_{\varphi,2}} \end{bmatrix}_{\substack{\varphi=(d,l) \\ \in [D]\times[L]}}$$

Notice that in the public parameters and the master key, we have different versions of the same variable, e.g., $a_{\varphi,1}, a_{\varphi,2}, a'_{\varphi,1}, a'_{\varphi,2}$. Although they seem to be redundant, they are actually needed to provide sufficient degrees of randomness for our proof to go through. The reasons for having these different versions will become clear once the reader has gone over the detailed proof provided in Appendix C.

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

1. Pick $O(D \cdot L)$ random integers from $\mathbb{G}^D \times \mathbb{Z}_p^{2|\Lambda^{\cup}(\mathbf{B})|}$:

$$\left[\widetilde{\mu}_d\right]_{d\in[D]}, \quad [\lambda_{ID,1}, \lambda_{ID,2}]_{ID\in\Lambda^{\cup}(\mathbf{B})}$$

such that $\prod_{d\in[D]} \widetilde{\mu}_d = \widetilde{\omega}$. The reason for having an overhead tilde for the variable $\widetilde{\mu}_d$ is to associate it with the variable $\widetilde{\omega}$, since they both belong to the group $\mathbb{G}$, and they satisfy the condition that $\prod_{d\in[D]} \widetilde{\mu}_d = \widetilde{\omega}$. We note that the random $\widetilde{\mu}_d$'s generated in this stage are later used to re-randomize the components of the decryption key. In this way, components in different dimensions are tied to each other; and components from one decryption key cannot be used in combination with components from another decryption key. This is how we prevent the collusion attack as shown in Figure 2 (b).

2. Compute and release a decryption key $\mathbf{DK} \in \mathbb{G}^{5|\Lambda^{\cup}(\mathbf{B})|}$. $\mathbf{DK}$ is composed of a portion $\mathbf{DK}(\mathbf{ID})$ for each $ID \in \Lambda^{\cup}(\mathbf{B})$. In the following definition for $\mathbf{DK}(ID)$, $\varphi = (d,l) = \Phi(ID)$ represents the dimension and depth of node ID; without risk of ambiguity, denote $\lambda_1 = \lambda_{ID,1}, \lambda_2 = \lambda_{ID,2}$. $\mathbf{DK}(ID)$ is defined below:

$$\widetilde{\mu}_d\big(y_{\varphi,1}^{ID} y'_{\varphi,1}\big)^{\lambda_1}\big(y_{\varphi,2}^{ID} y'_{\varphi,2}\big)^{\lambda_2}, \;\; \mathsf{a}_{\varphi,1}^{-\lambda_1}, \; \mathsf{b}_{\varphi,1}^{-\lambda_1}, \; \mathsf{a}_{\varphi,2}^{-\lambda_2}, \; \mathsf{b}_{\varphi,2}^{-\lambda_2}$$

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 $O(L^D)$. Also observe that we multiply the first element of $\mathbf{DK}(ID)$ by $\widetilde{\mu}_d$. This illustrates the *binding* technique used to tie together components in different dimensions. In this way, components in one decryption key cannot be used in combination with components in another decryption key; therefore, we successfully prevent the collusion attack.

$\mathbf{Encrypt}(\mathbf{PK}, \mathbf{X}, \mathbf{Msg})$ We create a block in the ciphertext for every $ID \in \mathbb{P}^{\cup}(\mathbf{X})$. Equivalently, for each dimension d and depth l, denote $\varphi = (d, l)$, we create a portion of the ciphertext corresponding to the node $\mathcal{I}_\varphi$, residing in the d^{th} tree at depth l, on the path $\mathbb{P}_d(\mathbf{X})$ to the root. We now describe the $\mathbf{Encrypt}$ algorithm in the following steps:

1. Select $2DL + 1$ random integers: select $r \in_R \mathbb{Z}_p$, select $[r_{\varphi,1}, r_{\varphi,2}]_{\varphi=(d,l)\in[D]\times[L]} \in_R \mathbb{Z}_p^{2DL}$.

2. For $\varphi = (d, l) \in [D] \times [L]$, define $\mathcal{I}_\varphi = \mathcal{I}_\varphi(\mathbf{X})$, i.e., the node at depth l in $\mathbb{P}_d(\mathbf{X})$ in the d^{th} dimension. Now compute and output the following ciphertext $\mathbf{C} \in \mathbb{G}' \times \mathbb{G}^{4DL+1}$:

$$\begin{matrix} (\mathbf{Msg}||0^{m'}) \cdot \Omega^{-r}, \;\; g^r, \\ \left[\begin{matrix} \left(b_{\varphi,1}{}^{\mathcal{I}_\varphi} b'_{\varphi,1}\right)^{r_{\varphi,1}}, \;\left(a_{\varphi,1}{}^{\mathcal{I}_\varphi} a'_{\varphi,1}\right)^{r-r_{\varphi,1}}, \\ \left(b_{\varphi,2}{}^{\mathcal{I}_\varphi} b'_{\varphi,2}\right)^{r_{\varphi,2}}, \;\left(a_{\varphi,2}{}^{\mathcal{I}_\varphi} a'_{\varphi,2}\right)^{r-r_{\varphi,2}} \end{matrix} \right]_{\substack{\varphi=(d,l)\in \\ [D]\times[L]}} \end{matrix}$$

$\mathbf{QueryDecrypt}(\mathbf{PK}, \mathbf{DK}, \mathbf{C})$ We first give an overview on how $\mathbf{QueryDecrypt}$ works. Recall that a decryption key $\mathbf{DK} = \left\{\mathbf{DK}(ID) \,\middle|\, ID \in \Lambda^{\cup}(\mathbf{B})\right\}$ is composed of a portion $\mathbf{DK}(ID)$ for each $ID \in \Lambda^{\cup}(\mathbf{B})$. We now reconstruct a decryption key for each simple hyper-rectangle $\mathsf{id}_{\mathbf{B}_0} \in \Lambda^{\times}(\mathbf{B})$ as below. We grab from $\mathbf{DK}$ a sub-key from each dimension: for each $d \in [D]$, grab a sub-key $\mathbf{DK}(ID_d)$ from the d^{th} dimension, where $ID_d \in \Lambda_d(\mathbf{B})$. The collection of sub-keys $\{\mathbf{DK}(ID_1), \mathbf{DK}(ID_2), \ldots, \mathbf{DK}(ID_D)\}$ can now be jointly used to decrypt a message encrypted under the simple hyper-rectangle $\mathsf{id}_{\mathbf{B}_0} = (ID_1, \ldots, ID_D)$.

We also need to find the correct blocks in the ciphertext to apply this key for $\mathsf{id}_{\mathbf{B}_0}$. Recall that the ciphertext is of the form $\mathbf{C} = \Big(c, c_0, [c_{\varphi,1}, c_{\varphi,2}, c_{\varphi,3}, c_{\varphi,4}]_{\varphi=(d,l)\in[D]\times[L]}\Big)$. For convenience, denote $c_\varphi := [c_{\varphi,1}, c_{\varphi,2}, c_{\varphi,3}, c_{\varphi,4}]$ for $\varphi = (d, l) \in [D] \times [L]$. c_φ is the block in the ciphertext corresponding to a node in the d^{th} dimension and at depth l of the tree. Define $\Phi(ID) := (d, l)$ to extract the dimension and depth of the node ID. Now for a sub-key $\mathbf{DK}(ID)$, define $\varphi = \Phi(ID)$, it is not hard to see that $\mathbf{DK}(ID)$ should be used in combination with the block c_φ in the ciphertext.

The following algorithm iterates through the simple hyper-rectangles in $\Lambda^{\times}(\mathbf{B})$ and checks if the ciphertext can decrypt to a valid message under each simple hyper-rectangle in $\Lambda^{\times}(\mathbf{B})$.

For each simple hyper-rectangle $\Lambda^{\times}(\mathbf{B}_0) = \{(ID_1, ID_2, \ldots, ID_D)\} \subseteq \Lambda^{\times}(\mathbf{B})$,

(1) Let $\mathbf{DK}(ID_d) = (\mathsf{k}_{ID_d,0}, \mathsf{k}_{ID_d,1}, \mathsf{k}_{ID_d,2}, \mathsf{k}_{ID_d,3}, \mathsf{k}_{ID_d,4})$ represent the element in $\mathbf{DK}$ for ID_d, where $d \in [D]$.

(2) Try to decrypt $\mathbf{C}$ under $\mathbf{B}_0$ with the collection $\{\mathbf{DK}(ID_1), \mathbf{DK}(ID_2), \ldots, \mathbf{DK}(ID_D)\}$ of sub-keys:

$$V \leftarrow c \cdot \prod_{\substack{d\in[D], \\ \varphi_d=\Phi(ID_d)}} \left[\mathbf{e}(c_0, \mathsf{k}_{ID_d,0}) \cdot \mathbf{e}(c_{\varphi_d,1}, \mathsf{k}_{ID_d,1}) \cdot \mathbf{e}(c_{\varphi_d,2}, \mathsf{k}_{ID_d,2}) \cdot \mathbf{e}(c_{\varphi_d,3}, \mathsf{k}_{ID_d,3}) \cdot \mathbf{e}(c_{\varphi_d,4}, \mathsf{k}_{ID_d,4})\right]$$

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

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 $\mathbf{QueryDecrypt}$ 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. The above numbers takes into account all group operations, include multiplication, exponentiation and bilinear pairing. However, since a pairing operation is typically more expensive than exponentiation (and far more expensive than multiplication) in known bilinear groups, we are particularly interested in reducing the number of pairings at time of decryption. Notice that we can precompute all pairings $\mathbf{e}(c_0, \mathsf{k}_{ID_d,0})$ and pairings $\mathbf{e}(c_{\varphi_d,i}, \mathsf{k}_{ID_d,i})$ for $1 \leq i \leq 4$, and store the results in a look-up table. Therefore, the decryption algorithm requires $O(D \log T)$ pairings in total.

6 Consistency, Security

The following two theorems state the consistency and security of our MRQED construction.

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

Theorem 6.2 (Selective security)**.** *The above defined MRQED construction is selectively secure against polynomial-time adversaries.*

Below we give an overview of the techniques used in the security proof. The detailed proofs of Theorem 6.1 and Theorem 6.2 are provided in Appendix C and Appendix B respectively. To prove the selective security of our MRQED^D construction, we decompose the selective MRQED game into two games: a selective confidentiality game and a selective anonymity game. By the hybrid argument, if no polynomial-time adversary has more than negligible advantage in either the confidentiality game or the anonymity game, then no polynomial-time adversary has more than negligible advantage in the combined selective MRQED game.

In the proof, we build a simulator that leverages an MRQED adversary to solve the D-BDH problem or the D-Linear problem. The simulator inherits parameters specified by the D-BDH/D-Linear instance, hence, it has incomplete information about the master key. Therefore, the crux of the proof is how to simulate the key derivation algorithm without knowing the complete master key. 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_{φ_1,n_1} and y'_{φ_1,n_1} ($\varphi_1 = (d_1, l_1)$) in the master key contain 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. Our proof techniques are similar to that presented in the AHIBE paper [15].

7 Practical Performance

In this section, we give a detailed analysis of the performance of the MRQED^D scheme given in Section 5 in practical scenarios. We use the conditional release of encrypted network audit logs as our motivating application.

Assumptions. To evaluate the scheme of Section 5 in this application, we detail a set of scenarios regarding the searchable fields present in the logs. We assume log entries contain the fields listed in Table 2. The 17-bit time field is sufficient to distinguish times over a period of about 15 years with a one hour resolution, or about three months at a one minute resolution. More precise times may be stored in the non-searchable portion of the message if desired. The protocol field corresponds

Field	Abbr.	Range	Distinct Values
Source IP	sip	$[0, T_{\mathsf{sip}} - 1]$	$T_{\mathsf{sip}} = 2^{32}$
Dest. IP	dip	$[0, T_{\mathsf{dip}} - 1]$	$T_{\mathsf{dip}} = 2^{32}$
Port	port	$[0, T_{\mathsf{port}} - 1]$	$T_{\mathsf{port}} = 2^{16}$
Time	time	$[0, T_{\mathsf{time}} - 1]$	$T_{\mathsf{time}} = 2^{17}$
Protocol	prot	$[0, T_{\mathsf{prot}} - 1]$	$T_{\mathsf{prot}} = 2^{8}$

Table 2: Fields appearing in a network audit log and their possible values.

to the actual bits of the corresponding field in an IP header (where, for example, 6 denotes TCP and 133 denotes Fibre Channel). Various subsets of these fields may be included as searchable attributes in MRQED^D. Other fields and any additional associated data such as a payload may be included as the encrypted message. Regardless of message length, we need only use the MRQED^D scheme to encrypt a single group element, which may be a randomly generated symmetric key (e.g., for AES) used to encrypt the message.

Benchmarks for the selected pairing were run on a modern workstation. The processor was a 64-bit, 3.2 Ghz Pentium 4. We used the Pairing-Based Cryptography (PBC) library [34], which is in turn based on the GNU Multiple Precision Arithmetic Library (GMP). The relevant results are given in Table 3. Using these benchmark numbers, we now estimate the performance of our encryption scheme under several scenarios for the network audit log application.

Operation	Time
pairing (no preprocessing)	5.5 ms
pairing (after preprocessing)	2.6 ms
preprocess pairing	5.9 ms
exponentiation in $\mathbb{G}, \widehat{\mathbb{G}}$	6.4 ms
exponentiation in $\mathbb{G}'$	0.6 ms
multiplication in $\mathbb{G}'$	5.1 μs

Table 3: Group arithmetic and pairing performance benchmarks on a modern workstation [34].

Public parameters and master key. The space required to store the public parameters and master key is logarithmic with respect to the number of possible attribute values. Specifically, denote the set of attributes as $A = \{\mathsf{sip}, \mathsf{dip}, \mathsf{port}, \mathsf{time}, \mathsf{prot}\}$. Then for each attribute $a \in A$, define the height of the tree $L_a = \log_2 T_a + 1$. For example, $L_{\mathsf{sip}} = 33$ and $L_{\mathsf{prot}} = 9$. Then the public parameters $\mathbf{PK}$ require a total of $8 \sum_{a \in A} L_a = 880$ elements of $\mathbb{G}$ and one element of $\mathbb{G}'$. Assuming

512-bit representations[2] of elements of $\mathbb{G}$ and $\mathbb{G}'$, the total size of $\mathbf{PK}$ is 55KB. The master key $\mathbf{SK}$ contains the same number of elements, again requiring 55KB of storage. More space efficient pairings than the one used in this estimate are available, but this one was selected for speed of evaluation.

Computation time for $\mathbf{Setup}$ is reasonable, given that it is only run once. Computing the public and private parameters in $\mathbf{Setup}$ requires roughly $16 \sum_{a \in A} L_a$ exponentiations and one pairing, for a total of about 11.3s. Time spent on multiplication in this case is negligible.

Encryption. Saving the group elements of a ciphertext requires $4 \sum_{a \in A} L_a + 2$ group elements, or 28KB. Note that we normally just encrypt a session key, so this is a constant overhead beyond the actual length of the message. Running $\mathbf{Encrypt}$ requires about two exponentiations for each group element, resulting in a time of about 5.6s. While significant, this overhead should be acceptable in most cases in the network audit log example. If audit logs are high volume, the best strategy may be to produce periodic summaries rather than separately encrypting each packet. The searchable attributes of such summaries would reflect the collection of entries they represent, and the full contents of the entries could be included as the encrypted message without incurring additional overhead. In systems containing a cryptographic accelerator chip supporting ECC (such as some routers), much higher performance is possible. For example, the Elliptic Semiconductor CLP-17 could reduce the time of exponentiation from 6.4ms to 30μs [19], resulting in a total encryption time as low as 27ms.

Key derivation and decryption. We now consider decryption keys and the running time of the decryption algorithm, the more interesting aspects of the scheme's computational and storage requirements. The space required to store a decryption key, the time to derive it, and the time to decrypt using it depend only on the ranges of attributes for which it permits decryption. Unlike the computational and storage requirements discussed thus far, these costs do not depend on the full range of possible values, only those associated with the key. These costs depend on the number of key components necessary to represent the permissible range along each dimension. For example, suppose a particular decryption key $\mathbf{DK}$ only allows decryption of entries with a destination port in the range $[3, 7]$ (perhaps placing other requirements on the other attributes). Referring back to Figure 1, we see that three tree nodes are necessary to cover this range, so $\mathbf{DeriveKey}$ would include these three for the destination port dimension in $\mathbf{DK}$. Similarly, given some decryption key $\mathbf{DK}$, we denote the number of tree nodes necessary to cover the decryption range in each of the dimensions $a \in A$ by $N_a = |\Lambda_a(\mathbf{B})|$ (using the notation of Section 5). So in this example, $N_{\mathsf{port}} = 3$. Note that for any $a \in A$, in the worst case, $N_a = 2L_a - 2$.

Now given N_a for each $a \in A$, we may compute the decryption costs. A decryption key consists of $5 \sum_{a \in A} N_a$ group elements and $\mathbf{DeriveKey}$ performs $8 \sum_{a \in A} N_a$ exponentiations. The number of operations necessary to decrypt using a key $\mathbf{DK}$ is slightly more subtle. While $\mathbf{QueryDecrypt}$ is $\Theta(\prod_{a \in A} L_a)$ (i.e., $\Theta((\log T)^D)$) overall, only $O(\sum_{a \in A} L_a)$ (i.e., $O(D \log T)$)

[2]We consider a type A pairing using the singular curve $y^2 = x^3 + x$ for the groups $\mathbb{G}$ and $\widehat{\mathbb{G}}$ with a base field size of 512-bits. Note that all groups involved have 160-bit group order; the storage requirements arise from the specific representation of elements in the elliptic curves.

pairings are required, as mentioned in Section 5.2. Specifically, we need only compute $5\sum_{a\in A} N_a$ pairings to populate a lookup table containing values of $\mathbf{e}(c_0, \mathsf{k}_{ID,0})$, $\mathbf{e}(c_{\varphi,1}, \mathsf{k}_{ID,1})$, $\mathbf{e}(c_{\varphi,2}, \mathsf{k}_{ID,2})$, $\mathbf{e}(c_{\varphi,3}, \mathsf{k}_{ID,3})$, $\mathbf{e}(c_{\varphi,4}, \mathsf{k}_{ID,4})$, and $\mathbf{e}(c_{\varphi,5}, \mathsf{k}_{ID,5})$. These values are enough to complete the $\mathbf{QueryDecrypt}$ algorithm. Assuming a key will normally be used to decrypt a batch of ciphertexts one after another, we may further reduce the cost of pairings by preprocessing with the key. As shown in Table 3, preprocessing reduces the pairing time by about half, at a one time cost (per decryption key $\mathbf{DK}$) equivalent to one or two decryptions. Computed naively, the sequence of trials in step one of $\mathbf{QueryDecrypt}$ end up requiring a total of $|A|\prod_{a\in A} N_a$ multiplications in $\mathbb{G}'$. This can be somewhat reduced. Let $S_1, \dots S_{|A|}$ be $\{\, N_a \,|\, a \in A \,\}$ sorted in ascending order: $S_1 \leq S_2 \leq \dots S_{|A|}$. Then by saving intermediate results between trials and ordering the dimensions appropriately, it is possible to complete step one with a total of $S_1 + S_1S_2 + S_1S_2S_3 + \dots S_1S_2 \cdots S_{|A|}$ multiplications.

Specific scenarios. We have now computed the costs associated with the storage and usage of a decryption key in terms of N_a for $a \in A$, but we have not yet specified N_a. If we assume the range for each attribute is randomly selected (uniformly), then for each $a \in A$, the expected value of N_a is $L_a - 1$. This results in a decryption key size of 33KB and a running time for $\mathbf{DeriveKey}$ of 5.4s. The corresponding worst-case decryption time[3] is 13.1s. We note that this is a major cost, and likely to be inconvenient if significant quantities of log entries must be decrypted. Fortunately, queries eliciting such long decryption times are not likely to be necessary in practice. In fact, fairly elaborate queries are possible while keeping decryption costs low.

In Table 4 we provide several examples that help demonstrate this. The first entry illustrates the fact that specifying a single value, all values, or a range of values falling on power-of-two boundaries (as in the case of an IP subnet) for some attribute a results in $N_a = 1$, reducing decryption time dramatically. In the next example, several attributes are required to be in general ranges, or, in the case of prot, selected from a small set. This results in larger numbers of key components and slightly longer decryption times. Still, the decryption time in this case is far below the time with each range randomly selected. As shown by the third example, larger ranges result in larger values of N_a and, again, somewhat larger, but still relatively low, decryption times.

8 Extensions and Discussions

8.1 The Dual Problem and Stock Trading through a Broker

In the MRQED problem, one encrypts a message $\mathbf{Msg}$ under a point $\mathbf{X}$ in multi-dimensional space, and given a hyper-rectangle $\mathbf{B}$, the master key owner can construct a capability, allowing an auditor to decrypt all entries satisfying $\mathbf{X} \in \mathbf{B}$. On the other hand, the privacy of the irrelevant entries are still preserved.

Informally, the natural dual problem to MRQED is where one encrypts under a hyper-rectangle $\mathbf{B}$, and given a point $\mathbf{X}$, the master key owner can construct a capability allowing an auditor to

[3] In reality, the average decryption time is smaller than this number, since upon a successful decryption, the $\mathbf{QueryDecrypt}$ algorithm exits after trying half of the combinations in expectation and thus performing half the worst-case multiplications.

Example Query	N_{sip}	N_{dip}	N_{port}	N_{time}	N_{prot}	Pairing Time	Worst-case Mult. Time	Worst-case Dec. Time
$\mathsf{sip} = 207.44.178.*$, $\mathsf{dip} = 216.187.103.169$, $\mathsf{port} = 22$, $\mathsf{time} = *$, $\mathsf{prot} = \text{TCP}$	1	1	1	1	1	65ms	< 0.1ms	65ms
$\mathsf{sip} \in [207.44.178.123, 207.44.182.247]$, $\mathsf{dip} = *$, $\mathsf{port} = 22$, $\mathsf{time} \in$ [5pm 10/31, 9am 11/5], $\mathsf{prot} \in$ {TCP, UDP, ICMP}	10	1	1	7	3	286ms	1.2ms	287ms
$\mathsf{sip} \in [207.44.178.123, 207.60.177.15]$, $\mathsf{dip} \in [207.44.178.123, 207.60.177.15]$, $\mathsf{port} \in [3024, 35792]$, $\mathsf{time} \in$ [10/31/2006, 10/31/2020], $\mathsf{prot} \in$ {TCP, UDP, ICMP}	20	20	15	17	3	0.98s	1.64s	2.62s

Table 4: Decryption times resulting from decryption keys of various sizes.

decrypt all entries satisfying $\mathbf{B} \ni \mathbf{X}$. Like in MRQED, we require that the privacy of all irrelevant entries be preserved. We now show an interesting application of the dual problem, and then show that MRQED implies a solution for the dual problem.

An interesting application of the dual problem is for trading stocks and other securities. Suppose an *investor* trades stocks through a *broker*. The investor specifies a price range and a time range, such that if the stock price falls within that range during a specific period of time, the broker can buy or sell the stock on behalf of the investor. This is usually referred to as a *stop order*, *limit order*, or *stop-limit order*. Sometimes, the investor may not fully trust the broker, and may wish to conceal the price and time ranges from the broker before an order is executed.

The dual problem can be applied in such scenarios to address the privacy concerns of investors. In particular, the *stock exchange*, or any third-party with knowledge of the real-time stock price can act as the trusted authority who owns the master key. For convenience, in the following description, we assume that the *stock exchange* is the trusted authority. The investor first encrypts the order along with the desired price and time ranges, and sends the encrypted order to the broker. Suppose that at a certain point of time t, the stock price is p. The stock exchange constructs a decryption key for the pair (t, p), and hands it to the broker. With this decryption key, the broker can decrypt all orders whose price and time ranges match the current price p and the current time t, and execute these orders. For orders whose price and time ranges do not match the current price and time, the broker cannot learn any additional information about these orders.

MRQED implies the dual problem. We use a two-dimensional example to illustrate how MRQED implies a solution for the dual problem.

- Dual.$\mathbf{Setup}$ $(\Sigma, [T]^2)$: Call MRQED.$\mathbf{Setup}$ $(\Sigma, [T]^4)$, and output the public key $\mathbf{PK}$, and master key $\mathbf{SK}$.

- Dual.$\mathbf{Encrypt}$ $(\mathbf{PK}, [x_1, x_2] \times [y_1, y_2], \mathbf{Msg})$: To encrypt a message $\mathbf{Msg}$ under the range

$[x_1, x_2] \times [y_1, y_2]$ in 2 dimensions, call MRQED.$\mathbf{Encrypt}$ $(\mathbf{PK}, (x_1, x_2, y_1, y_2), \mathbf{Msg})$. Observe that here a range $[x_1, x_2] \times [y_1, y_2]$ in $[T]^2$ is mapped to a point (x_1, x_2, y_1, y_2) in $[T]^4$.

- Dual.$\mathbf{DeriveKey}$ $(\mathbf{PK}, \mathbf{SK}, (x, y))$: To generate a decryption key for the point $(x, y) \in [T]^2$, call MRQED.$\mathbf{DeriveKey}$ $(\mathbf{PK}, \mathbf{SK}, [1, x] \times [x, T] \times [1, y] \times [y, T])$.

- Dual.$\mathbf{QueryDecrypt}$ $(\mathbf{PK}, \mathbf{DK}, \mathbf{C})$: To try to decrypt a ciphertext $\mathbf{C}$ using the decryption key $\mathbf{DK}$, call MRQED.$\mathbf{QueryDecrypt}$ $(\mathbf{PK}, \mathbf{DK}, \mathbf{C})$.

In essence, the above construction maps a range $[x_1, x_2] \times [y_1, y_2] \subseteq [T]^2$ to a point $(x_1, x_2, y_1, y_2) \in [T]^4$, and testing if a point (x, y) is within the range $[x_1, x_2] \times [y_1, y_2]$ is equivalent to testing whether $(x_1, x_2, y_1, y_2) \in [1, x] \times [x, T] \times [1, y] \times [y, T]$. It is easy to verify that the security of the MRQED scheme guarantees a similar notion of security for the dual construction, i.e., if a decryption key fails to decrypt a certain ciphertext entry, then a probablistic polynomial adversary cannot learn any additional information about that entry.

8.2 Adaptive 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. We present the formal definition for MRQED adaptive-ID security in Appendix D . Previous research has shown that IBE schemes secure in the selective-ID sense can be converted to schemes fully secure [6, 18, 45, 36] with some loss in security. In particular, Boneh and Boyen prove the following theorem:

Theorem 8.1 ([6]). *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.

9 Conclusion

We design an encryption scheme that allows us to encrypt an arbitrary message and a set of attributes. An authority holding a master key can issue a search capability to an authorized party, allowing it to decrypt data entries whose attributes fall within specific ranges; while the privacy of other data entries is preserved. We prove the security of our scheme under the D-BDH and the D-Linear assumptions in certain bilinear groups. We also study the practical performance of our construction in network audit log applications. Apart from network audit logs, MRQED can be useful in various other applications such as financial audit logs, untrusted email servers and medical privacy. In particular, we show that the dual problem can be useful for investors who wish to trade stocks through a broker in a privacy-preserving manner.

10 Acknowledgments

We would especially like to thank Brent Waters, Dan Boneh, Matthew Wachs, and Eno Thereska for their valuable suggestions on how to improve the paper. We also thank the anonymous reviewers for their insightful comments.

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] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order preserving encryption for numeric data. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 563–574, 2004.

[3] Giuseppe Ateniese, Marina Blanton, and Jonathan Kirsch. Secret handshakes with dynamic and fuzzy matching. In *Network and Distributed System Security Symposium*, 2007.

[4] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *ASIACRYPT '01: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security*, pages 566–582, 2001.

[5] John Bethencourt, Dawn Song, and Brent Waters. New constructions and practical applications for private stream searching (extended abstract). In *SP '06: Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P'06)*, pages 132–139, 2006.

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

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

[8] 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.

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

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

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

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

[13] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. To appear in the Theory of Cryptography Conference (TCC), 2007.

[14] Dan Boneh and Brent Waters. A fully collusion resistant broadcast, trace and revoke system. In *CCS*, 2006.

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

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

[17] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT*, pages 207–222, 2004.

[18] 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.

[19] The Elliptic Semiconductor CLP-17 high performance elliptic curve cryptography point multiplier core: Product brief.
`http://www.ellipticsemi.com/pdf/CLP-17_60102.pdf`.

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

[21] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. In *CCS*, 2006.

[22] Ernesto Damiani, S. De Capitani Vimercati, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati. Balancing confidentiality and efficiency in untrusted relational dbmss. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 93–102, 2003.

[23] D. Davis, F. Monrose, and M. K. Reiter. Time-scoped searching of encrypted audit logs. In *Proceeding of the International Conference on Information and Communications Security (ICICS)*, 2004.

[24] Symantec deepsight threat management system technology brief. `https://tms.symantec.com`.

[25] The dshield project. `http://www.dshield.org`.

[26] 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.

[27] Oded Goldreich. Secure multi-party computation. Volume 2, Foundations of Cryptography, 1998.

[28] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, 1996.

[29] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98, 2006.

[30] Hakan Hacigumus, Bala Iyer, Chen Li, and Sharad Mehrotra. Executing sql over encrypted data in the database-service-provider model. In *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 216–227, 2002.

[31] 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.

[32] 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.

[33] Patrick Lincoln, Phillip A. Porras, and Vitaly Shmatikov. Privacy-preserving sharing and correlation of security alerts. In *USENIX Security Symposium*, pages 239–254, 2004.

[34] Ben Lynn. The Pairing-Based Cryptography (PBC) library. `http://crypto.stanford.edu/pbc`.

[35] The mynetwatchman project. `http://www.mynetwatchman.com`.

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

[37] Rafail Ostrovsky. Software protection and simulation on oblivious RAMs. Ph.D. thesis, MIT, 1992. Preliminary version in STOC 1990.

[38] Rafail Ostrovsky and William E. Skeith III. Private searching on streaming data. In *CRYPTO*, pages 223–240, 2005.

[39] Vern Paxson. Bro: A system for detecting network intruders in real-time. In *USENIX Security Symposium*, 1998.

[40] P. Porras and P. Neumann. EMERALD: Event monitoring enabling responses to anomalous live disturbances. In *National Information Systems Security Conference*, 1997.

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

[42] 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.

[43] The Snort open source network intrusion detection system. `http://www.snort.org`.

[44] 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.

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

[46] 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 (NDSS)*, San Diego, CA, February 2004.

[47] 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 Notations

In Table 5, we summarize the notations used throughout this paper.

Notation	Explanation	First Defined
$[s, t]$	integers s through t	Sec. 3
$[a]$	integers 1 through a	Sec. 3
D	number of dimensions	Sec. 3
T	number of discrete values in each dimension	Sec. 3
$\mathbb{L}_\Delta$	multi-dimensional lattice	Sec. 3
$\mathbf{X}$	a point in the lattice	Sec. 3
$\mathbf{B}$	a hyper-rectangle	Sec. 3
Σ	security parameter	Sec. 3
$\mathbf{PK}$	public key	Sec. 3
$\mathbf{SK}$	master key	Sec. 3
$\mathbf{DK}$	decryption key	Sec. 3
$\mathbf{Msg}$	message to encrypt	Sec. 3
$\mathbb{M}$	message space	Sec. 3
$\mathbf{G}$	a bilinear instance	Sec. 3.3
$\mathbb{G}$	bilinear group	Sec. 3.3
$\mathbb{G}'$	target group	Sec. 3.3
e	bilinear pairing function	Sec. 3.3
g	generator of $\mathbb{G}$	Sec. 3.3
$\mathbb{Z}_p$	additive group of integers modular a prime p	Sec. 3.3
$\mathbb{Z}_p^*$	multiplicative group of integers modular a prime p	Sec. 5.2
$\mathsf{tr}(T)$	binary interval tree over integers 1 through T	Sec. 4.2
ID	identity of a tree node	Sec. 4.2
$\mathsf{cv}(ID)$	range represented by a tree node ID	Sec. 4.2
$\mathbb{P}(x)$	path from the root to the leaf node representing x	Sec. 4.2
$\Lambda(s, t)$	set of nodes representing the range [s, t]	Sec. 4.2
$\Lambda_d(\mathbf{B})$	set of nodes representing the range specified by $\mathbf{B}$ in the d^{th} dimension	Sec. 4.3
$\mathbf{B}_0$	simple hyper-rectangle	Sec. 4.3
$\mathsf{id}_{\mathbf{B}_0}$	identity vector of the simple hyper-rectangle $\mathbf{B}_0$	Sec. 4.3
$\Lambda^\times(\mathbf{B})$	hyper-rectangle $\mathbf{B}$ as a collection of simple hyper-rectangles	Sec. 4.3
$\mathbb{P}_d(\mathbf{X})$	path to root in the d^{th} dimension for the point $\mathbf{X}$	Sec. 4.3
$\mathbb{P}^\times(\mathbf{X})$	cross-product of all D paths to root for the point $\mathbf{X}$	Sec. 4.3
$\mathbb{P}^\cup(\mathbf{X})$	union of all D paths to root for the point $\mathbf{X}$	Sec. 5.1
$\Lambda^\cup(\mathbf{B})$	hyper-rectangle $\mathbf{B}$ as a set of tree nodes	Sec. 5.1
L	height of interval tree	Sec. 5.2
$\Phi(ID)$	a function that outputs the dimension and depth of some node ID	Sec. 5.2
$\varphi = (d, l)$	usually used in subscripts to indicate the dimension and depth respectively	Sec. 5.2
$\mathcal{I}_\varphi(\mathbf{X})$ where $\varphi = (d, l)$	the node at depth l in the path $\mathbb{P}_d(\mathbf{X})$ of the d^{th} dimension	Sec. 5.2

Table 5: Notations.

B Proof of Consistency

Proof of Theorem 6.1:

Let $\mathbf{C} = \Big(c, c_0, [c_{\varphi,1}, c_{\varphi,2}, c_{\varphi,3}, c_{\varphi,4}]_{\varphi=(d,l)\in[D]\times[L]}\Big)$ 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 $d \in [D]$, $\mathcal{I}_{\varphi_d}(\mathbf{X}) = ID_d$. For simplicity, let $\xi(x) = \mathbf{e}(g,g)^x$, and denote $\mathcal{I}_{\varphi_d} = \mathcal{I}_{\varphi_d}(\mathbf{X})$. Now decryption for $\mathbf{B_0}$ proceeds as follows:

$$
\begin{aligned}
V =& (\mathbf{Msg}||0^{m'}) \cdot \Omega^{-r} \cdot \prod_{d\in[D]} \mathbf{e}\Big(g^r, \widetilde{\mu}_d \big({y_{\varphi_d,1}}^{ID_d} y'_{\varphi_d,1}\big)^{\lambda_{ID_d,1}} \big({y_{\varphi_d,2}}^{ID_d} y'_{\varphi_d,2}\big)^{\lambda_{ID_d,2}}\Big) \\
&\cdot \prod_{d\in[D],n\in[2]} \mathbf{e}\Big({\mathsf{a}_{\varphi_d,n}}^{-\lambda_{ID_d,n}}, \big({b_{\varphi_d,n}}^{\mathcal{I}_{\varphi_d}} b'_{\varphi_d,n}\big)^{r_{\varphi_d,n}}\Big) \cdot \prod_{d\in[D],n\in[2]} \mathbf{e}\Big({\mathsf{b}_{\varphi_d,n}}^{-\lambda_{ID_d,n}}, \big({a_{\varphi_d,n}}^{\mathcal{I}_{\varphi_d}} a'_{\varphi_d,n}\big)^{r-r_{\varphi_d,n}}\Big) \\
=& (\mathbf{Msg}||0^{m'}) \cdot \Omega^{-r} \cdot \mathbf{e}\left(g^r, \widetilde{\omega}\right) \cdot \xi\left(r \cdot \sum_{\substack{d\in[D],\\ n\in[2]}} \alpha_{\varphi_d,n}\beta_{\varphi_d,n}\lambda_{ID_d,n} \left(\theta_{\varphi_d,n} ID_d + \theta'_{\varphi_d,n}\right)\right) \\
&\cdot \xi\left(\sum_{\substack{d\in[D],\\ n\in[2]}} \alpha_{\varphi_d,n}(-\lambda_{ID_d,n}) r_{\varphi_d,n}\beta_{\varphi_d,n} \left(\theta_{\varphi_d,n}\mathcal{I}_{\varphi_d} + \theta'_{\varphi_d,n}\right)\right) \\
&\cdot \xi\left(\sum_{\substack{d\in[D],\\ n\in[2]}} \beta_{\varphi_d,n}(-\lambda_{ID_d,n}) \left(r - r_{\varphi_d,n}\right) \alpha_{\varphi_d,n} \left(\theta_{\varphi_d,n}\mathcal{I}_{\varphi_d} + \theta'_{\varphi_d,n}\right)\right) \\
=& (\mathbf{Msg}||0^{m'}) \cdot \Omega^{-r} \cdot \mathbf{e}\left(g^r, \widetilde{\omega}\right) \cdot \xi\left(r \cdot \sum_{\substack{d\in[D],\\ n\in[2]}} \alpha_{\varphi_d,n}\beta_{\varphi_d,n}\lambda_{ID_d,n} \left(\theta_{\varphi_d,n} ID_d + \theta'_{\varphi_d,n}\right)\right) \\
&\cdot \xi\left(r \cdot \sum_{\substack{d\in[D],\\ n\in[2]}} \alpha_{\varphi_d,n}\beta_{\varphi_d,n}(-\lambda_{ID_d,n}) \left(\theta_{\varphi_d,n}\mathcal{I}_{\varphi_d} + \theta'_{\varphi_d,n}\right)\right) \\
=& \mathbf{Msg}||0^{m'}.
\end{aligned}
$$

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

$$\begin{aligned} V &= (\mathbf{Msg}||0^{m'}) \cdot \frac{\xi\left(r \cdot \sum_{\substack{d\in[D],\\ n\in[2]}} \alpha_{\varphi_d,n}\beta_{\varphi_d,n}\lambda_{ID_d,n}\left(\theta_{\varphi_d,n}ID_d + \theta'_{\varphi_d,n}\right)\right)}{\xi\left(r \cdot \sum_{\substack{d\in[D],\\ n\in[2]}} \alpha_{\varphi_d,n}\beta_{\varphi_d,n}\lambda_{ID_d,n}\left(\theta_{\varphi_d,n}\mathcal{I}_{\varphi_d} + \theta'_{\varphi_d,n}\right)\right)} \\ &= (\mathbf{Msg}||0^{m'}) \cdot Q^r \end{aligned}$$

where

$$Q = \xi\left(\sum_{\substack{d\in[D],\\ n\in[2]}} \alpha_{\varphi_d,n}\beta_{\varphi_d,n}\lambda_{ID_d,n}\left(\theta_{\varphi_d,n}ID_d + \theta'_{\varphi_d,n}\right) - \sum_{\substack{d\in[D],\\ n\in[2]}} \alpha_{\varphi_d,n}\beta_{\varphi_d,n}\lambda_{ID_d,n}\left(\theta_{\varphi_d,n}\mathcal{I}_{\varphi_d} + \theta'_{\varphi_d,n}\right)\right)$$

With probability $1 - 1/p$, $Q \neq 1$, and the ciphertext is distributed uniformly at random in $\mathbb{G}'$. 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 Proof of Security

To prove the selective security of our MRQED^D construction, we decompose the selective MRQED game into two games: a selective confidentiality game and a selective anonymity game. By the hybrid argument, if no polynomial-time adversary has more than negligible advantage in either the confidentiality game or the anonymity game, then no polynomial-time adversary has more than negligible advantage in the combined selective MRQED game. The terminology *confidentiality* and *anonymity* that we use here is adopted from AIBE schemes.

Definition 8 (MRQED selective confidentiality game)**.** *The MRQED selective 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 $\mathbf{Setup}(\mathbf{\Sigma}, \mathbb{L}_{\mathbf{\Delta}})$ algorithm to generate $\mathbf{PK}$, $\mathbf{SK}$. It gives $\mathbf{PK}$ to the adversary, but does not divulge $\mathbf{SK}$.
- **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 9 (MRQED selective anonymity game)**.** *The MRQED selective 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 $\mathbf{Setup}(\boldsymbol{\Sigma}, \mathbb{L}_{\boldsymbol{\Delta}})$ algorithm to generate $\mathbf{PK}, \mathbf{SK}$. It gives $\mathbf{PK}$ to the adversary, but does not divulge $\mathbf{SK}$.
- **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 10 (IND-sID-CPA)**.** *An MRQED scheme is IND-sID-CPA secure if all polynomial-time adversaries have at most a negligible advantage in the confidentiality game.*

Definition 11 (ANON-sID-CPA)**.** *An MRQED scheme is ANON-sID-CPA secure if all polynomial-time adversaries have at most a negligible advantage in the anonymity game.*

Lemma C.1. *If an MRQED scheme is both IND-sID-CPA secure and ANON-sID-CPA secure, then the MRQED scheme is selectively secure.*

Proof. By the hybrid argument. ■

Hence, it suffices to prove our MRQED construction IND-sID-CPA and ANON-sID-CPA secure. We say that an MRQED 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 C.2 (Confidentiality)**.** *Suppose* $\mathbf{G}$ *satisfies the* (τ, ϵ) *D-BDH assumption, then the above defined MRQED scheme is* (τ', q, ϵ) *IND-sID-CPA secure, where* $\tau' < \tau - \Theta(qD \log T)$.

Theorem C.3 (Anonymity)**.** *Suppose* $\mathbf{G}$ *satisfies the* (τ, ϵ) *D-Linear assumption, then the above defined MRQED 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.

■

C.1 Proof: Confidentiality

Proof of Theorem C.2:

We reduce the semantic security of MRQED 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 $Z = \mathbf{e}(g, g)^{z_1 z_2 z_3}$. And to do this, the simulator leverages an MRQED IND-sID-CPA adversary, $\mathcal{A}$.

We describe a reduction such that if $Z = \mathbf{e}(g, g)^{z_1 z_2 z_3}$, 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 $\mathbf{X}^* \in \mathbb{L}_\Delta$ 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 from ${\mathbb{Z}_p}^{12DL}$:
$$\left[\alpha_{\varphi,n}, \beta_{\varphi,n}, \theta_{\varphi,n}, \theta'_{\varphi,n}, \bar{\theta}_{\varphi,n}, \bar{\theta}'_{\varphi,n}\right]_{\varphi=(d,l)\in[D]\times[L],n\in[2]}$$
subject to the constraint that
$$\left[\bar{\theta}_{\varphi,n}\mathcal{I}^*_\varphi + \bar{\theta}'_{\varphi,n} = 0\right]_{\varphi=(d,l)\in[D]\times[L],n\in[2]}$$
where $\mathcal{I}^*_\varphi = \mathcal{I}_\varphi(\mathbf{X}^*)$. We also require that the α's, β's, $\bar{\theta}$'s and $\bar{\theta}'$'s are forcibly non-zero.

2. Release the following public parameters to the adversary.
$$\Omega \leftarrow \mathbf{e}(g_1, g_2), \begin{bmatrix} a_{\varphi,n} \leftarrow \left(g^{\theta_{\varphi,n}} {g_1}^{\bar{\theta}_{\varphi,n}}\right)^{\alpha_{\varphi,n}}, & a'_{\varphi,n} \leftarrow \left(g^{\theta'_{\varphi,n}} {g_1}^{\bar{\theta}'_{\varphi,n}}\right)^{\alpha_{\varphi,n}}, \\ b_{\varphi,n} \leftarrow \left(g^{\theta_{\varphi,n}} {g_1}^{\bar{\theta}_{\varphi,n}}\right)^{\beta_{\varphi,n}}, & b'_{\varphi,n} \leftarrow \left(g^{\theta'_{\varphi,n}} {g_1}^{\bar{\theta}'_{\varphi,n}}\right)^{\beta_{\varphi,n}}, \end{bmatrix}_{\varphi=(d,l)\in[D]\times[L],n\in[2]}$$
Note that this posits that $\omega = z_1 z_2$; in addition, both ω and $\widetilde{\omega}$ are both unknown to the simulator.

3. Compute what it can of the master key.
$$\begin{bmatrix} \mathsf{a}_{\varphi,n} \leftarrow g^{\alpha_{\varphi,n}}, & \mathsf{b}_{\varphi,n} \leftarrow g^{\beta_{\varphi,n}}, \\ y_{\varphi,n} \leftarrow \left(g^{\theta_{\varphi,n}} {g_1}^{\bar{\theta}_{\varphi,n}}\right)^{\alpha_{\varphi,n}\beta_{\varphi,n}}, & y'_{\varphi,n} \leftarrow \left(g^{\theta'_{\varphi,n}} {g_1}^{\bar{\theta}'_{\varphi,n}}\right)^{\alpha_{\varphi,n}\beta_{\varphi,n}} \end{bmatrix}_{\varphi=(d,l)\in[D]\times[L],n\in[2]}$$
Portion $\widetilde{\omega}$ 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 from $\mathbb{Z}_p^{D+2|\Lambda^\cup(\mathbf{B})|}$:

$$\left[\mu_d\right]_{d\in[D]},\ \left[\widetilde{\lambda}_{ID,n_0}\right]_{ID\in\Lambda_{d_0}(\mathbf{B})},\ \left[\lambda_{ID,n}\right]_{ID\in\Lambda_{d_0}(\mathbf{B}),n\neq n_0},\ \left[\lambda_{ID,n}\right]_{ID\in\Lambda^\cup(\mathbf{B})-\Lambda_{d_0}(\mathbf{B}),n\in[2]}$$

   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 $\mathbf{DK}(ID) = \left(\mathsf{k}_{ID,0}, \left[\mathsf{k}^{(\mathsf{a})}_{ID,1}, \mathsf{k}^{(\mathsf{b})}_{ID,1}\right], \left[\mathsf{k}^{(\mathsf{a})}_{ID,2}, \mathsf{k}^{(\mathsf{b})}_{ID,2}\right]\right)$ 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:

$$\mathsf{k}_{ID,0} \leftarrow g^{\mu_d} \cdot \prod_{n\in[2]} \left(y_{\varphi,n}{}^{ID} y'_{\varphi,n}\right)^{\lambda_{ID,n}},$$
$$\left[\mathsf{k}^{(\mathsf{a})}_{ID,n} \leftarrow \mathsf{a}_{\varphi,n}{}^{-\lambda_{ID,n}}, \mathsf{k}^{(\mathsf{b})}_{ID,n} \leftarrow \mathsf{b}_{\varphi,n}{}^{-\lambda_{ID,n}}\right]_{n\in[2]}$$

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:

$$\mathsf{k}_{ID,0} \leftarrow \widetilde{\omega} g^{\mu_{d_0}} \cdot \prod_{n\in[2]} \left(y_{\varphi_0,n}{}^{ID} y'_{\varphi_0,n}\right)^{\lambda_{ID,n}},$$
$$\left[\mathsf{k}^{(\mathsf{a})}_{ID,n} \leftarrow \mathsf{a}_{\varphi_0,n}{}^{-\lambda_{ID,n}}, \mathsf{k}^{(\mathsf{b})}_{ID,n} \leftarrow \mathsf{b}_{\varphi_0,n}{}^{-\lambda_{ID,n}}\right]_{n\in[2]}$$

   where

$$\lambda_{ID,n_0} = \widetilde{\lambda}_{ID,n_0} - \frac{z_2}{\alpha_{\varphi_0,n_0}\beta_{\varphi_0,n_0}\bar{\Theta}_{\varphi_0,n_0}} \tag{2}$$
$$\bar{\Theta}_{\varphi_0,n_0} = \bar{\theta}_{\varphi_0,n_0} ID + \bar{\theta}'_{\varphi_0,n_0} \neq 0.$$

   This ensures that λ_{ID,n_0} is distributed uniformly at random in $\mathbb{Z}_p$. And since $\bar{\theta}_{\varphi_0,n_0}\mathcal{I}^*_{\varphi_0} + \bar{\theta}'_{\varphi_0,n_0} = 0$; 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 λ_{ID,n_0} (since it does not know z_2), it can compute $\mathsf{a}_{\varphi_0,n_0}{}^{-\lambda_{ID,n_0}}$ and $\mathsf{b}_{\varphi_0,n_0}{}^{-\lambda_{ID,n_0}}$ given g^{z_2}. Since the simulator does not know $\widetilde{\omega}$, we now explain how to compute $\mathsf{k}_{ID,0}$. The simulator rewrites the equation for $\mathsf{k}_{ID,0}$ as

$$\mathsf{k}_{ID,0} = \left[g^{\mu_{d_0}} \cdot \left(y_{\varphi_0,2}{}^{ID} y'_{\varphi_0,2}\right)^{\lambda_{ID,2}}\right] \cdot \widetilde{\omega} \cdot \left(y_{\varphi_0,1}{}^{ID} y'_{\varphi_0,1}\right)^{\lambda_{ID,1}}$$

   Let $\Psi = g^{\mu_{d_0}} \cdot \left(y_{\varphi_0,2}{}^{ID} y'_{\varphi_0,2}\right)^{\lambda_{ID,2}}$, then $\mathsf{k}_{ID,0} = \Psi \cdot \widetilde{\omega} \cdot \left(y_{\varphi_0,n_0}{}^{ID} y'_{\varphi_0,n_0}\right)^{\lambda_{ID,n_0}}$ 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 λ_{ID,n_0} (since it does not know z_2), it is capable of computing $\mathsf{k}_{ID,0}$ given g^{z_1} and g^{z_2}; since if we rewrite $\mathsf{k}_{ID,0}$ as below, we can see that the exponent only contains z_1 and z_2 to the first degree. For convenience, we omit the subscripts φ_0, n_0 and ID below by letting $\alpha = \alpha_{\varphi_0,n_0}$, $\beta = \beta_{\varphi_0,n_0}$, $\theta = \theta_{\varphi_0,n_0}$, $\theta' = \theta'_{\varphi_0,n_0}$, $\bar{\theta} = \bar{\theta}_{\varphi_0,n_0}$, $\bar{\theta}' = \bar{\theta}'_{\varphi_0,n_0}$, $y = y_{\varphi_0,n_0}$, $y' = y'_{\varphi_0,n_0}$, $\overline{\Theta} = \overline{\Theta}_{\varphi_0,n_0}$, $\lambda = \lambda_{ID,n_0}$, $\widetilde{\lambda} = \widetilde{\lambda}_{ID,n_0}$.

$$\begin{aligned}\mathsf{k}_{ID,0} =& \Psi \cdot g^{z_1 z_2} \cdot \left(y^{ID} y'\right)^{\lambda} = \Psi \cdot g^{z_1 z_2} \cdot \left(g^{\alpha\beta(\theta + z_1\bar{\theta})ID} g^{\alpha\beta(\theta' + z_1\bar{\theta}')}\right)^{\widetilde{\lambda} - z_2/(\alpha\beta\overline{\Theta})} \\ &= \Psi \cdot g^{z_1 z_2} \cdot g^{-z_1 z_2(\bar{\theta}\cdot ID + \bar{\theta}')/\overline{\Theta}} \cdot g^{\mathsf{f}(z_1,z_2,\alpha,\beta,\theta,\theta',\bar{\theta},\bar{\theta}',\widetilde{\lambda},\overline{\Theta},ID)} = \Psi \cdot g^{\mathsf{f}(z_1,z_2,\alpha,\beta,\theta,\theta',\bar{\theta},\bar{\theta}',\widetilde{\lambda},\overline{\Theta},ID)}\end{aligned}$$

where $\mathsf{f}(z_1, z_2, \alpha, \beta, \theta, \theta', \bar{\theta}, \bar{\theta}', \widetilde{\lambda}, \overline{\Theta}, ID)$ is a polynomial where variables z_1 and z_2 have maximum degree 1.

Challenge: The adversary gives the simulator two messages $\mathbf{Msg}_0$ and $\mathbf{Msg}_1$. The simulator picks a random bit b, and encrypts $\mathbf{Msg}_b$ under point $\mathbf{X}^*$ as below:

1. Pick random integers $[r_{\varphi,n}]_{\varphi=(d,l)\in[D]\times[L],n\in[2]} \in \mathbb{Z}_p^{2DL}$.
2. Compute and release the following as the ciphertext.

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

Note that this implies that $r = z_3$; and if $Z = \mathbf{e}(g,g)^{z_1 z_2 z_3}$, it is easy to verify that the ciphertext is well-formed, due to the fact that $\left[\bar{\theta}_{\varphi,n}\mathcal{I}^*_{\varphi} + \bar{\theta}'_{\varphi,n} = 0\right]_{\varphi=(d,l)\in[D]\times[L],n\in[2]}$. 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.2 Proof: Anonymity

In Definition 9 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 $\mathtt{Setup}(\mathbf{\Sigma}, \mathbb{L}_{\mathbf{\Delta}})$ algorithm. This new definition can be further translated 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 9 [15].

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.

$$
\begin{aligned}
\mathbb{W}_{real}: &\quad \text{The challenge ciphertext is } \left(c, c_0, [c^{(b)}_{(1,1),1}, c^{(a)}_{(1,1),1}], \ldots, [c^{(b)}_{(D,L),2}, c^{(a)}_{(D,L),2}]\right);\\
\mathbb{W}_{0}: &\quad \text{The challenge ciphertext is } \left(*, c_0, [c^{(b)}_{(1,1),1}, c^{(a)}_{(1,1),1}], \ldots, [c^{(b)}_{(D,L),2}, c^{(a)}_{(D,L),2}]\right);\\
\mathbb{W}_{1,1,1}: &\quad \text{The challenge ciphertext is } \left(*, c_0, [*, *], [c^{(b)}_{(1,1),2}, c^{(a)}_{(1,1),2}], \ldots, [c^{(b)}_{(D,L),2}, c^{(a)}_{(D,L),2}]\right);\\
\mathbb{W}_{1,1,2}: &\quad \text{The challenge ciphertext is } \left(*, c_0, [*, *], [*, *], [c^{(b)}_{(1,2),1}, c^{(a)}_{(1,2),1}], \ldots, [c^{(b)}_{(D,L),2}, c^{(a)}_{(D,L),2}]\right);\\
\ldots &\\
\mathbb{W}_{D,L,1}: &\quad \text{The challenge ciphertext is } \left(*, c_0, [*, *], [*, *], \ldots, [*, *], [c^{(b)}_{(D,L),2}, c^{(a)}_{(D,L),2}]\right);\\
\mathbb{W}_{D,L,2}: &\quad \text{The challenge ciphertext is } (*, c_0, [*, *], [*, *], \ldots, [*, *], [*, *]).
\end{aligned}
$$

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

We reduce the anonymity of our MRQED scheme to the hardness of the D-Linear problem. We rewrite the D-Linear problem as given $[g, g^{z_1}, g^{z_2}, Y, g^{z_2 z_4}, g^{z_3+z_4}] \in \mathbb{G}^6$, where z_1, z_2, z_3, z_4 are picked at random from $\mathbb{Z}_p$, 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 $g_1 = g^{z_1}$, $g_2 = g^{z_2}$, $g^{\times}_{24} = g^{z_2 z_4}$, $g^{+}_{34} = g^{z_3+z_4}$.

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

Lemma C.4. *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* $\mathbb{W}_{d_1,l_1,n_1}$ *and the preceding game with more than* $\epsilon + 1/p$ *probability.*

Proof of Lemma C.4: 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 $[c^{(b)}_{(d_1,l_1),n_1}, c^{(a)}_{(d_1,l_1),n_1}]$ 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 $\mathcal{I}^*_\varphi = \mathcal{I}_\varphi(\mathbf{X}^*)$.

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

1. Pick the following parameters at random from $\mathbb{Z}_p^{12DL-3}$:

$$\omega, \quad \left[\alpha_{\varphi,n}, \beta_{\varphi,n}, \bar{\theta}_{\varphi,n}, \bar{\theta}'_{\varphi,n}\right]_{\varphi=(d,l)\in[D]\times[L], n\in[2], (\varphi,n)\neq(\varphi_1,n_1)}, \quad \left[\theta_{\varphi,n}, \theta'_{\varphi,n}\right]_{\varphi=(d,l)\in[D]\times[L], n\in[2]}$$

subject to the constraint that

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

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

We require that the α's, βs, $\bar{\theta}$'s and $\bar{\theta}'$'s are forcibly non-zero. In addition, later in Equation (5), we will need that $\theta_{\varphi_1,n_1}\mathcal{I}^*_{\varphi_1} + \theta'_{\varphi_1,n_1} \neq 0$. Hence, the simulator simply aborts if it happens to pick θ such that $\theta_{\varphi_1,n_1}\mathcal{I}^*_{\varphi_1} + \theta'_{\varphi_1,n_1} = 0$. 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.4.

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

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

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:

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

Note that the simulator does not know y_{φ_1,n_1} and y'_{φ_1,n_1}.

The following lemma shows that even if we do not know the parameters z_1, z_2, y_{φ_1,n_1} or y'_{φ_1,n_1}, we can still compute certain terms efficiently.

Lemma C.5. *In step (d_1, l_1, n_1) of the hybrid argument, 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 $\lambda_1 \in \mathbb{Z}_p$. Then, even though the simulator does know y_{φ_1,n_1}, it can efficiently generate the following term, such that the its resulting distribution is the same as when λ_2 is picked uniformly at random.*

$$\left(y^{ID_1}_{\varphi_1,n_1} y'_{\varphi_1,n_1}\right)^{\lambda_1} \cdot \left(y^{ID_2}_{\varphi_2,n_2} y'_{\varphi_2,n_2}\right)^{\lambda_2} \tag{3}$$

Moreover, the following two terms can also be computed efficiently

$$\mathsf{a}^{-\lambda_2}_{\varphi_2,n_2}, \mathsf{b}^{-\lambda_2}_{\varphi_2,n_2}. \tag{4}$$

Proof. For simplicity, let $\alpha = \alpha_{\varphi_2,n_2}$, $\beta = \beta_{\varphi_2,n_2}$. For $i \in [2]$, we use simply θ_i to denote θ_{φ_i,n_i}, and θ'_i to denote θ'_{φ_i,n_1}. We use simply $\bar{\theta}_2$ to denote $\bar{\theta}_{\varphi_2,n_2}$, and $\bar{\theta}'_2$ to denote $\bar{\theta}'_{\varphi_2,n_2}$. Notice we do not define $\bar{\theta}_1$, since $\bar{\theta}_{\varphi_1,n_1}$ and $\bar{\theta}'_{\varphi_1,n_1}$ are not defined. Define for $i \in [2]$, $\Theta_i = \theta_i \cdot ID_i + \theta'_i$ and define $\overline{\Theta}_2 = \bar{\theta}_2 \cdot ID_2 + \bar{\theta}'_2$.

Recall that the simulator picked parameters such that $\bar{\theta}_2 \mathcal{I}^*_{\varphi_2} + \bar{\theta}'_2 = 0$. In addition, since $ID_2 \neq \mathcal{I}^*_{\varphi_2}$, and $\bar{\theta}_2 \neq 0$,

$$\overline{\Theta}_2 = \bar{\theta}_2 \cdot ID_2 + \bar{\theta}'_2 \neq 0$$

First, the simulator pick λ uniformly at random and define

$$\lambda_2 = \lambda - \frac{z_2 \lambda_1 \Theta_1}{\alpha\beta\overline{\Theta}_2}.$$

Observe that λ_2 is distributed uniformly, but we cannot compute λ_2 efficiently because we do not know z_2. However, since we know g^{z_2}, we can compute g^{λ_2} efficiently. Hence, it follows that we can compute the two terms in (4) efficiently in the following way.

$$\mathsf{a}_{\varphi_2,n_2}^{-\lambda_2} = (g^{\lambda_2})^{-\alpha}, \mathsf{b}_{\varphi_2,n_2}^{-\lambda_2} = (g^{\lambda_2})^{-\beta}.$$

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

$$\begin{aligned}&\left(y_{\varphi_1,n_1}^{ID_1} y'_{\varphi_1,n_1}\right)^{\lambda_1} \cdot \left(y_{\varphi_2,n_2}^{ID_2} y'_{\varphi_2,n_2}\right)^{\lambda_2} = g^{z_1 z_2 \lambda_1 \left(\theta_1 ID_1 + \theta'_1\right)} \cdot \left(g^{\alpha\beta(\theta_2 + z_1\bar{\theta}_2) ID_2} g^{\alpha\beta(\theta'_2 + z_1 \bar{\theta}'_2)}\right)^{\lambda_2} \\ =&g^{z_1 z_2 \lambda_1 \Theta_1 + \alpha\beta(\Theta_2 + z_1 \overline{\Theta}_2)(\lambda - z_2\lambda_1\Theta_1/\alpha\beta\overline{\Theta}_2)} = g^{\alpha\beta\Theta_2\lambda} \cdot (g^{z_1})^{\alpha\beta\overline{\Theta}_2\lambda} \cdot (g^{z_2})^{-\lambda_1\Theta_1\Theta_2/\overline{\Theta}_2},\end{aligned}$$

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

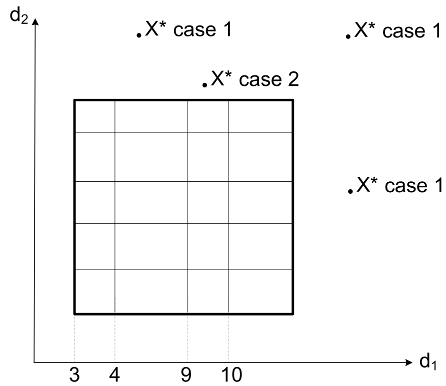
Phase 1: Suppose the adversary makes a decryption query for the hyper-rectangle $\mathbf{B}(s_1, t_1, \ldots, s_D, t_D)$. Since $\mathbf{B}$ does not contain $\mathbf{X}^*$, there exists a dimension $d_0 \in [D]$ such that $x^*_{d_0} \notin [s_{d_0}, t_{d_0}]$, where $x^*_{d_0}$ is $\mathbf{X}^*$ projected 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 $\Phi(ID) = \varphi_1$, $ID \neq \mathcal{I}_{\varphi_1}(\mathbf{X}^*)$.

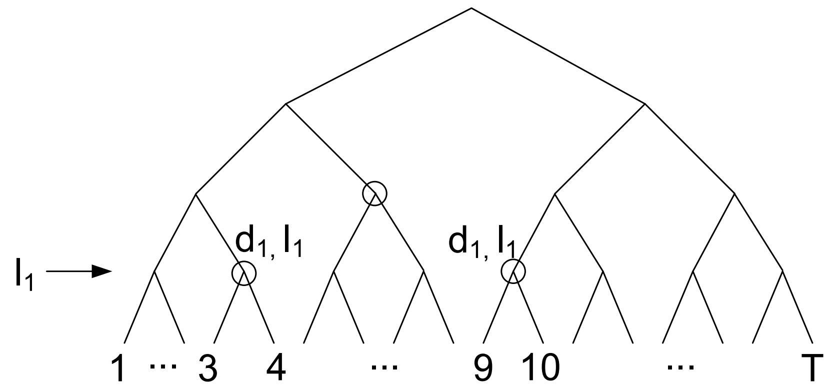
Case 2: There exists $ID \in \Lambda_{d_1}(\mathbf{B})$ such that $\Phi(ID) = \varphi_1$ and $ID = \mathcal{I}_{\varphi_1}(\mathbf{X}^*)$. Note that in this case, for all $ID' \in \Lambda_{d_1}(\mathbf{B})$ such that $ID' \neq ID$, $ID' \neq \mathcal{I}_{\varphi'}(\mathbf{X}^*)$, where $\varphi' = \Phi(ID')$; 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 $[\widetilde{\mu}_d]_{d\in[D]} \in_R \mathbb{G}^D$, such that $\prod_{d\in[D]} \widetilde{\mu}_d = \widetilde{\omega}$.



(a)



(b)

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.

(b) For each $ID \in \Lambda^{\cup}(\mathbf{B})$ where $\varphi := \Phi(ID) \neq \varphi_1$, pick at random $\lambda_{ID,1}, \lambda_{ID,2}$. Let $\mathbf{DK}(ID) = \left(\mathsf{k}_{ID,0}, [\mathsf{k}^{(\mathsf{a})}_{ID,1}, \mathsf{k}^{(\mathsf{b})}_{ID,1}], [\mathsf{k}^{(\mathsf{a})}_{ID,2}, \mathsf{k}^{(\mathsf{b})}_{ID,2}]\right)$ represent the element in $\mathbf{DK}$ for ID, compute and release $\mathbf{DK}(ID)$ as below:

$$\mathsf{k}_{ID,0} \leftarrow \widetilde{\mu}_d \cdot \prod_{n \in [2]} \left(y^{ID}_{\varphi,n} y'_{\varphi,n}\right)^{\lambda_{ID,n}},$$
$$\left[\mathsf{k}^{(\mathsf{a})}_{ID,n} \leftarrow \mathsf{a}^{-\lambda_{ID,n}}_{\varphi,n}, \mathsf{k}^{(\mathsf{b})}_{ID,n} \leftarrow \mathsf{b}^{-\lambda_{ID,n}}_{\varphi,n}\right]_{n \in [2]}$$

(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:

$$\mathsf{k}_{ID,0} \leftarrow \widetilde{\mu}_{d_1} \cdot \prod_{n \in [2]} \left(y^{ID}_{\varphi_1,n} y'_{\varphi_1,n}\right)^{\lambda_{ID,n}},$$
$$\left[\mathsf{k}^{(\mathsf{a})}_{ID,n} \leftarrow \mathsf{a}^{-\lambda_{ID,n}}_{\varphi_1,n}, \mathsf{k}^{(\mathsf{b})}_{ID,n} \leftarrow \mathsf{b}^{-\lambda_{ID,n}}_{\varphi_1,n}\right]_{n \in [2]}$$

Since the simulator does not know y_{φ_1,n_1} or y'_{φ_1,n_1}, it needs to use Lemma C.5 to generate $\mathbf{DK}(ID)$. Let $n' \neq n_1$. To apply Lemma C.5, the simulator first picks at random λ_{ID,n_1}, and rewrites $\mathsf{k}_{ID,0}$ as

$$\mathsf{k}_{ID,0} = \widetilde{\mu}_{d_1} \cdot \left(y^{ID}_{\varphi_1,n_1} y'_{\varphi_1,n_1}\right)^{\lambda_{ID,n_1}} \cdot \left(y^{ID}_{\varphi_1,n'} y'_{\varphi_1,n'}\right)^{\lambda_{ID,n'}}$$

Since $ID \neq \mathcal{I}_{\varphi_1}(\mathbf{X}^*)$, the simulator can apply Lemma C.5 by substituting (d_2, l_2, n_2) in the lemma with (d_1, l_1, n'), and λ_1 with λ_{ID,n_1}; in addition, both ID_1 and ID_2 in the lemma are substituted with ID.

Case 2: (a) Pick at random $[\mu_d]_{d \in [D]} \in_R \mathbb{Z}_p$ 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 $\lambda_{ID,1}, \lambda_{ID,2}$. Let $\mathbf{DK}(ID) = \left(\mathsf{k}_{ID,0}, [\mathsf{k}^{(\mathsf{a})}_{ID,1}, \mathsf{k}^{(\mathsf{b})}_{ID,1}], [\mathsf{k}^{(\mathsf{a})}_{ID,2}, \mathsf{k}^{(\mathsf{b})}_{ID,2}]\right)$ represent the element in $\mathbf{DK}$ for ID, compute and release $\mathbf{DK}(ID)$ as below:

$$\mathsf{k}_{ID,0} \leftarrow g^{\mu_d} \cdot \prod_{n\in[2]} \left(y^{ID}_{\varphi,n} y'_{\varphi,n}\right)^{\lambda_{ID,n}}$$
$$\left[\mathsf{k}^{(\mathsf{a})}_{ID,n} \leftarrow \mathsf{a}^{-\lambda_{ID,n}}_{\varphi,n}, \mathsf{k}^{(\mathsf{b})}_{ID,n} \leftarrow \mathsf{b}^{-\lambda_{ID,n}}_{\varphi,n}\right]_{n\in[2]}$$

(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 $\lambda_{\overline{ID},n_1} \in_R \mathbb{Z}_p$. Define $\Upsilon = \left(y^{\overline{ID}}_{\varphi_1,n_1} y'_{\varphi_1,n_1}\right)^{\lambda_{\overline{ID},n_1}}$.

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

$$\mathsf{k}_{ID,0} \leftarrow g^{\mu_{d_0}} \cdot \Upsilon \cdot \prod_{n\in[2]} \left(y^{ID}_{\varphi_0,n} y'_{\varphi_0,n}\right)^{\lambda_{ID,n}}$$
$$\left[\mathsf{k}^{(\mathsf{a})}_{ID,n} \leftarrow \mathsf{a}^{-\lambda_{ID,n}}_{\varphi_0,n}, \mathsf{k}^{(\mathsf{b})}_{ID,n} \leftarrow \mathsf{b}^{-\lambda_{ID,n}}_{\varphi_0,n}\right]_{n\in[2]}$$

This implies that $\widetilde{\mu}_{d_0} = g^{\mu_{d_0}} \cdot \Upsilon$. Note that Υ cannot be computed efficiently, as the simulator does not know y_{φ_1,n_1} or y'_{φ_1,n_1}. However, since $ID \neq \mathcal{I}_{\varphi_0}(\mathbf{X}^*)$, the simulator can apply Lemma C.5 by substituting (d_2, l_2, n_2) in the lemma with $(d_0, l, 1)$, λ_1 with $\lambda_{\overline{ID},n_1}$, ID_1 with $\overline{ID}$, and ID_2 with ID. The remaining terms in $\mathsf{k}_{ID,0}$ 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 $\mathbf{DK}(ID)$:

$$\mathsf{k}_{ID,0} \leftarrow g^{\mu_{d_1}} \cdot \Upsilon^{-1} \cdot \prod_{n\in[2]} \left(y^{ID}_{\varphi'_1,n} y'_{\varphi'_1,n}\right)^{\lambda_{ID,n}}$$
$$\left[\mathsf{k}^{(\mathsf{a})}_{ID,n} \leftarrow \mathsf{a}^{-\lambda_{ID,n}}_{\varphi'_1,n}, \mathsf{k}^{(\mathsf{b})}_{ID,n} \leftarrow \mathsf{b}^{-\lambda_{ID,n}}_{\varphi'_1,n}\right]_{n\in[2]}$$

This implies that $\widetilde{\mu}_{d_1} = g^{\mu_{d_1}} \cdot \Upsilon^{-1}$. Note that Υ^{-1} cannot be computed efficiently, as the simulator does not know y_{φ_1,n_1} or y'_{φ_1,n_1}. However, since $ID \neq \mathcal{I}_{\varphi'_1}(\mathbf{X}^*)$, the simulator can apply Lemma C.5, by substituting (d_2, l_2, n_2) in the lemma with $(d_1, l, 1)$, λ_1 with $-\lambda_{\overline{ID},n_1}$, ID_1 with $\overline{ID}$, and ID_2 with ID. The remaining terms in $\mathsf{k}_{ID,0}$ can be computed efficiently.

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

$$\mathsf{k}_{\overline{ID},0} \leftarrow g^{\mu_{d_1}} \cdot \Upsilon^{-1} \cdot \prod_{n\in[2]} \left(y^{\overline{ID}}_{\varphi_1,n} y'_{\varphi_1,n}\right)^{\lambda_{\overline{ID},n}},$$
$$\left[\mathsf{k}^{(\mathsf{a})}_{\overline{ID},n} \leftarrow \mathsf{a}^{-\lambda_{\overline{ID},n}}_{\varphi_1,n}, \mathsf{k}^{(\mathsf{b})}_{\overline{ID},n} \leftarrow \mathsf{b}^{-\lambda_{\overline{ID},n}}_{\varphi_1,n}\right]_{n\in[2]}$$

As before, here $\widetilde{\mu}_{d_1} = g^{\mu_{d_1}} \cdot \Upsilon^{-1}$. $\mathsf{k}_{\overline{ID},0}$ can be computed because the terms containing y_{φ_1,n_1} and y'_{φ_1,n_1} cancel out, leaving $\mathsf{k}_{\overline{ID},0} = g^{\mu_{d_1}} \cdot \left(y^{\overline{ID}}_{\varphi_1,n'} y'_{\varphi_1,n'}\right)^{\lambda_{\overline{ID},n'}}$.

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

$$\mathsf{k}_{ID,0} \leftarrow g^{\mu_{d_1}} \cdot \Upsilon^{-1} \cdot \prod_{n\in[2]} \left(y^{ID}_{\varphi_1,n} y'_{\varphi_1,n}\right)^{\lambda_{ID,n}},$$
$$\left[\mathsf{k}^{(\mathsf{a})}_{ID,n} \leftarrow \mathsf{a}^{-\lambda_{ID,n}}_{\varphi_1,n}, \mathsf{k}^{(\mathsf{b})}_{ID,n} \leftarrow \mathsf{b}^{-\lambda_{ID,n}}_{\varphi_1,n}\right]_{n\in[2]}$$

Again, to be able to generate $\mathsf{k}_{ID,0}$, Lemma C.5 is required. However, in this case, a slight complication is involved, since two terms in $\mathsf{k}_{ID,0}$ contain y_{φ_1,n_1} and y'_{φ_1,n_1}:

$$\begin{aligned}
\mathsf{k}^{(\mathrm{O})}_{ID} &= g^{\mu_{d_1}} \cdot \Upsilon^{-1} \cdot \prod_{n\in[2]} \left(y^{ID}_{\varphi_1,n} y'_{\varphi_1,n}\right)^{\lambda_{ID,n}} \\
&= g^{\mu_{d_1}} \cdot \left(y^{\overline{ID}}_{\varphi_1,n_1} y'_{\varphi_1,n_1}\right)^{-\lambda_{\overline{ID},n_1}} \cdot \prod_{n\in[2]} \left(y^{ID}_{\varphi_1,n} y'_{\varphi_1,n}\right)^{\lambda_{ID,n}} \\
&= g^{\mu_{d_1}} \cdot \left(\left(y^{\overline{ID}}_{\varphi_1,n_1} y'_{\varphi_1,n_1}\right)^{-\lambda_{\overline{ID},n_1}} \cdot \left(y^{ID}_{\varphi_1,n_1} y'_{\varphi_1,n_1}\right)^{\lambda_{ID,n_1}}\right) \cdot \left(y^{ID}_{\varphi_1,n'} y'_{\varphi_1,n'}\right)^{\lambda_{ID,n'}}
\end{aligned}$$

Now the simulator picks λ_{ID,n_1} at random from $\mathbb{Z}_p^*$, and computes

$$\widetilde{\lambda}_{ID,n_1} = \lambda_{ID,n_1} \frac{\theta_{\varphi_1,n_1} \cdot ID + \theta'_{\varphi_1,n_1}}{\theta_{\varphi_1,n_1} \cdot \overline{ID} + \theta'_{\varphi_1,n_1}} - \lambda^{(\overline{ID})}_{n_1} \tag{5}$$

Here we require that $\theta_{\varphi_1,n_1} \cdot \overline{ID} + \theta'_{\varphi_1,n_1} \neq 0$. Notice that $\overline{ID} = \mathcal{I}^*_{\varphi_1}$. As we explained in the **Setup** stage, the simulator aborts if it happens to pick $\theta_{\varphi_1,(n_1,j)}$'s such that $\theta_{\varphi_1,n_1}\mathcal{I}^*_{\varphi_1} + \theta'_{\varphi_1,n_1} = 0$. Hence,

$$\mathsf{k}_{ID,0} = g^{\mu_{d_1}} \cdot \left(y^{\overline{ID}}_{\varphi_1,n_1} y'_{\varphi_1,n_1}\right)^{\widetilde{\lambda}_{ID,n_1}} \cdot \left(y^{ID}_{\varphi_1,n'} y'_{\varphi_1,n'}\right)^{\lambda_{ID,n'}}$$

And now the simulator can apply Lemma C.5 by substituting (d_2, l_2, n_2) in the lemma with (d_1, l_1, n'), λ_1 with $\widetilde{\lambda}_{ID,n_1}$, ID_1 with $\overline{ID}$, and ID_2 with ID.

Challenge: On receiving a message $\mathbf{Msg}$ from the adversary, the simulator does the following:

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

2. Compute and release the following as the ciphertext.

$$*, g^{+}_{34}, [*,*], \ldots, [*,*], \quad (g^{\times}_{24})^{\theta_{\varphi_1,n_1}\mathcal{I}^*_{\varphi_1}+\theta'_{\varphi_1,n_1}}, \quad Y^{\theta_{\varphi_1,n_1}\mathcal{I}^*_{\varphi_1}+\theta'_{\varphi_1,n_1}},$$
$$\left[g^{r_{\varphi,n}\beta_{\varphi,n}\left(\theta_{\varphi,n}\mathcal{I}^*_{\varphi}+\theta'_{\varphi,n}\right)}, \quad \left(g^{+}_{34} \cdot g^{-r_{\varphi,n}}\right)^{\alpha_{\varphi,n}\left(\theta_{\varphi,n}\mathcal{I}^*_{\varphi}+\theta'_{\varphi,n}\right)}\right]_{(d_1,l_1,n_1)<(d,l,n)<(D,L,2),\varphi=(d,l)}$$

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

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

If Y is a random number, then term $c^{(a)}_{(d_1,l_1),n_1}$ 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 $Y = g^{z_1 z_3}$. 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 $\mathbb{G}$. ■

Proof of Theorem C.3: The theorem follows naturally from Lemma C.4 and the hybrid argument. ■

D Adaptive-ID Security

Definition 12 (MRQED adaptive security)**.** *An MRQED scheme is* ***adaptively secure*** *if all polynomial-time adversaries have at most a negligible advantage in the adaptive security game defined below:*

- **Setup**: The challenger runs the $\mathbf{Setup}(\mathbf{\Sigma}, \mathbb{L}_{\mathbf{\Delta}})$ algorithm to generate $\mathbf{PK}, \mathbf{SK}$. It gives $\mathbf{PK}$ to the adversary, but does not divulge $\mathbf{SK}$.
- **Phase 1**: The adversary adaptively issues decryption key queries for hyper-rectangles $\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_{q_0}$.
- **Challenge**: The adversary submits two pairs $(\mathbf{X}^*_0, \mathbf{Msg}_0), (\mathbf{X}^*_1, \mathbf{Msg}_1)$, where $\mathbf{X}^*_0, \mathbf{X}^*_1 \in \mathbb{L}_\Delta$, and $\mathbf{Msg}_0, \mathbf{Msg}_1 \in \mathbb{M}$ are two equal length messages. Furthermore, $\mathbf{X}^*_0$ and $\mathbf{X}^*_1$ are not contained in any hyper-rectangles queried in **Phase 1**, i.e., for $0 < i \le q_0$, $\mathbf{X}^*_0 \notin \mathbf{B}_i$, and $\mathbf{X}^*_1 \notin \mathbf{B}_i$. Now the challenger flips a random coin, b, and encrypts $\mathbf{Msg}_b$ under $\mathbf{X}^*_b$. The ciphertext is passed to the adversary.
- **Phase 2**: Phase 1 is repeated. The adversary adaptively issues decryption key queries for hyper-rectangles $\mathbf{B}_{q_0+1}, \mathbf{B}_{q_0+2}, \ldots, \mathbf{B}_q$. As before, all hyper-rectangles queried in this stage must not contain $\mathbf{X}^*_0$ and $\mathbf{X}^*_1$ specified in the previous stage.
- **Guess**: The adversary outputs a guess b' of b.

We define the adversary $\mathcal{A}$'s advantage in the above game as

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

The difference between the two notions of security is that in selective security, the adversary commits to two points $\mathbf{X}^*_0$ and $\mathbf{X}^*_1$ at the beginning of the security game. Therefore, selective security is weaker than adaptive security. In Appendix C, we prove the selective security of our construction under the Decisional BDH and the Decisional Linear Assumption in bilinear groups of prime order.

Previous research has shown that IBE schemes secure in the selective-ID sense can be converted to schemes fully secure [6, 18, 45, 36] with some loss in security. In particular, Boneh and Boyen prove the following theorem:

Theorem D.1 ([6])**.** *A (t, q, ϵ)-selective identity secure IBE system (*$\mathsf{IND\text{-}sID\text{-}CPA}$*) that admits N distinct identities is also a $(t, q, N\epsilon)$-fully secure IBE (*$\mathsf{IND\text{-}ID\text{-}CPA}$*).*

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.

E Trivial MRQED Construction

We first describe the trivial construction for one dimension. Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ denote a secure public key encryption scheme. $\mathcal{K}, \mathcal{E}, \mathcal{D}$ represent the key generation, encryption and decryption algorithm respectively. We build a MRQED^1 based on $\mathcal{AE}$ as below.

- During $\mathbf{Setup}$, one runs $\mathcal{K}$, the key generation algorithm, $O(T^2)$ times to generate the following public and private keys:

$$\mathbf{PK} = \left\{\mathsf{pk}_{s,t}\middle|1 \leq s \leq t \leq T\right\}, \quad \mathbf{SK} = \left\{\mathsf{sk}_{s,t}\middle|1 \leq s \leq t \leq T\right\}$$

- To encrypt a pair $(\mathbf{Msg}, x)$ where x is a point between 1 and T, first define for $1 \leq s \leq t \leq T$

$$\delta_{s,t}(\mathbf{Msg}, x) = \begin{cases} \mathbf{Msg} & \text{if } s \leq x \leq t \\ \perp & \text{otherwise} \end{cases}$$

  where $\perp$ denotes the "invalid message". Now one runs the encryption algorithm $\mathcal{E}$, and for all ranges $[s, t] \subseteq [1, T]$, one encrypts $\delta_{s,t}(\mathbf{Msg}, x)$ under $\mathsf{pk}_{s,t}$. The result of encryption is a tuple of length T^2, denoted $(c_{1,1}, c_{1,2}, \ldots, c_{T,T})$.

- To release a decryption key $\mathbf{DK}_{s,t}$ for range $[s, t] \subseteq [1, T]$, one releases the key $\mathsf{sk}_{s,t}$.

- To decrypt a ciphertext $\mathbf{C} = (c_{1,1}, c_{1,2}, \ldots, c_{T,T})$ with $\mathbf{DK}_{s,t}$, one uses $\mathbf{DK}_{s,t}$ to decrypt $c_{s,t}$. Decryption either yields $\perp$, if the point x encrypted does not fall within the range $[s, t]$; or it yields the message $\mathbf{Msg}$, if x falls within $[s, t]$.

Clearly, the trivial MRQED^1 construction results in $O(T^2)$ public key size, $O(T^2)$ encryption overhead and ciphertext size, $O(1)$ decryption key size and $O(1)$ decryption cost.

One can easily extend the trivial construction into multiple dimensions. The resulting MRQED^D scheme requires that one encrypt $\delta_{\mathbf{B}}(\mathbf{Msg}, \mathbf{X})$ for all hyper-rectangles $\mathbf{B}$ in space. Therefore, the trivial MRQED^D scheme has $O(T^{2D})$ public key size, $O(T^{2D})$ encryption cost and ciphertext size, $O(1)$ decryption key size and $O(1)$ decryption cost.

F AIBE-Based MRQED^D Construction

In Section 4.2, we described an AIBE-based MRQED^1 scheme. The same idea can be applied to construct an MRQED^D scheme by making the following analogy between MRQED^1 and MRQED^D.

In MRQED^1, if a range $[s, t]$ can be represented by a single node in $\mathsf{tr}(T)$, then we say that $[s, t]$ is a *simple range*. In other words, a range $[s, t] \subseteq [1, T]$ is a simple range *iff* $|\Lambda(s, t)| = 1$.

Recall that in MRQED^1, $\mathsf{tr}(T)$ is the set of all nodes in the tree. We can also regard $\mathsf{tr}(T)$ as the collection of all simple ranges: for $ID \in \mathsf{tr}(T)$, $\mathsf{cv}(ID)$ is a simple range. Therefore, we can think of an ID as denoting a simple range in $[1, T]$. In MRQED^1, when we encrypt under a point x, we encrypt under all IDs in $\mathsf{tr}(T)$ covering x. In other words, we encrypt under every simple range containing point x. When we issue decryption keys for range $[s, t]$, we issue a key for every $ID \in \Lambda(s, t)$, that is to say, we denote $[s, t]$ as the collection of simple ranges, and issue a key for every simple range in that collection.

The analog of a simple range in multiple dimensions is a *simple hyper-rectangle*. To define a simple hyper-rectangle, we first build D interval trees, one corresponding to each dimension. We assign a globally unique ID to each tree node. Simply put, a simple hyper-rectangle is a hyper-rectangle $\mathbf{B}_0$ in space, such that $\mathbf{B}_0$ can be represented by a single node in the tree of every dimension. More specifically, a hyper-rectangle $\mathbf{B}(\mathbf{s_1}, \mathbf{t_1}, \ldots, \mathbf{s_D}, \mathbf{t_D})$ in space is composed of a range along each dimension. If for all $1 \leq i \leq D$, $|\Lambda(s_i, t_i)| = 1$, i.e., $[s_i, t_i]$ is a simple range in the i^{th} dimension, then we say that the hyper-rectangle $\mathbf{B}(s_1, t_1, \ldots, s_D, t_D)$ is a *simple hyper-rectangle*. A simple hyper-rectangle can be defined by a single node from each dimension. We can assign a unique identity to each simple-rectangle $\mathbf{B}_0(s_1, t_1, \ldots, s_D, t_D)$ in space. Define

$$\mathsf{id}_{\mathbf{B}_0} = (ID_1, ID_2, \ldots, ID_D),$$

where $ID_i (1 \leq i \leq D)$ is the node representing $[s_i, t_i]$ in the i^{th} dimension.

Analogous to $\mathsf{tr}(T)$, we now define $\mathbb{U}$ to be the set $\left\{\mathsf{id}_{\mathbf{B}_0} \middle| \mathbf{B}_0 \text{ is a simple hyper-rectangle}\right\}$. $\mathbb{U}$ can be thought of as the set of all simple hyper-rectangles in space. Analogous to $\mathsf{cv}(ID)$ defined for one dimension, define $\mathsf{cv}(\mathsf{id}_{\mathbf{B}_0}) = \mathbf{B}_0$. Below we define $\mathbb{P}^\times$ and $\Lambda^\times$ analogous to $\mathbb{P}$ and Λ in one dimension:

- **Set of id's covering a point $\mathbf{X}$.** For a point $\mathbf{X}$ in space, $\mathbb{P}^\times(\mathbf{X})$ is the set of all id's in $\mathbb{U}$ such that $\mathbf{X} \in \mathsf{cv}(\mathsf{id})$; hence, in this case, $\mathbb{P}^\times(\mathbf{X})$ is the set of all simple hyper-rectangles that contain the point $\mathbf{X}$. Let $\mathbf{X} = (x_1, x_2, \ldots, x_D)$. It is not hard to see that the cross-product $\mathbb{P}_1(\mathbf{X}) \times \mathbb{P}_2(\mathbf{X}) \times \ldots \times \mathbb{P}_D(\mathbf{X})$ define all simple hyper-rectangles containing $\mathbf{X}$. Therefore, $\mathbb{P}(\mathbf{X}) = \mathbb{P}_1(\mathbf{X}) \times \mathbb{P}_2(\mathbf{X}) \times \ldots \times \mathbb{P}_D(\mathbf{X})$; and for any $\mathbf{X} \in \mathbb{L}_\Delta$, $|\mathbb{P}(\mathbf{X})| = O\left((\log T)^D\right)$.

- **Hyper-rectangle as a collection of id's.** For any hyper-rectangle $\mathbf{B}(s_1, t_1, \ldots, s_D, t_D)$ in space, $\Lambda^\times(\mathbf{B})$ is the the minimal subset of $\mathbb{U}$ such that $\bigcup_{\mathsf{id} \in \Lambda^\times(\mathbf{B})} \mathsf{cv}(\mathsf{id}) = \mathbf{B}$. In other words, $\Lambda^\times(\mathbf{B})$ is the minimal set of simple hyper-rectangles that jointly cover the hyper-rectangle $\mathbf{B}$. For convenience, denote $\Lambda_d(\mathbf{B}) := \Lambda(s_d, t_d)$, $\Lambda_d(\mathbf{B})$ is the minimal set of nodes covering range $[s_d, t_d]$ in the d^{th} dimension. It is not hard to see that

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

In particular, for every $\mathsf{id} = (ID_1, ID_2, \ldots, ID_D) \in \Lambda^\times(\mathbf{B})$, where $ID_i (1 \leq i \leq D)$ is a node in the tree corresponding to the i^{th} dimension, id defines a simple hyper-rectangle $\mathbf{B}_0$; and $\Lambda^\times(\mathbf{B}_0) = \{(ID_1, ID_2, \ldots, ID_D)\}$. It is not hard to check that for any hyper-rectangle $\mathbf{B} \subseteq \mathbb{L}_\Delta$, $|\Lambda^\times(\mathbf{B})| = O\left((\log T)^D\right)$.

The above definitions satisfy the following properties: For a point $\mathbf{X}$ and a hyper-rectangle $\mathbf{B}$, if $\mathbf{X} \in \mathbf{B}$, then $\mathbb{P}^\times(\mathbf{X}) \cap \Lambda^\times(\mathbf{B}) \neq \emptyset$; in addition, they intersect at only one simple hyper-rectangle. Otherwise, if $\mathbf{X} \notin \mathbf{B}$, then $\mathbb{P}^\times(\mathbf{X}) \cap \Lambda^\times(\mathbf{B}) = \emptyset$.

Now we may apply the same AIBE-based construction of Section 4.2 to the multi-dimension case. The resulting scheme encrypts a message $\mathbf{Msg}$ under every simple hyper-rectangle in spaces that contain the point $\mathbf{X} = (x_1, x_2, \ldots, x_D)$. This is equivalent to encrypting $\mathbf{Msg}$ under every $\mathsf{id} \in \mathbb{P}^\times(\mathbf{X})$, where $\mathbb{P}^\times(\mathbf{X})$ is the cross-product of all paths from a leaf node representing x_i, to the root of the tree in the i^{th} dimension: $\mathbb{P}^\times(\mathbf{X}) = \mathbb{P}_1(\mathbf{X}) \times \mathbb{P}_2(\mathbf{X}) \times \ldots \times \mathbb{P}_D(\mathbf{X})$. To release the decryption key for hyper-rectangle $\mathbf{B}$, it releases the key for a set of simple hyper-rectangles that compose $\mathbf{B}$. In other words, we release the key for every $\mathsf{id} \in \Lambda^\times(\mathbf{B})$. Since both $\mathbb{P}^\times(\mathbf{X})$ and $\Lambda^\times(\mathbf{B})$ have size $O\left((\log T)^D\right)$, the AIBE-based MRQED^D scheme has $O\left((\log T)^D\right)$ encryption cost, ciphertext size, and decryption key size.

Now we examine the cost of decryption. Suppose $\mathbf{DK_B}$ is our decryption key for hyper-rectangle $\mathbf{B}$, and $\mathbf{DK_B}$ is composed of a key $k_{\mathbf{B}_0}$ for every simple hyper-rectangle in $\Lambda^\times(\mathbf{B})$. Let $\mathbf{C}$ denote a ciphertext under point $\mathbf{X}$. $\mathbf{C}$ consists of a component $c_{\mathbf{B}_0}$ for every simple hyper-rectangle containing the point $\mathbf{X}$. Now if we naively try every $k_{\mathbf{B}_0}$ over every $c_{\mathbf{B}'_0}$, then decryption cost would be $O\left(|\mathbb{P}^\times(\mathbf{X})| \cdot |\Lambda^\times(\mathbf{B})|\right)$, and in this case, $O\left((\log T)^{2D}\right)$. However, just as in the one dimensional case, we know that ID and ID' cannot be equal if they are at different depths in the tree. Define $\ell(ID)$ to extract the depth of node ID in its tree. For $\mathsf{id} = (ID_1, ID_2, \ldots, ID_D)$, where ID_i is a node in the tree corresponding to the i^{th} dimension, define $\ell(\mathsf{id}) = (\ell(ID_1), \ell(ID_2), \ldots, \ell(ID_D))$. Therefore, we only need to try $k_{\mathbf{B}_0}$ over $c_{\mathbf{B}'_0}$ when $\ell(\mathsf{id}_{\mathbf{B}_0}) = \ell(\mathsf{id}_{\mathbf{B}'_0})$. Of course, we cannot directly release $\mathbf{B}'_0$ for the ciphertext $c_{\mathbf{B}'_0}$, since the ciphertext is meant to hide the point $\mathbf{X}$ being encrypted. However, observe that since we are encrypting a message under all $\mathsf{id} \in \mathbb{P}_1(\mathbf{X}) \times \mathbb{P}_2(\mathbf{X}) \times \ldots \times \mathbb{P}_D(\mathbf{X})$, $\ell(\mathsf{id})$ naturally establishes a bijection between $c_{\mathsf{id}} \in \mathbf{C}$ and $(l_1, l_2, \ldots, l_D) \in [L]^D$. Therefore, we can release $\ell(\mathsf{id})$ along with every ciphertext c_{id} without leaking any information about $\mathbf{X}$. Now since we only need to try $k_{\mathbf{B}_0}$ over $c_{\mathbf{B}'_0}$ when $\ell(\mathsf{id}_{\mathbf{B}_0}) = \ell(\mathsf{id}_{\mathbf{B}'_0})$, the decryption cost is reduced to $O\left((\log T)^D\right)$ instead of $O\left((\log T)^{2D}\right)$.