Delegatable Order-Revealing Encryption

Yuan Li
School of Computer Science, Fudan University
Shanghai, China
15110240022@fudan.edu.cn

Hongbing Wang
School of Computer Science, Fudan University
Shanghai, China
wanghongbing@fudan.edu.cn

Yunlei Zhao*
School of Computer Science, Fudan University
Shanghai, China
ylzhao@fudan.edu.cn

ABSTRACT

Order-revealing encryption (ORE) is a basic cryptographic primitive for ciphertext comparisons based on the order relationship of plaintexts while maintaining the privacy of them. In the data era we are experiencing, cross-dataset transactions become ubiquitous in practice. However, almost all the previous ORE schemes can only support comparisons on ciphertexts from the *same* user, which does not meet the requirement for the multi-user environment.

In this work, we introduce and design ORE schemes with delegation functionality, which is referred to as delegatable ORE (DORE). The "delegation" here is an authorization that allows for efficient ciphertext comparisons among different users. To the best of our knowledge, it is the first ORE that allows an user to delegate the comparison privilege for his ciphertexts, which also opens the door for future explorations. At the heart of the construction and analysis of DORE is a new building tool proposed in this work, named *delegatable equality-revealing encoding* (DERE), which might be of independent interest.

CCS CONCEPTS

• **Security and privacy → Management and querying of encrypted data**.

KEYWORDS

order-revealing encryption, encrypted database, delegation

ACM Reference Format:
Yuan Li, Hongbing Wang, and Yunlei Zhao. 2019. Delegatable Order-Revealing Encryption. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS '19), July 9–12, 2019, Auckland, New Zealand.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3321705.3329829

1 INTRODUCTION

The rapid development of cloud technology provides us a convenient way to fulfil various storage and computational tasks in cloud. Therefore, from the economic perspective, it is a better choice to put the database (DB) server's workload on the cloud, and deal with the database transactions with the aid of cloud. However, directly deploying the tasks on the cloud does not meet the privacy requirement in some specific scenarios, such as military, commerce and medical treatment scenes where data are usually regarded as sensitive information. In order to avoid the exposure of sensitive data, a feasible and practical solution is to encrypt the sensitive data into ciphertexts, only on which the server accomplishes the database transactions[22, 23, 37], e.g., CryptDB [26].

In an encrypted database (EDB) system, SQL queries related to keyword search could be handled by the symmetric searchable encryption [9, 15, 16, 20, 32–34] and public key encryption with keyword search [7] without loss of data confidentiality. The SQL queries related to addition and equality test could be handled by homomorphic encryption [24] and deterministic encryption [2], respectively. To implement the order comparison functionality while preserving the privacy of sensitive data, cryptographic primitives such as order-preserving encryption (OPE) and order-revealing encryption (ORE) have been introduced and developed.

An OPE scheme has the property that ciphertexts preserve the numerical order of the plaintexts. The notion of OPE was first introduced by Agrawal et al. [1] in 2005. Boldyreva et al. [4] defined the security model, and proposed an efficient OPE scheme by using a tool called hyper geometric distribution. Unfortunately, the OPE construction proposed in [4] was later shown to be insecure [5], as it reveals at least half of bit information about the plaintext. In 2013, Popa et al. [25] constructed the first OPE scheme with ideal security, specifically indistinguishability against order chosen-plaintext attack (IND-OCPA), by using the data structure of B-tree. However, the ideally secure OPE scheme presented in [25] requires significant communication overheads, which makes the scheme far from being practical. Later, its communication complexity is further reduced (in an average sense) by Kerschbaum and Schroepfer [19]. Recently, a partial order-preserving encoding (POPE) scheme was proposed by Roche et al. [28]. The POPE scheme makes a tradeoff between data insertion and order comparison; In particular, it has a highly efficient insertion algorithm, so that it is more applicable to systems where much more data insertions are needed to be executed than order comparisons. Xiao et al. [36] gave a methodology to extend OPE into multi-user systems, whereas their solution could only support the OPE instead of ORE. The negative result [25], that efficient and non-interactive

AsiaCCS '19, July 9–12, 2019, Auckland, New Zealand

ACM ISBN 978-1-4503-6752-3/19/07…$15.00
https://doi.org/10.1145/3321705.3329829

OPE does not exist, inherently limits the application of Xiao et al.'s protocol.

The ORE can be viewed as the generalization of OPE. In an ORE scheme, the order relationship of plaintexts could be obtained via evaluating certain functions on the corresponding ciphertexts. The motivation of ORE is to circumvent the negative conclusion that no efficient and non-interactive OPE can achieve the best-possible security, namely the ideal security. The concept of ORE was first proposed by Boneh et al. [8], and they also gave a concrete construction of ORE, which is ideally secure based on the powerful yet less mature cryptographic tool, i.e., multilinear maps. Due to the heavy computation burden of current multilinear maps, Chenette et al. [12] introduced an efficient ORE scheme at the expense of leaking the index of the first different digit bit with respect to the plaintexts. To better balance the efficiency and security, Lewi and Wu [21] proposed a generalization of the ORE scheme presented in [12], which leaks the index of the first different block of the plaintexts. With bilinear maps, Cash et al. [11] later constructed a more secure ORE scheme, which only leaks the "equality pattern" of the first different digit bit. Furthermore, Cash et al. [10] introduced a new notion called parameter-hiding ORE, and gave a construction based on the asymmetric bilinear map. The parameter-hiding ORE provides a strong security in the case that the database entries are drawn identically and independently from a distribution of known shape, but for which the mean and variance are not known. Recently, several works about the attacks to the ORE were studied, e.g., the multi-column attack [13], the augmented auxiliary-information-using inference attack [18], file-injection attack [35]. Recently, Eom et al. [14] designed a multi-client ORE which supports ciphertexts comparison among different clients. However, their cryptosystem relies on a trusted third party to manage all clients' secret keys, which may cause potential security problem and limits its application.

1.1 Motivation

In general, multiple-database queries and cross-database queries are commonly utilized to handle database transactions. Below, we consider the following two specific examples.

In modern commercial environment, the business cooperation between any two companies may have some data analysis requirements about the product price or sale, e.g., search the most expensive product, search the top N selling products.

In medical treatment environment, several institutions may need to jointly research about the HIV/AIDS. Such an task probably have the requirement of order-based comparisons (e.g., maximum, minimum, sort, range) on the relevant data stored in the databases of each institution.

In the scenarios described above, all the data in each entity's database are first encrypted, and then outsourced to the cloud server for convenient, low-cost and privacy-preserving services. Naturally, we need the cloud server to perform the order-based operations upon these encrypted data. Unfortunately, almost all the existing ORE schemes only support comparisons on the data of a single user, which obviously do not meet the requirement for the scenarios described above. This much reduces the applicability of ORE in practice.

As a first attempt to achieve ORE with the functionality of ciphertext comparisons on data of different users, throughout this work we assume that the entities who would like this functionality are trusted. We argue that the task remains highly non-trivial even in this case.

To provide order comparison among different entities, a natural way is to preset the users' private keys to be the same. However, this method has some inherent drawbacks, making it less flexible and applicable in reality. On the one hand, these entities need cooperations in advance in order to allow ciphertext comparisons among them. However, in most application scenarios, e.g., modern commercial environment, an entity can hardly know which entities to cooperate in advance. On the other hand, supposing an entity A would like ciphertext comparisons with B and C while B and C do not want comparisons between them, in this case different private keys are need for the pair of (A, B) and that of (A, C), making the ciphertext size of entity A linearly dependent upon on the number of cooperative entities.

Another naive approach is that, the entity A can privately send its private key to B, who later downloads A's encrypted databases from the cloud server and decrypts them to do the comparison; Or B re-encrypts his own databases by using A's private key, and lets the cloud do the comparison upon the ciphertexts. This method suffers from a heavy storage and bandwidth burden, loses the convenience and advantage brought by cloud, and is apparently not applicable to resource-constrained scenarios, e.g., wireless networks and mobile devices. Much worse, totally surrendering ones' private keys is highly undesirable for most applications.

An alternative and feasible approach is to let each entity to generate a token to delegate ciphertext comparison to a third party, e.g., the cloud. With this approach, the third party can compare the ciphertexts among entities who have submitted their tokens. This way, multiple databases encrypted with different keys by the same entity can also be compared. However, for all the existing practical ORE schemes, this approach at least requires an *interactive* token generation protocol (e.g., via secure multi-party computation in general) between each pair of entities, which significantly limits its flexibility and practicality. In addition, most existing ORE schemes, e.g., [11, 12, 21], have the encryption structure that each binary bit and its prefix are encoded via a pseudorandom function (PRF) with the entity's private key. For these ORE schemes, the interactively generated tokens should also establish relationship for the outputs of PRF using different private keys; To our knowledge, PRF with this feature is not at hand yet.

1.2 Contribution

In this work, we introduce and design ORE schemes with *non-interactive* delegation functionality, which is referred to

as delegatable ORE (DORE) for presentation simplicity. The "delegation" here is an authorization that allows for efficient ciphertext comparisons among different users. To the best of our knowledge, it is the first attempt to construct ORE schemes for efficiently comparing ciphertexts among different entities. At the heart of the construction and analysis of DORE is a new primitive proposed in this work, which is named *delegatable equality-revealing encoding* (DERE). A DERE scheme supports the equality test (by the delegatable authority) on ciphertexts from different entities.

We present the syntaxes, and formal security definitions for both DORE and DERE. We then present a generic, provably secure construction of DORE based on DERE, with respect to leakage $\mathcal{L}_{\mathsf{msdb}}$ (standing for the index of the most significant different bit) that was introduced and defined in [21]. Finally, a concrete DERE scheme is proposed in asymmetric bilinear groups, and its security is proved in the generic group and random oracle model by following the methodology in [3]. We end this work with some discussions on DORE applications and future research directions.

The proposed DORE scheme is feasible to be implemented in the cloud-based encrypted database (EDB) system. The time-consuming pairing-based comparison algorithm is run on the server-side, whereas the encryption algorithm on the client side only performs relatively efficient group operations. In addition, it allows for fine-grained access control. Table 1 gives a brief comparison between our DORE and some existing efficient ORE schemes. In this table, the parameter n is the maximum bit-length of messages in ORE/DORE. The symbols $\tau_{\mathbb{G}_1}$ and $\tau_{\mathbb{G}_2}$ are the sizes of elements in bilinear groups $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. We use e, p, mc, h and prf to denote the time costs for performing a group exponentiation in $\mathbb{G}_1$, pairing, modular comparison, hash and pseudorandom function, respectively. The symbols msdb and $\mathsf{eq\text{-}pattern}$ denote the index of the most significant different bit and the equality pattern of msdb as defined in [12], respectively.

2 PRELIMINARIES

For all $n \in \mathbb{N}$, we use $[n]$ to denote the integer set $\{1, \cdots, n\}$. A string or value α means a binary one, and $|\alpha|$ is its binary length. If α and β are two strings, $\alpha||\beta$ is their concatenation. For a binary encoding $m \in \{0,1\}^n$ and $i \in [n]$, we use $m[i]$ to denote the i-th bit of m. Thus, the binary encoding for $m \in \{0,1\}^n$ can be written as $m[1] \cdots m[n]$. By $\mathsf{pref}(m, i)$, we denote the string of the first i bits of m. If $\vec{t}$ is a vector, its length is denoted by $|\vec{t}|$. If $\mathbb{S}$ is a finite set then $|\mathbb{S}|$ is its cardinality, and $x \leftarrow \mathbb{S}$ is the operation of picking an element uniformly at random from $\mathbb{S}$. Let PPT stand for probabilistic polynomial-time. If $\mathcal{A}$ is a probabilistic algorithm, $\mathcal{A}(x_1, x_2, \cdots; \rho)$ is the result of running $\mathcal{A}$ on inputs $x_1, x_2, \cdots$ and random coins ρ. Let $y \leftarrow \mathcal{A}(x_1, x_2, \cdots; \rho)$ denote the experiment of picking ρ at random and letting y be $A(x_1, x_2, \cdots; \rho)$. For simplicity, the random coins are usually omitted when specifying probabilistic algorithms. We say a function $f(\lambda)$ is negligible in a security parameter λ if $f = o(1/\lambda^c)$ for all $c \in \mathbb{N}$. We formally define $\mathsf{order}(m_i, m_j)$ as

$$\mathsf{order}(m_i, m_j) = \begin{cases} 0, & m_i = m_j \\ 1, & m_i > m_j \\ 2, & m_i < m_j \end{cases}$$

where m_i and m_j are two integers.

Bilinear maps. A cryptographic bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are groups with prime order p, satisfies the following three properties:

(1) *Computable:* the map e is efficiently computable.
(2) *Non-degenerate:* if g_1 is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$, then $e(g_1, g_2)$ is a generator of $\mathbb{G}_T$.
(3) *Bilinear:* for all $P \in \mathbb{G}_1$, all $Q \in \mathbb{G}_2$ and all $a, b \in \mathbb{Z}_p^*$, we have $e(P^a, Q^b) = e(P, Q)^{ab}$.

A bilinear map is called symmetric or Type-1 bilinear map if $\mathbb{G}_1$=$\mathbb{G}_2$; otherwise, it is asymmetric. The asymmetric bilinear map is called as Type-2 bilinear map if there is an efficiently computable isomorphism either from $\mathbb{G}_1$ to $\mathbb{G}_2$ or from $\mathbb{G}_2$ to $\mathbb{G}_1$, whereas the one without such isomorphisms is known as Type-3 bilinear maps. From the implementation perspective, Type-3 bilinear map is the most efficient one [27]. Unless specified noted otherwise, all the cryptographic constructions in this work assume Type-3 bilinear maps.

Generic group model. The generic group model is an ideal model proposed in [31]. In the generic group model, each group has a completely random encoding. In other words, for any group operation, the only way that the adversary can get the operation results is issuing queries to an oracle. Inspired by this idea, Boneh et al. [6] introduced the generic bilinear group model, where all the group and pairing operations are accomplished as oracles to the adversary. We follow the idea of proof in work [3], where the lower bound of adversary's advantage is analyzed to prove the security in random oracle model and generic group model.

3 DELEGATABLE EQUALITY-REVEALING ENCODING (DERE)

3.1 Syntax of DERE

Definition 3.1. A delegatable equality-revealing encoding (DERE) scheme consists of four polynomial-time algorithms:

- $\mathsf{KGen}(\lambda)$: it takes as input a security parameter λ, and generates a public/private key pair (pk, sk) for a user in the cryptosystem.
- $\mathsf{Enc}(m, sk)$: it takes as input a message $m \in \mathcal{M}$ and a private key sk where $\mathcal{M}$ is the message space, and outputs an encoding c.
- $\mathsf{TGen}(sk_{\mathsf{ID}_i}, pk_{\mathsf{ID}_j})$: it takes as input an entity ID_i's private key sk_{ID_i} and ID_j's public key pk_{ID_j}, where $i, j \in [N]$ and N is the maximum number of users in the cryptosystem, and outputs a token denoted $t_{\mathsf{ID}_i,\mathsf{ID}_j}$. Here, i and j can be equal.
- $\mathsf{Test}(c, c', \vec{t})$: it takes as input two encodings c, c' and a token vector $\vec{t}$, where $|\vec{t}|$ is 1 or 2. It returns 0 or 1.

	ciphertext size	encryption cost	comparison cost	leakage	delegation	token size
ORE in [12]	$3\log n$	$n\mathsf{prf}$	$O(n\mathsf{mc})$	msdb	×	×
ORE in [11]	$2n(\tau_{\mathbb{G}_1}+\tau_{\mathbb{G}_2})$	$12n\mathsf{e}$	$O(n^2\mathsf{p})$	eq-pattern	×	×
Our DORE	$2n\tau_{\mathbb{G}_1}$	$6n\mathsf{e}$	$O(n\mathsf{p})$	msdb	✓	$2\tau_{\mathbb{G}_2}$

Table 1: Comparison for several ORE/DORE Schemes

Correctness. We say a DERE scheme is *correct* if for any two public/private key pair $(pk_{\mathsf{ID}_i}, sk_{\mathsf{ID}_i})$ and $(pk_{\mathsf{ID}_j}, sk_{\mathsf{ID}_j})$ generated by algorithm KGen, three messages $m_0, m_1, m_2 \in \{0,1\}^*$, where $i, j \in [N]$, then for $c_0 \leftarrow \mathsf{Enc}(m_0, sk_{\mathsf{ID}_i})$, $c_1 \leftarrow \mathsf{Enc}(m_1, sk_{\mathsf{ID}_i})$, $c_2 \leftarrow \mathsf{Enc}(m_2, sk_{\mathsf{ID}_j})$, $t_{\mathsf{ID}_i,\mathsf{ID}_j} \leftarrow \mathsf{TGen}(sk_{\mathsf{ID}_i}, pk_{\mathsf{ID}_j})$ and $t_{\mathsf{ID}_j,\mathsf{ID}_i} \leftarrow \mathsf{TGen}(sk_{\mathsf{ID}_j}, pk_{\mathsf{ID}_i})$, it holds that:

(1) With overwhelming probability, $\mathsf{Test}(c_0, c_1, \vec{t} = \langle t_{\mathsf{ID}_i,\mathsf{ID}_j}\rangle)$ outputs 1 if $m_0 = m_1$, and 0 otherwise. This means that, with a delegation token from user i to any other user (including $j = i$ itself), one can make equality test on the ciphertexts generated by i.
(2) With overwhelming probability, $\mathsf{Test}(c_0, c_2, \vec{t} = \langle t_{\mathsf{ID}_i,\mathsf{ID}_j}, t_{\mathsf{ID}_j,\mathsf{ID}_i}\rangle)$ outputs 1 if $m_0 = m_2$, and 0 otherwise.

3.2 Security Model for DERE

In this section, we present the security model of DERE. The security model is defined by a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, which is denoted by $\mathbf{G}^{\mathsf{DERE}}_{\mathsf{ind},\mathcal{A}}$ as follows.

> Setup: In the cryptosystem, the security parameter and the maximum number of users are set to be λ and N, respectively. The challenger maintains a table $\mathbb{T}_{\mathsf{key}}$, which is initially empty, to manage users' public/private keys.
> Challenge: The challenger $\mathcal{C}$ first chooses a random coin $b \leftarrow \{0,1\}$ as the challenge bit, and the adversary $\mathcal{A}$ can adaptively issue queries to private key generation oracle $\mathcal{O}^{\mathsf{sk}}$, public key generation oracle $\mathcal{O}^{\mathsf{pk}}$, token generation oracle $\mathcal{O}^{\mathsf{tk}}$ and encoding oracle $\mathcal{O}^{\mathsf{enc}}$, with some restrictions to be detailed later.
> Guess: The adversary finally outputs its guess $b' \in \{0,1\}$ and wins the game if $b = b'$.

The adversary's queries are responded by challenger as follows:

- $\mathcal{O}^{\mathsf{sk}}(\cdot)$: For an identity ID_u, where $u \in [N]$, the challenger looks up the table $\mathbb{T}_{\mathsf{key}}$ to get $(\mathsf{ID}_u, pk_{\mathsf{ID}_u}, sk_{\mathsf{ID}_u})$; otherwise, it runs $\mathsf{KGen}(\lambda)$ to get sk_{ID_u} and pk_{ID_u}, and adds $(\mathsf{ID}_u, pk_{\mathsf{ID}_u}, sk_{\mathsf{ID}_u})$ into $\mathbb{T}_{\mathsf{key}}$. Finally, the challenger returns sk_{ID_u} to the adversary.
- $\mathcal{O}^{\mathsf{pk}}(\cdot)$: For an identity ID_u, where $u \in [N]$, the challenger looks up the table $\mathbb{T}_{\mathsf{key}}$ to get $(\mathsf{ID}_u, pk_{\mathsf{ID}_u}, sk_{\mathsf{ID}_u})$; otherwise, it runs $\mathsf{KGen}(\lambda)$ to get sk_{ID_u} and pk_{ID_u}, and adds $(\mathsf{ID}_u, pk_{\mathsf{ID}_u}, sk_{\mathsf{ID}_u})$ into the table $\mathbb{T}_{\mathsf{key}}$. Finally, the challenger returns pk_{ID_u} to the adversary.
- $\mathcal{O}^{\mathsf{tk}}(\cdot,\cdot)$: For a pair of identities $(\mathsf{ID}_u, \mathsf{ID}_v)$, where $u, v \in [N]$, the challenger looks up the table $\mathbb{T}_{\mathsf{key}}$ to get sk_{ID_u} and pk_{ID_v}; Otherwise it performs key generation algorithm to obtain sk_{ID_u} and pk_{ID_v}, and updates $\mathbb{T}_{\mathsf{key}}$. Next, the challenger runs $\mathsf{TGen}(sk_{\mathsf{ID}_u}, pk_{\mathsf{ID}_v})$ to get a token $t_{\mathsf{ID}_u,\mathsf{ID}_v}$. Finally, the challenger returns $t_{\mathsf{ID}_u,\mathsf{ID}_v}$ to the adversary.
- $\mathcal{O}^{\mathsf{enc}}(\cdot,\cdot,\cdot)$: For a tuple (ID, m_0, m_1), where $m_0, m_1 \in \{0,1\}^*$, the challenger runs $\mathbf{Enc}(m_b, sk_{\mathsf{ID}})$ to get an encoding c, and returns c to the adversary. Note that, for DERE, we do not require m_0 and m_1 be of the same length.

Next, we specify the restrictions in the above game. Before that, we first define *open authority* for an identity set. Specifically, an *identity set* $\mathbb{U}$ *with open authority* is defined according to the following two cases:

(1) $|\mathbb{U}| = 1$: Assume $\mathsf{ID}_i \in \mathbb{U}$ for some $i \in [N]$. Then, we require that a pair of identities $(\mathsf{ID}_i, \mathsf{ID}_j)$, for any $j \in [N]$, must have been queried to the token generation oracle $\mathcal{O}^{\mathsf{tk}}$, and the token $t_{\mathsf{ID}_i,\mathsf{ID}_j}$ has been published.
(2) $|\mathbb{U}| \geq 2$: For any pair of two different identities $\mathsf{ID}_i, \mathsf{ID}_j \in \mathbb{U}$, we have that both $(\mathsf{ID}_i, \mathsf{ID}_j)$ and $(\mathsf{ID}_j, \mathsf{ID}_i)$ must have been queried to $\mathcal{O}^{\mathsf{tk}}$, and the tokens $t_{\mathsf{ID}_i,\mathsf{ID}_j}$ and $t_{\mathsf{ID}_j,\mathsf{ID}_i}$ have been published.

For an identity set $\mathbb{U}$ with open authority, we have the guarantee that the encodings generated by all the entities in $\mathbb{U}$ can be tested on the equality for the corresponding messages. Next, we define the leakage function $\mathcal{L}_d$, which is exactly the equality relationship where k is any positive integer.

$$\mathcal{L}_d(m_1, \cdots, m_k) = \{1(m_i = m_j) : 1 \leq i, j \leq k\}.$$

Intuitively, for a tuple (ID, m_0, m_1) issued by $\mathcal{A}$ to encoding oracle $\mathcal{O}^{\mathsf{enc}}$ to obtain an encoding c, where $m_0 \neq m_1$, $\mathcal{A}$ could trivially know the challenge bit b in the following cases:

Case-1: $\mathcal{O}^{\mathsf{tk}}$ and $\mathcal{O}^{\mathsf{enc}}$. $\mathcal{A}$ issues queries to $\mathcal{O}^{\mathsf{tk}}$, and makes $\mathbb{U}$ an identity set with open authority, where $\mathsf{ID} \in \mathbb{U}$. It implies that, for any $\mathsf{ID}_i, \mathsf{ID}_j \in \mathbb{U}$, all the tokens $t_{\mathsf{ID}_i,\mathsf{ID}_j}$ and $t_{\mathsf{ID}_j,\mathsf{ID}_i}$ are published. For an identity $\mathsf{ID}_i \in \mathbb{U}$, $\mathcal{A}$ further issues a tuple $(\mathsf{ID}_i, m_1, m_1)$ to the encoding oracle $\mathcal{O}^{\mathsf{enc}}$ to obtain an encoding c'. Finally, if $\mathsf{ID} = \mathsf{ID}_i$ it runs and outputs $\mathsf{Test}(c, c', \vec{t} = \langle t_{\mathsf{ID},\mathsf{ID}_j}\rangle)$ for $\mathsf{ID}_j \in \mathbb{U}$; otherwise, it runs and outputs $\mathsf{Test}(c, c', \vec{t} = \langle t_{\mathsf{ID},\mathsf{ID}_i}, t_{\mathsf{ID}_i,\mathsf{ID}}\rangle)$.

Case-2: $\mathcal{O}^{\mathsf{sk}}$.

Case-2a: $\mathcal{O}^{\mathsf{sk}}$ and $\mathcal{O}^{\mathsf{pk}}$. $\mathcal{A}$ issues queries to $\mathcal{O}^{\mathsf{sk}}$ to obtains sk_{ID}. Next, it runs algorithm $\mathbf{Enc}(m_1, sk_{\mathsf{ID}})$ to get an encoding c', gets pk_{ID_i} from oracle $\mathcal{O}^{\mathsf{pk}}$, and performs algorithm $\mathsf{TGen}(sk_{\mathsf{ID}}, pk_{\mathsf{ID}_i})$ to get $t_{\mathsf{ID},\mathsf{ID}_i}$, where $i \in [N]$. Finally, $\mathcal{A}$ runs and outputs $\mathsf{Test}(c, c', \vec{t} = \langle t_{\mathsf{ID},\mathsf{ID}_i}\rangle)$.
Case-2b: $\mathcal{O}^{\mathsf{sk}}$ and $\mathcal{O}^{\mathsf{tk}}$ and $\mathcal{O}^{\mathsf{pk}}$. $\mathcal{A}$ issues queries to $\mathcal{O}^{\mathsf{sk}}$ to obtains sk_{ID_i}, where $i \in [N]$ but $\mathsf{ID} \neq \mathsf{ID}_i$, and issues a query to $\mathcal{O}^{\mathsf{tk}}$ to obtains $t_{\mathsf{ID},\mathsf{ID}_i}$. Next, it runs algorithm $\mathbf{Enc}(m_1, sk_{\mathsf{ID}_i})$ to get an encoding c', gets pk_{ID} from oracle $\mathcal{O}^{\mathsf{pk}}$, and performs algorithm $\mathsf{TGen}(sk_{\mathsf{ID}_i}, pk_{\mathsf{ID}})$ to get $t_{\mathsf{ID}_i,\mathsf{ID}}$. Finally, $\mathcal{A}$ runs and outputs $\mathsf{Test}(c, c', \vec{t} = \langle t_{\mathsf{ID},\mathsf{ID}_i}, t_{\mathsf{ID}_i,\mathsf{ID}}\rangle)$.

In order to avoid such a trivial win, some restrictions must get enforced. Specifically, we make the following restrictions in the above game $\mathbf{G}^{\mathsf{DERE}}_{\mathsf{ind},\mathcal{A}}$.

Restriction-1: For each identity set $\mathbb{U}$ with open authority and the tuples $(\mathsf{ID}_1, m_{1,0}, m_{1,1}), \cdots, (\mathsf{ID}_q, m_{q,0}, m_{q,1})$ corresponding to the queries to encoding oracle $\mathcal{O}^{\mathsf{enc}}$, where $\mathsf{ID}_1, \cdots, \mathsf{ID}_q \in \mathbb{U}$, we have $\mathcal{L}_d(m_{1,0}, \cdots, m_{q,0}) = \mathcal{L}_d(m_{1,1}, \cdots, m_{q,1})$. This is to avoid trivial win in the above Case-1.

Restriction-2: If the adversary $\mathcal{A}$ has issued identity ID to private key generation oracle $\mathcal{O}^{\mathsf{sk}}$, then we require

Restriction-2a: $\mathcal{A}$ cannot query (ID, m_0, m_1) to the encoding oracle $\mathcal{O}^{\mathsf{enc}}$ such that $m_0 \neq m_1$. This is to avoid trivial win in the above Case-2a.
Restriction-2b: $\mathcal{A}$ cannot query tuple $(\mathsf{ID}_i, m_0, m_1)$ to the encoding $\mathcal{O}^{\mathsf{enc}}$ oracle, where $m_0 \neq m_1$, $i \in [N]$ and $\mathsf{ID} \neq \mathsf{ID}_i$, if $t_{\mathsf{ID}_i,\mathsf{ID}}$ is obtained from token generation oracle $\mathcal{O}^{\mathsf{tk}}$. This is to avoid trivial win in Case-2b.

Example: For an identity set $\mathbb{U} = \{\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{ID}_3\}$ with open authority, the adversary issues tuples $\{(\mathsf{ID}_1, m_{10}, m_{11}), (\mathsf{ID}_2, m_{20}, m_{21}), (\mathsf{ID}_3, m_{30}, m_{31}), (\mathsf{ID}_4, m_{40}, m_{41}) (\mathsf{ID}_5, m_{50}, m_{51})\}$ to the encoding oracle, obtains sk_{ID_4} from $\mathcal{O}^{\mathsf{sk}}$ and $t_{\mathsf{ID}_5,\mathsf{ID}_4}$. We have

(1) $\mathcal{L}_d(m_{10}, m_{20}, m_{30}) = \mathcal{L}_d(m_{11}, m_{21}, m_{31})$ according to Restriction-1.
(2) $m_{40} = m_{41}$ according to Restriction-2a.
(3) $m_{50} = m_{51}$ according to Restriction-2b.

Based on the above security game $\mathbf{G}^{\mathsf{DERE}}_{\mathsf{ind},\mathcal{A}}$, we define the adversary $\mathcal{A}$'s advantage in $\mathbf{G}^{\mathsf{DERE}}_{\mathsf{ind},\mathcal{A}}$ as

$$\mathrm{Adv}^{\mathsf{DERE}}_{\mathsf{ind},\mathcal{A}}(\lambda) = |\Pr[b = b'] - \frac{1}{2}|.$$

Next, we define the indistinguishability of encodings under authority delegation and distinct chosen-plaintext attack (IND-AD-DCPA).

Definition 3.2. A DERE scheme is IND-AD-DCPA secure, if for any PPT adversary $\mathcal{A}$ its advantage $\mathrm{Adv}^{\mathsf{DERE}}_{\mathsf{ind},\mathcal{A}}(\lambda)$ is negligible.

3.3 Construction and Analysis of DERE

Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be three groups with the same prime order p, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a Type-3 bilinear map. We further assume that g_1 and g_2 are generators in $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Let $H : \{0,1\}^* \to \mathbb{G}_1$ be a cryptographic hash function. The public system parameters are $\mathsf{pp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, H)$. Our DERE scheme consists of the following four algorithms:

- $\mathsf{KGen}(\lambda)$: On input the security parameter $\lambda = |p|$, it chooses a, x uniformly at random from $\mathbb{Z}_p^*$ and sets $pk = g_2^a, sk = (a, x)$.
- $\mathsf{Enc}(m, sk)$: On input the message $m \in \{0,1\}^*$ and user's private key $sk = (a, x)$, the algorithm picks $r \leftarrow \mathbb{Z}_p$, and computes
$$c_1 = (g_1^{rx} H(m))^a, \quad c_2 = g_1^r.$$
The algorithm outputs $c = (c_1, c_2)$ as the encoding of m.
- $\mathsf{TGen}(sk_{\mathsf{ID}_i}, pk_{\mathsf{ID}_j})$: On input the private key $sk_{\mathsf{ID}_i} = (a, x)$ of user ID_i, public key $pk_{\mathsf{ID}_j} = g_2^b$ of user ID_j, where $i, j \in [N]$, the algorithm computes
$$t_1 = pk_{\mathsf{ID}_j} = g_2^b, \quad t_2 = (pk_{\mathsf{ID}_j})^{a \cdot x} = g_2^{abx}.$$
The algorithm outputs the token $t = (t_1, t_2)$.
- $\mathsf{Test}(c, c', \vec{t})$: On input encodings c, c' and a token vector $\vec{t}$, it parses $c = (c_1, c_2)$ and $c' = (c_1', c_2')$. Next, the algorithm acts as follows:
  - If $|\vec{t}| = 1$, where $\vec{t} = \langle t\rangle$, it parses $t = (t_1, t_2)$, and then computes
  $$d_1 = \frac{e(c_1, t_1)}{e(c_2, t_2)}, \quad d_2 = \frac{e(c_1', t_1)}{e(c_2', t_2)}.$$
  - If $|\vec{t}| = 2$, where $\vec{t} = \langle t, t'\rangle$, it parses $t = (t_1, t_2)$ and $t' = (t_1', t_2')$, and then computes
  $$d_1 = \frac{e(c_1, t_1)}{e(c_2, t_2)}, \quad d_2 = \frac{e(c_1', t_1')}{e(c_2', t_2')}.$$
  The algorithm outputs 1 if $d_1 = d_2$, and 0 otherwise.

THEOREM 3.3. *Assuming H is collision-resistant, the above DERE scheme satisfies correctness.*

The correctness of DERE can be simply verified, and we omit the detail proof here.

The following theorem shows that our proposed DERE scheme achieves the IND-AD-DCPA security in random oracle model and generic bilinear group model, and works in [3, 29, 30] use both random oracle model and generic (bilinear) group model to analyze the security. The proof follows the methodology in [3, 6].

THEOREM 3.4. *In the random oracle model and generic bilinear group model, for any adversary $\mathcal{A}$ against the above DERE scheme, we have A's advantage to win game $\mathbf{G}^{DERE}_{ind,\mathcal{A}}$ is*

$$Adv^{DERE}_{ind,\mathcal{A}} \leq \frac{9(q + 5q_e + 2q_{tk} + q_{pk} + q_H + 2)^2}{2p},$$

where $\mathcal{A}$ makes a total of at most q queries to the oracles to compute the group operation in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ and the pairing operation $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ in the generic bilinear group model, makes q_e encoding queries, q_{tk} token generation queries, q_{pk} public key generation queries and q_H hash queries.

Table 2: Description of polynomials

$p_{1,i} \in \mathbb{F}_p[X_{1,1}, \cdots, X_{1,q_1}, R_1, \cdots, R_{q_\mathsf{e}}, A_{1,1}, \cdots, A_{1,q_1}, M_1, \cdots, M_{q_3}, Z_1, \cdots, Z_{q_2}]$
$p_{2,i} \in \mathbb{F}_p[X_{2,1}, \cdots, X_{2,q_4}, A_{2,1}, \cdots, A_{2,q_7}]$
$q_i \in \mathbb{F}_p[X_{t,1}, \cdots, X_{t,q_5}, R_1, \cdots, R_{q_\mathsf{e}}, A_{t,1}, \cdots, A_{t,q_6}, M_1, \cdots, M_{q_3}, Z_1, \cdots, Z_{q_2}]$,

Proof. First, the challenger $\mathcal{C}$ maintains three lists of pairs, $\mathbb{L}_1 = \{(p_{1,i}, \epsilon_{1,i}) : i = 1, \cdots, \tau_1\}$, $\mathbb{L}_2 = \{(p_{2,i}, \epsilon_{2,i}) : i = 1, \cdots, \tau_2\}$ and $\mathbb{L}_t = \{(q_i, \epsilon_{t,i}) : i = 1, \cdots, \tau_t\}$, where τ_1, τ_2 and τ_t are initialized to be $1, 1$ and 0, respectively. We have polynomials $p_{1,i}, p_{2,i}$ and q_i which are described in table 2, where $q_1, q_2 \leq q_\mathsf{e}$, $q_3 \leq q_H + 2q_2$, $q_4 \leq q_\mathsf{tk}$, $q_5 \leq q_1 + q_4$, $q_6 \leq q_1 + q_\mathsf{pk} + 2q_\mathsf{tk}$, $q_7 \leq q_\mathsf{pk} + 2q_\mathsf{tk}$, q_e is the number of encoding queries, q_H is the number of hash queries, q_tk is the number of token generation queries, q_pk is the number of public key queries. The group element $\epsilon_{*,*}$ in $\mathbb{G}_1$, $\mathbb{G}_2$ or $\mathbb{G}_t$ is actually represented as strings in $\{0,1\}^{\mathsf{len}}$, where len denotes the binary length of string that is utilized to represent element in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_t$. In fact, lists $\mathbb{L}_0$ and $\mathbb{L}_1$ are initialized to record the generators in $\mathbb{G}_1$ and $\mathbb{G}_2$.

In the above polynomials, more concretely, variables A_* and X_* are used to simulate the private key, R_* is for the simulation of the random number in encoding generation, M_* is for the simulation of the messages with respect to the hash function H, and Z_* is for the simulation of the message with respect to the challenge encoding, the tuple of which contains two different messages.

In addition, $\mathcal{C}$ maintains a table $\mathbb{T}_H = \{(m_i, M_i) : i = 1, \cdots, m_{q_3}\}$ to simulate hash function $H : \{0,1\}^* \to \mathbb{G}_1$ as a random oracle to $\mathcal{A}$, a table $\mathbb{T} = \{(m_{i,0}, m_{i,1}, Z_i) : i = 1, \cdots, q_2\}$ to simulate the challenge encoding, and a table $\mathbb{T}_k = \{(\mathsf{ID}_i, a_i, x_i, A_i, X_i) : i = 1, \cdots, q_\mathsf{k}\}$ to simulate the user's public/private key, where $q_\mathsf{k} \leq q_\mathsf{e} + q_\mathsf{tk} + q_\mathsf{pk} + q_\mathsf{sk}$ and q_sk is the number of queries to the private-key generation oracle.

Group operations in $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$. For a query in $\mathbb{G}_1$, two strings $\epsilon_{1,i}$ and $\epsilon_{1,j}$, where $1 \leq i, j \leq \tau_1$, are issued. To handle group multiplication or division, polynomial addition or subtraction is utilized. In other words, $p_{1,\tau'} = p_{1,i} \pm p_{1,j}$ are used for $\epsilon_{1,i} \cdot \epsilon_{1,j}$ and $\epsilon_{1,i}/\epsilon_{1,j}$, respectively. If $p_{1,\tau'_0} = p_{1,\ell}$ for $\ell \leq \tau_1$, then we set $\epsilon_{1,\tau'_1} = \epsilon_{1,\ell}$; otherwise, we set $\epsilon_{1,\tau'}$ to a new random string in $\{0,1\}^{\mathsf{len}} \backslash \{\epsilon_{1,1}, \cdots, \epsilon_{1,\tau_1}\}$. Meanwhile, the challenger $\mathcal{C}$ adds the pair $(p_{1,\tau'_1}, \epsilon_{1,\tau'_1})$ to the list $\mathbb{L}_1$ and updates $\tau_1 = \tau_1 + 1$. Finally, ϵ_{1,τ'_1} is sent to $\mathcal{A}$ as a reply to the oracle query of group operation in $\mathbb{G}_1$.

Queries for group operations in $\mathbb{G}_2, \mathbb{G}_T$ are analogously handled as in $\mathbb{G}_1$ with $\mathbb{L}_2, \tau_2$ and $\mathbb{L}_t, \tau_t$.

Bilinear map. For a query in this case, two strings $\epsilon_{1,i}$ and $\epsilon_{2,j}$ are issued, where $1 \leq i \leq \tau_1$ and $1 \leq j \leq \tau_2$. Multiplication for $p_{1,i}$ and $p_{2,j}$ are performed, and $q_{\tau'_t} = p_{1,i} \cdot p_{2,j}$. If $q_{\tau'_t} = q_\ell$ for $\ell \leq \tau_t$, then we set $\epsilon_{t,\tau'_t} = \epsilon_{t,\ell}$; otherwise, we set ϵ_{t,τ'_t} to a new random string in $\{0,1\}^{\mathsf{len}} \backslash \{\epsilon_{t,1}, \cdots, \epsilon_{t,\tau_t}\}$. Meanwhile, the challenger $\mathcal{C}$ adds the pair $(q_{\tau'_t}, \epsilon_{t,\tau'_t})$ to the list $\mathbb{L}_t$ and updates $\tau_t = \tau_t + 1$. Finally, ϵ_{t,τ'_t} is sent to $\mathcal{A}$ as a reply to the oracle query of pairing operation.

Hash function H. For a query on message m, if there exists an entry (m, M) in the table $\mathbb{T}_H$, the challenger finds the $\epsilon_{1,j}$ corresponding to M in $\mathbb{L}_1$ and returns $\epsilon_{1,j}$ to the adversary, where $1 \leq j \leq \tau_1$; otherwise, it chooses a new variable M and a random string $\epsilon_{1,\tau_1+1} = \{0,1\}^{\mathsf{len}} \backslash \{\epsilon_{1,1}, \cdots, \epsilon_{1,\tau}\}$, and updates $\mathbb{T}_H = \mathbb{T}_H \cup \{(m, M)\}$ and $\mathbb{L}_1 = \mathbb{L}_1 \cup \{(M, \epsilon_{1,\tau_1+1})\}$.

Encoding generation. For a tuple (ID, m_0, m_1), $\mathcal{C}$ acts as follows according to two different cases:

(1) $m_0 = m_1$: $\mathcal{C}$ looks up the table $\mathbb{T}_H$ to find the entry (m_0, M) and gets M; otherwise, $\mathcal{C}$ chooses a new variable M, and updates the table $\mathbb{T}_H = \mathbb{T}_H \cup \{(m_0, M)\}$.
(2) $m_0 \neq m_1$: $\mathcal{C}$ looks up the table $\mathbb{T}$ to find the entry (m_0, m_1, Z) and gets Z; otherwise, $\mathcal{C}$ chooses a new variable Z, and updates the table $\mathbb{T} = \mathbb{T} \cup \{(m_0, m_1, Z)\}$. For m_b where $b \in \{0,1\}$, if the table $\mathbb{T}_H$ does not have the entries corresponding to m_b, then $\mathcal{C}$ chooses a new variable M_b, and adds (m_b, M_b) into $\mathbb{T}$.

If there exist variables X and A corresponding to the identity ID in the table $\mathbb{T}_k$, $\mathcal{C}$ gets the variables X and A; otherwise, $\mathcal{C}$ chooses two new variable X and A corresponding to the identity ID, and updates $\mathbb{T}_k = \mathbb{T}_k \cup (\mathsf{ID}, *, *, A, X)$. Next, $\mathcal{C}$ chooses a new variable R, and manages list $\mathbb{L}_1$ to get $\epsilon_{1,0}$ corresponding to polynomial $(RX + M)A$ ($m_0 = m_1$, for the above case 1) or $(RX + Z)A$ ($m_0 \neq m_1$, for the above case 2), and $\epsilon_{1,1}$ corresponding to polynomial R. Finally, $(\epsilon_{1,0}, \epsilon_{1,1})$ is sent to $\mathcal{A}$ as a reply to the encoding query.

Key generation. Key generation is simulated as follows.

- Private key: For an identity ID, $\mathcal{C}$ looks up the table $\mathbb{T}_k$ to find a and x corresponding to the entry $(\mathsf{ID}, a, x, A, X)$; otherwise, $\mathcal{C}$ randomly chooses $a, x \leftarrow \mathbb{F}_p$, and updates the entry corresponding to ID in $\mathbb{T}_k$ to be $(\mathsf{ID}, a, x, A, X)$. Finally, $\mathcal{C}$ returns (a, x) to $\mathcal{A}$ as the reply to $\mathcal{A}$'s query for private key generation.
- Public key: For an identity ID, $\mathcal{C}$ looks up the table $\mathbb{T}_k$ to find A corresponding to the entry $(\mathsf{ID}, *, *, A, *)$, and manages $\mathbb{L}_2$ to get $\epsilon_{2,*}$ corresponding to A. Finally, $\mathcal{C}$ returns $\epsilon_{2,*}$ to $\mathcal{A}$ as the reply to public key query.

Token generation. For a pair of identities $(\mathsf{ID}_i, \mathsf{ID}_j)$, where $1 \leq i, j \leq N$, $\mathcal{C}$ looks up table $\mathbb{T}_k$ to get the values A_i and X_i corresponding to the entry $(\mathsf{ID}_i, *, *, A_i, X_i)$; otherwise, $\mathcal{C}$ chooses two new variables A_i and X_i and updates the entry corresponding to the identity ID_i to be $(\mathsf{ID}_i, *, *, A_i, X_i)$. $\mathcal{C}$ gets A_j corresponding to ID_j with the management of $\mathbb{T}_k$ and the token generation algorithm. Next, $\mathcal{C}$ uses variables A_i, X_i and A_j to simulate the token $\mathsf{t}_{\mathsf{ID}_i,\mathsf{ID}_j}$. Concretely, $\mathcal{C}$ manages lists $\mathbb{L}_2$ to get $\epsilon_{2,0}$ and $\epsilon_{2,1}$ corresponding to polynomials A_j and $A_i A_j X_i$ in $\mathbb{G}_2$, respectively. Finally, $(\epsilon_{2,0}, \epsilon_{2,1})$ is returned to $\mathcal{A}$ as the reply to token generation query.

Once $\mathcal{A}$ finishes the challenge phase, it outputs a bit $b' \in \{0,1\}$ as its guess. For any two A and X corresponding to the identity ID whose private key has been queried, $\mathcal{C}$ assigns values a and x to the variables A and X, respectively. After the assignment, we have new polynomials with respect to variables A_*, X_*, R_*, M_* and Z_*, and denote by $\overline{p}_{1,*}, \overline{p}_{2,*}$ and $\overline{q}_*$ these new polynomials corresponding to $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$, respectively.

For the variables in $\overline{p}_{1,*}, \overline{p}_{2,*}$ and $\overline{q}_*$, each variable Z_i, which is recorded in entry $(m_{i,0}, m_{i,1}, Z_i)$ of $\mathbb{T}$ where $1 \leq i \leq q_{\mathsf{e}}$, is substituted by M corresponding to the value $m_{i,b}$ recorded in tables $\mathbb{T}_H$. Next, $\mathcal{C}$ independently chooses values a_*, x_*, r_* and α_* from $\mathbb{F}_p$, and assigns them to the variables A_*, X_*, R_* and M_*, respectively.

If the simulation is perfect in this method, then $\mathcal{A}$'s advantage to win the game is exactly 0, as b is independent of $\mathcal{A}$'s view. Next, we show that the probability that simulation fails is negligible, which then concludes the proof. The failure of simulation depends upon the following three conditions.

(1) $\overline{p}_{1,\ell} - \overline{p}_{1,\ell'} = 0$ with respect to the values $x_1, \cdots, x_{q'_1}$, $r_1, \cdots, r_{q_{\mathsf{e}}}$, $a_1, \cdots, a_{q'_1}$, $\alpha_1, \cdots, \alpha_{q'_5}$, and polynomials $\overline{p}_{1,\ell'}$ and $\overline{p}_{1,j}$ are not equal.
(2) $\overline{p}_{2,\ell} - \overline{p}_{2,j} = 0$ with respect to the values $x_1, \cdots, x_{q'_2}$, $a_1, \cdots, a_{q'_6}$, and polynomials $\overline{p}_{2,i}$ and $\overline{p}_{2,j}$ are not equal.
(3) $\overline{q}_\ell - \overline{q}_{\ell'} = 0$ with respect to the values $x_1, \cdots, x_{q'_3}$, $r_1, \cdots, r_{q_{\mathsf{e}}}$, $a_1, \cdots, a_{q'_4}$, $\alpha_1, \cdots, \alpha_{q'_5}$, and polynomials $\overline{q}_i$ and $\overline{q}_j$ are not equal.

where $q'_1 \leq q_{\mathsf{e}}$, $q'_2 \leq q_{\mathsf{tk}}$, $q'_3 \leq q'_1 + q'_2$, $q'_4 \leq q'_1 + q_{\mathsf{pk}} + 2q_{\mathsf{tk}}$, $q'_5 \leq q_H + 2q_{\mathsf{e}}$ and $q'_6 \leq q_{\mathsf{pk}} + 2q_{\mathsf{tk}}$. Let fail be the event that any one of the above three conditions holds. Next, we bound the probability that fail occurs.

We consider the substitution from Z_i to $M_{i,b}$, where $M_{i,b}$ corresponds to $m_{i,b}$ in $\mathbb{T}_H$ and $(m_{i,0}, m_{i,1}, Z_i)$ is recorded in $\mathbb{T}$ such that $m_{i,0} \neq m_{i,1}$. We claim that the symbol substitution does not create any new equality relationship for polynomials w.r.t. $\overline{p}_{2,*}$. In other words, if $\overline{p}_{2,\ell} - \overline{p}_{2,\ell'} \neq 0$ for all ℓ, ℓ' before the substitution, then $\overline{p}_{2,\ell} - \overline{p}_{2,\ell'} \neq 0$ also holds after we set $Z_i = M_{i,b}$, where $1 \leq \ell, \ell' \leq \tau_2$. This obviously holds, as the substitution only affects the encodings which are unrelated to polynomials w.r.t. $\overline{p}_{2,*}$.

Next, we show that the substitution does not create any new equality relationship for polynomials w.r.t. $\overline{p}_{1,*}$ and $\overline{q}_*$, respectively. For the substitution from polynomial $(R_v X_{\mathsf{ID}_i} + Z_\ell)A_{\mathsf{ID}_i}$ to polynomial $(R_v X_{\mathsf{ID}_i} + M_{\ell,b})A_{\mathsf{ID}_i}$ with respect to $\overline{p}_{1,*}$, where $i \in [N]$ and $1 \leq v, \ell \leq q_{\mathsf{e}}$, we analyze as follows:

$\mathsf{Polynomials}$ w.r.t. $\overline{p}_{1,*}$: Observe that the polynomial $(R_v X_{\mathsf{ID}_i} + Z_\ell)A_{\mathsf{ID}_i}$ contains a term $R_v X_{\mathsf{ID}_i} A_{\mathsf{ID}_i}$ that is independent of other polynomials with respect to $\overline{p}_{1,*}$. The reason is that, with other polynomials (other than this specific polynomial $(R_v X_{\mathsf{ID}_i} + Z_\ell)A_{\mathsf{ID}_i}$) with respect to $\overline{p}_{1,*}$, we cannot calculate a polynomial $\overline{p}_{1,\ell}$ that contains a term $R_v X_{\mathsf{ID}_i}$ by the assumption that R_v is not used to simulate other encodings. Therefore, polynomials $(R_v X_{\mathsf{ID}_i} + Z_\ell)A_{\mathsf{ID}_i}$ and $(R_v X_{\mathsf{ID}_i} + M_{\ell,b})A_{\mathsf{ID}_i}$ are independent of other polynomials w.r.t. $\overline{p}_{1,\ell}$, which implies that the substitution does not create any new equality relationship for polynomials w.r.t. $\overline{p}_{1,*}$.

$\mathsf{Polynomials}$ w.r.t. $\overline{q}_*$: We further consider the following two cases:

- If $\mathcal{A}$ has not issued queries to get a token $t_{\mathsf{ID}_i,\mathsf{ID}_j}$ for an arbitrary $i \in [N]$, then there is no polynomial $\overline{p}_{2,\ell}$ which contains X_{ID_i}. It implies that we cannot calculate a polynomial $\overline{q}_{\ell'}$, which contains $R_v X_{\mathsf{ID}_i}$, without these two polynomials $(R_v X_{\mathsf{ID}_i} + Z_\ell)A_{\mathsf{ID}_i}$ or $(R_v X_{\mathsf{ID}_i} + M_{\ell,b})A_{\mathsf{ID}_i}$ with respect to $\overline{p}_{1,*}$. Therefore, the polynomials w.r.t. $\overline{q}_*$ generated by $(R_v X_{\mathsf{ID}_i} + Z_\ell)A_{\mathsf{ID}_i}$ and $(R_v X_{\mathsf{ID}_i} + M_{\ell,b})A_{\mathsf{ID}_i}$ are independent of other polynomials w.r.t. $\overline{q}_*$, which implies that the substitution does not create any new equality relationship for a polynomial w.r.t. $\overline{q}_*$.
- Now, we consider the case that $\mathcal{A}$ has issued queries to get a token $t_{\mathsf{ID}_i,\mathsf{ID}_j}$ for an arbitrary $j \in [N]$. Since the two polynomials $(R_v X_{\mathsf{ID}_i} + M_{\ell,b})A_{\mathsf{ID}_i}$ and $(R_v X_{\mathsf{ID}_i} + Z_\ell)A_{\mathsf{ID}_i}$, with respect to $\overline{p}_{1,*}$, contain a term $R_v X_u A_u$, the possible new equality for polynomials w.r.t. $\overline{q}_*$ comes from the multiplication (in simulation for the pairing operation) of R_v w.r.t. $\overline{p}_{1,*}$ and $X_{\mathsf{ID}_i} A_{\mathsf{ID}_i} A_{\mathsf{ID}_j}$ w.r.t. $\overline{p}_{2,*}$. In this case, it creates a polynomial $A_{\mathsf{ID}_i} A_{\mathsf{ID}_j} M_{\ell,b}$ or $A_{\mathsf{ID}_i} A_{\mathsf{ID}_j} Z_\ell$ w.r.t. $\overline{q}_*$. There are only two possible ways from which the polynomial $A_{\mathsf{ID}_i} A_{\mathsf{ID}_j} Z_\ell$ can be derived:
  (1) From polynomials $(R_{v'} X_{\mathsf{ID}_i} + Z_\ell)A_{\mathsf{ID}_i}$ and $R_{v'}$ w.r.t. $\overline{p}_{1,*}$, and polynomials $A_{\mathsf{ID}_i} A_{\mathsf{ID}_j} X_{\mathsf{ID}_i}$ and A_{ID_j} w.r.t. $\overline{p}_{2,*}$;
  (2) or, from polynomials $(R_{v'} X_{\mathsf{ID}_j} + Z_\ell)A_{\mathsf{ID}_j}$ and $R_{v'}$ w.r.t. $\overline{p}_{1,*}$, and polynomials $A_{\mathsf{ID}_i} A_{\mathsf{ID}_j} X_{\mathsf{ID}_j}$ and A_{ID_i} w.r.t. $\overline{p}_{2,*}$.

  If the substitution creates new equality relationship for a polynomial w.r.t. $\overline{q}_*$, it implies that the adversary has issued queries in either of the following two cases:
  – The first case is that $\mathcal{A}$ has issued the tuples $T = (\mathsf{ID}_i, m_0, m_1)$ and $T' = (\mathsf{ID}_i, m'_0, m'_1)$ to $\mathcal{O}^{\mathsf{enc}}$, where $m_0 \neq m_1$, $m_b = m'_b$, $m_{1-b} \neq m'_{1-b}$, $t_{\mathsf{ID}_i,\mathsf{ID}_j}$ has been published, and sk_{ID_i} cannot be corrupted. The encoding corresponding to the tuple T is simulated by polynomials $(R_{v'} X_{\mathsf{ID}_i} + Z_\ell)A_{\mathsf{ID}_i}$ and $R_{v'}$ w.r.t. $\overline{p}_{1,*}$, and this encoding can be computed with the token $t_{\mathsf{ID}_i,\mathsf{ID}_j}$ to get an intermediate result in $\mathbb{G}_T$, where the result is simulated by the polynomial $Z_\ell A_{\mathsf{ID}_i} A_{\mathsf{ID}_j}$ w.r.t. $\overline{q}_*$. The intermediate polynomials w.r.t. $\overline{q}_*$ are $Z_\ell A_{\mathsf{ID}_i} A_{\mathsf{ID}_j}$, $Z'_\ell A_{\mathsf{ID}_i} A_{\mathsf{ID}_j}$ before the substitution, and $M_{\ell,b} A_{\mathsf{ID}_i} A_{\mathsf{ID}_j}$, $M_{\ell,b} A_{\mathsf{ID}_i} A_{\mathsf{ID}_j}$ after the substitution. We observe that the substitution creates a new equality relationship for polynomials w.r.t. $\overline{q}_*$ from the intermediate results. However, the queries for $\mathcal{O}^{\mathsf{enc}}$ violate the $\mathsf{Restriction\text{-}1}$ of DERE, as we have $\mathcal{L}_{\mathsf{d}}(m_0, m'_0) \neq \mathcal{L}_{\mathsf{d}}(m_1, m'_1)$ here.
  – The second case is that $\mathcal{A}$ has issued the tuples $T = (\mathsf{ID}_i, m_0, m_1)$ and $T' = (\mathsf{ID}_j, m'_0, m'_1)$ to $\mathcal{O}^{\mathsf{enc}}$, where $m_0 \neq m_1$, $m_b = m'_b$, $m_{1-b} \neq m'_{1-b}$, $t_{\mathsf{ID}_i,\mathsf{ID}_j}$ and $t_{\mathsf{ID}_j,\mathsf{ID}_i}$ have been published, and $sk_{\mathsf{ID}_i}, sk_{\mathsf{ID}_j}$ cannot be corrupted. Similarly to the above case, the substitution creates a new equality relationship for

polynomials w.r.t. $\overline{q}_*$ from the intermediate results. The queries for $\mathcal{O}^{\mathsf{enc}}$ in this case also violate the Restriction-1 of DERE, as we have $\mathcal{L}_{\mathsf{d}}(m_0, m) \neq \mathcal{L}_{\mathsf{d}}(m_1, m)$ here.

Therefore, the substitution does not create new equality relationship for polynomials w.r.t. $\overline{q}_*$.

Next, we need to bound the probability that the "unexpected collisions" happen in polynomials w.r.t. $\overline{p}_{1,*}, \overline{p}_{2,*}$ and $\overline{q}_*$. Note that the maximum degree in polynomials w.r.t. $\overline{p}_{1,*}, \overline{p}_{2,*}$ and $\overline{q}_*$ is 6, and there are no more than $3\binom{q+q'_3+q'_4+q'_5+q_{\mathsf{e}}+2}{2}$ for such pairs $(\overline{p}_{1,i}, \overline{p}_{1,j})$, $(\overline{p}_{2,i}, \overline{p}_{2,j})$ and $(\overline{q}_i, \overline{q}_j)$. By the Schwartz-Zippel lemma, we have

$$\Pr[\mathsf{fail}] \leq 3\binom{q+q'_3+q'_4+q'_5+q_{\mathsf{e}}+2}{2}\frac{6}{p} \leq \frac{9(q+5q_{\mathsf{e}}+2q_{\mathsf{tk}}+q_{\mathsf{pk}}+q_H+2)^2}{p}.$$

If the event fail does not occur, $\mathcal{A}$'s view is identical to that in the real game. Since b is exactly independent from $\mathcal{A}$'s view in this case, we have $\Pr[b = b' | \neg\mathsf{fail}] = 1/2$. Consequently, we have

$$\Pr[b = b'] \leq \Pr[b = b' | \neg\mathsf{fail}] \cdot \Pr[\neg\mathsf{fail}] + \Pr[\mathsf{fail}] = \frac{1}{2} + \frac{\Pr[\mathsf{fail}]}{2}$$

$$\Pr[b = b'] \geq \Pr[b = b' | \neg\mathsf{fail}] \cdot \Pr[\neg\mathsf{fail}] = \frac{1}{2} - \frac{\Pr[\mathsf{fail}]}{2}$$

That is, $|\Pr[b = b'] - \frac{1}{2}| \leq \frac{\Pr[\mathsf{fail}]}{2}$, which finishes the proof of Theorem 3.4. □

4 DELEGATABLE ORDER-REVEALING ENCRYPTION (DORE)

4.1 Syntax of DORE

Definition 4.1. A DORE scheme consists of four polynomial-time algorithms:

- $\mathsf{DORE.KGen}(\lambda)$: it takes as input a security parameter λ, and generates the public/private key pair (pk, sk) for a user in the cryptosystem.
- $\mathsf{DORE.Enc}(m, sk)$: it takes as input a message $m \in \mathcal{M}$ and the private key sk, where $\mathcal{M}$ is the message space, and generates a ciphertext c of m.
- $\mathsf{DORE.TGen}(sk_{\mathsf{ID}_i}, pk_{\mathsf{ID}_j})$: it takes as input ID_i's private key sk_{ID_i} and ID_j's public key pk_{ID_j}, where $i, j \in [N]$ and N is the maximum number of users in the system, and outputs a token denoted $t_{\mathsf{ID}_i,\mathsf{ID}_j}$.
- $\mathsf{DORE.Comp}(c, c', \vec{t})$: it takes as input two ciphertexts c, c' and a token vector $\vec{t}$. It returns 0, 1 or 2.

Similar to DERE, the case of ciphertext comparison for a single entity is implicitly included in the above description of DORE. Specifically, with a token vector $\vec{t} = \langle t_{\mathsf{ID}_i,\mathsf{ID}_j} \rangle$, where $j \in [N]$, we can perform the comparison algorithm $\mathsf{DORE.Comp}$ on any two ciphertexts c and c' being encrypted by the private key of user ID_i.

Correctness. For any two public/private key pairs $(pk_{\mathsf{ID}_i}, sk_{\mathsf{ID}_i})$, $(pk_{\mathsf{ID}_j}, sk_{\mathsf{ID}_j})$ generated by $\mathsf{DORE.TGen}$, where $i, j \in [N]$, and three messages $m_0, m_1, m_2 \in \mathcal{M}$, let $t_{\mathsf{ID}_i,\mathsf{ID}_j}$ and $t_{\mathsf{ID}_j,\mathsf{ID}_i}$ be two tokens generated by $\mathsf{DORE.TGen}(sk_{\mathsf{ID}_i}, pk_{\mathsf{ID}_j})$ and $\mathsf{DORE.TGen}(sk_{\mathsf{ID}_j}, pk_{\mathsf{ID}_i})$, respectively, let c_0, c_1 and c_2 be three ciphertexts generated by $\mathsf{DORE.Enc}(m_0, sk_{\mathsf{ID}_i})$, $\mathsf{DORE.Enc}(m_1, sk_{\mathsf{ID}_i})$ and $\mathsf{DORE.Enc}(m_2, sk_{\mathsf{ID}_j})$, respectively. A DORE scheme is *correct*, if it holds that:

- With overwhelming probability, we have

$$\mathsf{DORE.Comp}(c_0, c_1, \vec{t} = \langle t_{\mathsf{ID}_i,\mathsf{ID}_j} \rangle) = \mathsf{order}(m_i, m_j).$$

  This means that, with a delegation token from user ID_i to any other user (including $j = i$ itself), one can do order comparison on the ciphertexts generated by ID_i.
- With overwhelming probability, we have

$$\mathsf{DORE.Comp}(c_0, c_2, \vec{t} = \langle t_{\mathsf{ID}_i,\mathsf{ID}_j}, t_{\mathsf{ID}_j,\mathsf{ID}_i} \rangle) = \mathsf{order}(m_i, m_j).$$

4.2 Security Model for DORE

The security definition of DORE is based on the following security game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, which is denoted by $\mathbf{G}^{\mathsf{dore}}_{\mathsf{ind},\mathcal{A}}$:

Setup: The security parameter and the maximum number of users are set to be λ and N, respectively. The challenger maintains a table $\mathbb{T}_{\mathsf{key}}$, which is initially set to be empty, to manage users' public/private keys.

Challenge: The challenger $\mathcal{C}$ first picks a random coin $b \in \{0, 1\}$ as the challenge bit, and the adversary $\mathcal{A}$ can adaptively issue queries to public key oracle $\mathcal{O}^{\mathsf{pk}}_{\mathsf{dore}}$, private key oracle $\mathcal{O}^{\mathsf{sk}}_{\mathsf{dore}}$, token generation oracle $\mathcal{O}^{\mathsf{tk}}_{\mathsf{dore}}$ and encryption oracle $\mathcal{O}^{\mathsf{enc}}_{\mathsf{dore}}$ with some restrictions to be specified below.

Guess: The adversary $\mathcal{A}$ finally outputs its guess $b' \in \{0, 1\}$ and wins the game if $b = b'$.

The adversary's queries are responded by the challenger as follows:

- $\mathcal{O}^{\mathsf{pk}}_{\mathsf{dore}}(\cdot)$: For an identity ID_u, where $u \in [N]$, the challenger looks up the table $\mathbb{T}_{key}$ to get pk_{ID_u} and sk_{ID_u}; otherwise, it runs $\mathsf{DORE.KGen}(\lambda)$ to get pk_{ID_u} and sk_{ID_u}, and adds $(\mathsf{ID}_u, pk_{\mathsf{ID}_u}, sk_{\mathsf{ID}_u})$ into the table $\mathbb{T}_{\mathsf{key}}$. Finally, the challenger returns pk_{ID_u} to the adversary.
- $\mathcal{O}^{\mathsf{sk}}_{\mathsf{dore}}(\cdot)$: For an identity ID_u, where $u \in [N]$, the challenger looks up the table $\mathbb{T}_{\mathsf{key}}$ to get pk_{ID_u} and sk_{ID_u}; otherwise, it runs the algorithm $\mathsf{DORE.KGen}(\lambda)$ to get pk_{ID_u} and sk_{ID_u}, and adds $(\mathsf{ID}_u, pk_{\mathsf{ID}_u}, sk_{\mathsf{ID}_u})$ into the table $\mathbb{T}_{\mathsf{key}}$. Finally, the challenger returns sk_{ID_u} to the adversary.
- $\mathcal{O}^{\mathsf{tk}}_{\mathsf{dore}}(\cdot,\cdot)$: For a pair of identities $(\mathsf{ID}_u, \mathsf{ID}_v)$, where $u, v \in [N]$, the challenger looks up the table $\mathbb{T}_{key}$ to get sk_{ID_u} and pk_{ID_v}, and runs $\mathsf{DORE.TGen}(sk_{\mathsf{ID}_u}, pk_{\mathsf{ID}_v})$ to generate the token $t_{\mathsf{ID}_u,\mathsf{ID}_v}$. Finally, the challenger returns $t_{\mathsf{ID}_u,\mathsf{ID}_v}$ to the adversary.
- $\mathcal{O}^{\mathsf{enc}}_{\mathsf{dore}}(\cdot,\cdot,\cdot)$: For a tuple (ID, m_0, m_1), the challenger runs $\mathsf{DORE.Enc}(m_b, sk_{\mathsf{ID}})$ to get c, and returns the ciphertext c to the adversary.

Let $\mathcal{L}$ be a leakage function, and the *open authority* for an identity set be defined the same way as in Section 3.2. To prevent the adversary from trivially winning the security

game, the above game $\mathbf{G}^{\mathsf{dore}}_{\mathsf{ind},\mathcal{A}}$ has some restrictions, which are analogously analyzed as for the restrictions in $\mathbf{G}^{\mathsf{DERE}}_{\mathsf{ind},\mathcal{A}}$, and are specified below.

Restriction-1: For each identity set $\mathbb{U}$ with open authority and the tuples $\{(\mathsf{ID}_1, m_{1,0}, m_{1,1}), \cdots, (\mathsf{ID}_q, m_{q_e,0}, m_{q_e,1})\}$ corresponding to the queries to encryption oracle $\mathcal{O}^{\mathsf{enc}}_{\mathsf{dore}}$, where $\mathsf{ID}_1, \cdots, \mathsf{ID}_q \in \mathbb{U}$, we have $\mathcal{L}(m_{1,0}, \cdots, m_{q,0}) = \mathcal{L}(m_{1,1}, \cdots, m_{q,1})$.

Restriction-2: If the adversary $\mathcal{A}$ has issued identity ID to private key generation oracle $\mathcal{O}^{\mathsf{sk}}_{\mathsf{dore}}$, i.e., $\mathcal{A}$ has obtained sk_{ID}, then we have

> Restriction-2a: $\mathcal{A}$ cannot query (ID, m_0, m_1) to the encryption oracle such that $m_0 \neq m_1$.
> Restriction-2b: $\mathcal{A}$ cannot query $(\mathsf{ID}_i, m_0, m_1)$ to the encryption oracle, where $m_0 \neq m_1$, $i \in [N]$ and $\mathsf{ID} \neq \mathsf{ID}_i$, if $t_{\mathsf{ID}_i,\mathsf{ID}}$ is obtained by $\mathcal{A}$ from token generation oracle.

Based on the above security game $\mathbf{G}^{\mathsf{dore}}_{\mathsf{ind},\mathcal{A}}$, we define the advantage of $\mathcal{A}$ in $\mathbf{G}^{\mathsf{dore}}_{\mathsf{ind},\mathcal{A}}$ as:

$$\mathrm{Adv}^{\mathsf{dore}}_{\mathsf{ind},\mathcal{A}}(\lambda) = |\Pr[b = b'] - \frac{1}{2}|.$$

Definition 4.2. A DORE scheme achieves the indistinguishability under authority delegation and leakage-consistent chosen plaintext attacks (IND-AD-LCPA) with respect to $\mathcal{L}$, if for any PPT adversary $\mathcal{A}$, its advantage $\mathrm{Adv}^{\mathsf{dore}}_{\mathsf{ind},\mathcal{A}}$ in game $\mathbf{G}^{\mathsf{dore}}_{\mathsf{ind},\mathcal{A}}$ is negligible.

The IND-AD-LCPA security is a generalization of the security definition of ORE. Specifically, if only one user is considered, the IND-AD-LCPA security captures the security definition of ORE (w.r.t. the same leakage function) that is introduced in [12]. Furthermore, if there is only one user and the leakage function is exactly the order, the IND-AD-LCPA is the IND-OCPA security introduced in [4] aiming at ideal security for OPE/ORE.

4.3 Construction of DORE

We first sketch the high-level idea of the DORE construction. Our DORE construction uses the DERE scheme as a key building tool, which provides the delegatable equality test functionality. With the delegatable functionality, we then design a framework in which the plain message is encoded such that delegatable order comparison on ciphertexts can be performed based upon the delegatable equality test. In more detail, the message is encoded bit-by-bit, and the ciphertext corresponding to each bit consists of a pair of DERE encodings, both of which encode the prefix of this bit and an identifier from $\{0, 1, 2\}$ where the choice of identifier depends on this bit. If the bit is 0, the identifiers of the two DERE encodings are 0 and 1, respectively; otherwise, the identifiers are 1 and 2, respectively. With the differential choice of identifiers, equality test on DERE encodings makes the ciphertext order comparison feasible w.r.t. the leakage $\mathcal{L}_{\mathsf{msdb}}$.

For presentation simplicity, we define the encoding $\mathcal{E}$ for a tuple (i, m, a) as:

$$\mathcal{E}(i, m, a) = (i, \mathsf{pref}(m, i-1) \| 0^{n-i}, a),$$

where n is the message length, $m \in \{0,1\}^n$, $i \in [n]$ and $a \in \{0, 1, 2\}$.

Given a DERE scheme, denoted by $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{TGen})$, the DORE construction is specified as follows:

- $\mathsf{DORE.KGen}(\lambda)$: On input a security parameter λ, it runs $\mathsf{KGen}(\lambda)$ to obtain a public/private key pair (pk, sk) of DERE scheme. It outputs (pk, sk) as the public/private key pair.
- $\mathsf{DORE.Enc}(m, sk)$: On input a message m and a private key sk, for each $i \in [n]$, it computes the ciphertext corresponding to the i-th bit of m, i.e., $m[i]$, it computes

  $$c_{i,0} = \mathsf{Enc}\Big(\mathcal{E}(i, m, 0), sk\Big), c_{i,1} = \mathsf{Enc}\Big(\mathcal{E}(i, m, 1), sk\Big),$$

  if $m[i] = 0$; otherwise, it computes

  $$c_{i,0} = \mathsf{Enc}\Big(\mathcal{E}(i, m, 1), sk\Big), c_{i,1} = \mathsf{Enc}\Big(\mathcal{E}(i, m, 2), sk\Big).$$

  Finally, the algorithm outputs $c = ((c_{1,0}, c_{1,1}), \cdots, (c_{n,0}, c_{n,1}))$ as the ciphertext corresponding to the message m.
- $\mathsf{DORE.TGen}(sk_{\mathsf{ID}_i}, pk_{\mathsf{ID}_j})$: On input ID_i's private key sk_{ID_i} and ID_j's public key pk_{ID_j}, where $i, j \in [N]$, the algorithm runs $\mathsf{TGen}(sk_{\mathsf{ID}_i}, pk_{\mathsf{ID}_j})$ to get $t_{\mathsf{ID}_i,\mathsf{ID}_j}$, and outputs $t_{\mathsf{ID}_i,\mathsf{ID}_j}$.[1]
- $\mathsf{DORE.Comp}(c, c', \vec{t})$: On input two ciphertexts c, c' and a token vector $\vec{t}$, it first parses

  $$c = \{(c_{1,0}, c_{1,1}), \cdots, (c_{n,0}, c_{n,1})\},$$
  $$c' = \{(c'_{1,0}, c'_{1,1}), \cdots, (c'_{n,0}, c'_{n,1})\}.$$

  For $(i = 1; i \leq n; i++)$:
  – If $\mathsf{Test}(c_{i,0}, c'_{i,1}, \vec{t}) = 1$, it returns 1 and stops.
  – Else if $\mathsf{Test}(c_{i,1}, c'_{i,0}, \vec{t}) = 1$, it returns 2 and stops.
  – Else it proceeds to the next loop.
  If $i = n + 1$, it returns 0.

The correctness of the above DORE scheme is direct from that of the underlying DERE. Obviously, the decryption can be done by utilizing encryption and comparison algorithms to do a binary search.

Efficiency. Since each encoding in DERE has 2 group elements in $\mathbb{G}_1$, each ciphertext of our DORE scheme has $4n$ group elements in $\mathbb{G}_1$. The encryption algorithm (resp., the token generation algorithm) needs to perform $6n$ (resp., 2) exponentiation computations. Comparing two ciphertexts performs $O(n)$ pairings. Nevertheless, the time-consuming pairing-based comparison algorithm is run on the server-side. The encryption algorithm on the client side only performs relatively efficient group operations. All exponentiations required for encryption are in $\mathbb{G}_1$, which could provide a higher efficiency for the client, since those exponentiations are cheaper than $\mathbb{G}_2$ or $\mathbb{G}_T$.

Fine-grained access control. In Cryptdb [26], OPE/ORE is combined with the onion structure to provide a fine-grained access control, where the OPE/ORE encryption is wrapped

[1] With this token, ID_i opens the order comparison authority of its own ciphertexts to ID_j's. Then, additionally with token $t_{\mathsf{ID}_j,\mathsf{ID}_i}$, the ciphertexts corresponding to ID_i and ID_j can be compared.

by an AES encryption level. To provide a fine-grained access control in multi-user environment, we may suggest to use parallel encryptions in the order level. Specifically, both OPE/ORE and DORE encryptions are implemented, which are then wrapped by AES encryption. Suppose two users A and B hope to search the entry corresponding to the maximum/minimum number of certain attribute in their databases. With the parallel encryption approach, the cloud server peels off the AES encryptions on top of OPE/ORE encryptions, and exploits the OPE/ORE encryptions to first find the entries corresponding the maximum/minimum number in A's database and those in B's database, respectively. Finally, the cloud server peels off the AES encryptions on top of the DORE encryptions *only for these resultant entries*. This way, the maximum/minimum search only requires one cross-database comparison by the cloud server, which also much mitigates the threat potentially caused by full-range comparisons between A's and B's databases.

4.4 Security Proof of DORE

We first review the following leakage function $\mathcal{L}_{msdb}$, which was originally introduced in [12]. We have

$$\mathcal{L}_{msdb}(m_1, \cdots, m_k) = \{\mathsf{msdb}(m_i, m_j) : 1 \le i < j \le k\},$$

where $\mathsf{msdb}(m_i, m_j) = (\mathsf{ind}_{\mathsf{msdb}}(m_i, m_j), \mathsf{order}(m_i, m_j))$, k and n are two positive integers, $m_i \in \{0,1\}^n$ for each $i \in [k]$, order denotes the order relationship of message m_i and m_j, and $\mathsf{ind}_{\mathsf{msdb}}(m_i, m_j)$ denotes the index of the most significant different bit of m_i and m_j. If $m_i = m_j$, we set $\mathsf{ind}_{\mathsf{msdb}}(m_i, m_j) = n + 1$. More specifically, for $m_i \neq m_j$, $\mathsf{ind}_{\mathsf{msdb}}(m_i, m_j)$ denotes the smallest index $\ell \in [n]$ such that $m_i[\ell] \neq m_j[\ell]$.

Theorem 4.3. *Assuming the underlying DERE is IND-AD-DCPA secure, the DORE scheme described above is IND-AD-LCPA secure w.r.t. the leakage function $\mathcal{L}_{msdb}$.*

Proof. The proof is based on the approach of game hopping, where a sequence of games are introduced. From the initial game (the real game) to the final one (the ideal game), we prove the indistinguishability between any pair of adjacent games. The proof is concluded by showing that the adversary's advantage in winning the final game is exactly 0.

Let $b \in \{0,1\}$ be the challenge bit picked randomly by the challenger. The sequence of games are defined as follows.

Game $\mathbf{G}_0$: This game is exactly the real one, namely $\mathbf{G}^{\mathsf{dore}}_{\mathsf{ind},\mathcal{A}}$.

The games $\mathbf{G}_1$, $\mathbf{G}_2$ and $\mathbf{G}_3$ are the same as $\mathbf{G}_0$ except the encryption generation of m_b for any tuple (ID, m_0, m_1) issued by $\mathcal{A}$, where $m_0 \neq m_1$ and denoting $i^* = \mathsf{msdb}(m_0, m_1)$.

Game $\mathbf{G}_1$: This game is the same as $\mathbf{G}_0$ except the ciphertext generation of $m[i^*]$, the challenger $\mathcal{C}$ computes

$$c_{i^*,0} = \mathsf{Enc}\Big(\mathcal{E}(i^*, m_b, 0), sk_{\mathsf{ID}}\Big), c_{i^*,1} = \mathsf{Enc}\Big(\mathcal{E}(i^*, m_b, 2), sk_{\mathsf{ID}}\Big),$$

and sets $(c_{i^*,0}, c_{i^*,1})$ as the ciphertext of $m_b[i^*]$. Note that, for $i \neq i^*$, the ciphertext of $m_b[i]$ is normally generated, namely as $\mathbf{G}_0$.

Game $\mathbf{G}_2$: This game is the same as $\mathbf{G}_1$ except the ciphertext generation of $m[i]$ where $i^* < i \le n$ and $m_0[i] \neq m_1[i]$, the challenger $\mathcal{C}$ works in the following way: $\mathcal{C}$ maintains a table $\mathbb{T}_1$, which is initially set to be empty and used to generate ciphertexts. If the entry $(\mathsf{pref}(m_0, i), \mathsf{pref}(m_1, i), Y_0', Y_1')$ already exists in table $\mathbb{T}_1$, then $\mathcal{C}$ sets $Y_0 = Y_0'$ and $Y_1 = Y_1'$; Otherwise, $\mathcal{C}$ chooses Y_0 and Y_1 uniformly at random from the message space of DERE, and adds $(\mathsf{pref}(m_0, i), \mathsf{pref}(m_1, i), Y_0, Y_1)$ into the table $\mathbb{T}_1$. Next, $\mathcal{C}$ computes

$$c_{i,0} = \mathsf{Enc}(Y_0, sk_{\mathsf{ID}}), \quad c_{i,1} = \mathsf{Enc}(Y_1, sk_{\mathsf{ID}}),$$

and sets $(c_{i,0}, c_{i,1})$ as the ciphertext of $m_b[i]$.

Game $\mathbf{G}_3$: This game is the same as $\mathbf{G}_2$ except the ciphertext generation of $m[i]$ where $i^* < i \le n$ and $m_0[i] = m_1[i]$, the challenger $\mathcal{C}$ works in the following way: $\mathcal{C}$ maintains a table $\mathbb{T}_2$ which is initially set to be empty. If the entry $(\mathsf{pref}(m_0, i-1), \mathsf{pref}(m_1, i-1), R_0, R_1, R_2)$ already exists in the table $\mathbb{T}_2$, then the challenger $\mathcal{C}$ gets R_0, R_1 and R_2; otherwise, $\mathcal{C}$ chooses R_0, R_1 and R_2 uniformly at random from the message space of DERE, and adds $(\mathsf{pref}(m_0, i-1), \mathsf{pref}(m_1, i-1), R_0, R_1, R_2)$ into the table $\mathbb{T}_2$. Next, $\mathcal{C}$ computes

$$c_{i,0} = \mathsf{Enc}(R_0, sk_{\mathsf{ID}}), \quad c_{i,1} = \mathsf{Enc}(R_1, sk_{\mathsf{ID}}),$$

if $m_b[i] = 0$; otherwise, it computes

$$c_{i,0} = \mathsf{Enc}(R_1, sk_{\mathsf{ID}}), \quad c_{i,1} = \mathsf{Enc}(R_2, sk_{\mathsf{ID}}).$$

$\mathcal{C}$ sets $(c_{i,0}, c_{i,1})$ as the ciphertext of $m_b[i]$.

Note that all the differences among the above games, namely $\mathbf{G}_0$, $\mathbf{G}_1$, $\mathbf{G}_2$ and $\mathbf{G}_3$, are exactly in the ciphertexts. For each challenge tuple (ID, m_0, m_1) where $m_0 \neq m_1$, let i^* denote the index of the most significant different bit of m_0 and m_1, i.e., $i^* = \mathsf{msdb}(m_0, m_1)$. Specifically, the ciphertexts of $m_b[i^*]$ are different between $\mathbf{G}_0$ and $\mathbf{G}_1$. The only difference between $\mathbf{G}_1$ and $\mathbf{G}_2$ is the ciphertext generation for $m_b[i]$, where $i > i^*$ and $m_0[i] \neq m_1[i]$. Finally, from $\mathbf{G}_2$ to the final game $\mathbf{G}_3$, we have a different strategy for the ciphertext generation of $m_b[i]$, where $i > i^*$ and $m_0[i] = m_1[i]$.

For presentation simplicity, we say two games are indistinguishable if the views of any PPT adversary $\mathcal{A}$ in these games are indistinguishable. To conclude the proof, we have the following four lemmas.

Lemma 4.4. *If the underlying DERE scheme is secure, game $\mathbf{G_0}$ and game $\mathbf{G_1}$ are indistinguishable for any PPT adversary $\mathcal{A}$.*

Lemma 4.5. *If the underlying DERE scheme is secure, game $\mathbf{G_1}$ and game $\mathbf{G_2}$ are indistinguishable for any PPT adversary $\mathcal{A}$.*

The detail proofs of Lemma 4.4 and 4.5 are presented in Appendix A.1 and A.2, respectively.

Lemma 4.6. *If the underlying DERE scheme is secure, game $\mathbf{G_2}$ and game $\mathbf{G_3}$ are indistinguishable for any PPT adversary $\mathcal{A}$.*

PROOF. For a PPT adversary $\mathcal{A}$ in $\mathbf{G}_2$ and $\mathbf{G}_3$, we build a PPT algorithm $\mathcal{D}$ which uses $\mathcal{A}$ as a subroutine to break the security of the underlying DERE scheme.

Description of $\mathcal{D}$. At the beginning of the challenge phase, $\mathcal{D}$ flips a random coin $b \leftarrow \{0,1\}$. Next, $\mathcal{D}$ simulates the oracles $\mathcal{O}^{\mathsf{tk}}_{\mathsf{dore}}$, $\mathcal{O}^{\mathsf{pk}}_{\mathsf{dore}}$ and $\mathcal{O}^{\mathsf{sk}}_{\mathsf{dore}}$ as in Section A.1. It maintains two tables $\mathbb{T}_1$ and $\mathbb{T}_2$ to record the information which will be used to generate the ciphertext. When $\mathcal{A}$ queries a tuple (ID, m_0, m_1) to the encryption oracle, it normally generates the ciphertext; otherwise, for each $i \in [n]$, it acts as follows (Let i^*= $\mathsf{msdb}(m_0, m_1)$):

- $1 \le i < i^*$: It normally generates the ciphertext of $m_b[i]$.
- $i = i^*$: It sets $y_0 = \mathcal{E}(i^*, m_b, 0)$ and $y_1 = \mathcal{E}(i^*, m_b, 2)$, and then issues (ID, y_0, y_0) and (ID, y_1, y_1) to $\mathcal{O}^{\mathsf{enc}}$ to get $c_{i^*,0}$ and $c_{i^*,1}$, respectively. It sets $(c_{i^*,0}, c_{i^*,1})$ as the ciphertext corresponding to $m_b[i^*]$.
- $i^* < i \le [n]$:
  - $m_0[i] \neq m_1[i]$: If there exists an entry $\big(\mathsf{pref}(m_0, i), \mathsf{pref}(m_1, i), Y_0, Y_1,\big)$ in $\mathbb{T}_1$, it chooses Y_0 and Y_1. Otherwise, it chooses Y_0 and Y_1 uniformly at random from the DERE message space, and adds $\big(\mathsf{pref}(m_0, i), \mathsf{pref}(m_1, i), Y_0, Y_1\big)$ into the table $\mathbb{T}_1$. It successively issues (ID, Y_0, Y_0) and (ID, Y_1, Y_1) to $\mathcal{O}^{\mathsf{enc}}$ to get $c_{i,0}$ and $c_{i,1}$, respectively. It sets $(c_{i,0}, c_{i,1})$ as the ciphertext of $m_b[i]$.
  - $m_0[i] = m_1[i]$: If there exists an item $\big(\mathsf{pref}(m_0, i-1), \mathsf{pref}(m_1, i-1), R_0, R_1, R_2\big)$ in $\mathbb{T}_2$, it chooses R_0, R_1 and R_2. Otherwise, it chooses R_0, R_1 and R_2 uniformly at random from the DERE message space, and adds $\big(\mathsf{pref}(m_0, i-1), \mathsf{pref}(m_1, i-1), R_0, R_1, R_2\big)$ into the table $\mathbb{T}_2$. it generates the ciphertext according to the following two cases:
    - $m_b[i] = 0$: It successively issues $(\mathsf{ID}, \mathcal{E}(i, m_b, 0), R_0)$ and $(\mathsf{ID}, \mathcal{E}(i, m_b, 1), R_1)$ to $\mathcal{O}^{\mathsf{enc}}$ to get $c_{i,0}$ and $c_{i,1}$, respectively.
    - $m_b[i] = 1$: It successively issues $(\mathsf{ID}, \mathcal{E}(i, m_b, 1), R_1)$ and $(\mathsf{ID}, \mathcal{E}(i, m_b, 2), R_2)$ to $\mathcal{O}^{\mathsf{enc}}$ to get $c_{i,0}$ and $c_{i,1}$, respectively.

The algorithm $\mathcal{D}$ returns $((c_{10}, c_{11}), \cdots, (c_{n0}, c_{n1}))$ as the ciphertext to $\mathcal{A}$'s query to the encryption oracle. Let $\overline{b}$ be the challenge bit picked by $\mathcal{C}$ in $\mathbf{G}^{\mathsf{DERE}}_{\mathsf{ind},\mathcal{D}}$. For the ciphertexts corresponding to $m_b[i]$, where $i^* < i \le n$ and $m_0[i] = m_0[i]$, we have the following observations:

- $\overline{b} = 0$: The ciphertext is $\big(\mathsf{Enc}\big(\mathcal{E}(i, m_b, 0), sk_{\mathsf{ID}}\big), \mathsf{Enc}\big(\mathcal{E}(i, m_b, 1), sk_{\mathsf{ID}}\big)\big)$ if $m_b[i]$=0, and $\big(\mathsf{Enc}\big(\mathcal{E}(i, m_b, 1), sk_{\mathsf{ID}}\big), \mathsf{Enc}\big(\mathcal{E}(i, m_b, 2), sk_{\mathsf{ID}}\big)\big)$ otherwise. Thus, $\mathcal{A}$'s view in this case is exactly identical to that in $\mathbf{G}_2$.
- $\overline{b} = 1$: The ciphertext is $\big(\mathsf{Enc}\big(R_0, sk_{\mathsf{ID}}\big), \mathsf{Enc}\big(R_1, sk_{\mathsf{ID}}\big)\big)$ if $m_b[i] = 0$, and $\big(\mathsf{Enc}\big(R_1, sk_{\mathsf{ID}}\big), \mathsf{Enc}\big(R_2, sk_{\mathsf{ID}}\big)\big)$ otherwise. In this case, $\mathcal{A}$'s view is exactly identical to that in $\mathbf{G}_3$.

We claim that $\mathcal{D}$ provides a successful simulation to $\mathcal{A}$. In other words, to simulate $\mathbf{G}_2$ or $\mathbf{G}_3$ to $\mathcal{A}$ which follows the restrictions in $\mathbf{G}^{\mathsf{dore}}_{\mathsf{ind},\mathcal{A}}$, $\mathcal{D}$ follows the restrictions in $\mathbf{G}^{\mathsf{DERE}}_{\mathsf{ind},\mathcal{D}}$. The analysis for this is analogous to that in the proof of Lemma 4.4, we omit it for simplicity.

Let $\mathcal{A}$'s advantage in winning $\mathbf{G}_2$ and $\mathbf{G}_3$ be $Adv^{\mathbf{G}_2}_{\mathcal{A}}$ and $Adv^{\mathbf{G}_3}_{\mathcal{A}}$, respectively. By analogous probability calculation as in Lemma 4.4, we have that $\mathcal{D}$'s advantage in winning $\mathbf{G}^{\mathsf{DERE}}_{\mathsf{ind},\mathcal{D}}$ is exactly $|Adv^{\mathbf{G}_2}_{\mathcal{A}} - Adv^{\mathbf{G}_3}_{\mathcal{A}}|/2$. This concludes the proof. □

LEMMA 4.7. *For any PPT adversary $\mathcal{A}$, it's advantage in game $\mathbf{G}_3$ is exactly 0.*

PROOF. For a tuple (ID, m_0, m_1) where $m_0 \neq m_1$, let $i^* = \mathsf{msdb}(m_0, m_1)$ and b be the challenge bit. For each i , $1 \le i < i^*$, the ciphertext corresponding to $m_b[i]$ is independent of b, because $m_0[i] = m_1[i]$ and the ciphertext of $m_b[i]$ is normally generated. The ciphertext corresponding to $m_b[i^*]$ is also independent of b, since the ciphertext of $m_b[i^*]$ is $\big(\mathsf{Enc}(\mathcal{E}(i^* - 1, m_b, 0), sk_{\mathsf{ID}}), \mathsf{Enc}(\mathcal{E}(i^* - 1, m_b, 2), sk_{\mathsf{ID}})\big)$ and $\mathsf{pref}(m_0, i^* - 1) = \mathsf{pref}(m_1, i^* - 1)$. The ciphertext corresponding to the i-th bit, for $i^* < i \le n$, depends on the random numbers with respect to the entry $(\mathsf{pref}(m_0, i), \mathsf{pref}(m_1, i), *, *)$ in $\mathbb{T}_1$ or $(\mathsf{pref}(m_0, i-1), \mathsf{pref}(m_1, i-1), *, *, *)$ in $\mathbb{T}_2$, which means that the -ciphertext of $m_b[i]$ is independent of the challenge bit b. Therefore, the view of $\mathcal{A}$ is completely independent of the challenge bit b, and consequently the advantage of $\mathcal{A}$ in game $\mathbf{G}_3$ is exactly 0. □

□

5 CONCLUSION AND FUTURE WORK

In the data era we are experiencing, cross-database transactions become ubiquitous in practice. Meanwhile, we need to protect the confidentiality of data for these transactions in more and more environments, e.g., military, medical or commercial application scenarios. For cross-database transactions, order-based comparison for data from different entities is obviously a basic operation. Our DORE could be viewed as a first step aiming for an efficient solution for such a task. Further improving the efficiency of DORE, and extending its functionalities, are promising future directions that have practical significance.

ACKNOWLEDGMENT

This work is supported in part by National Key Research and Development Program of China under Grant No. 2017YFB0802000, National Natural Science Foundation of China under Grant Nos. 61472084 and U1536205, Shanghai Innovation Action Project under Grant No. 16DZ1100200, Shanghai Science and Technology Development Funds under Grant No. 16JC1400801, and Shandong Provincial Key Research and Development Program of China under Grant Nos. 2017CXG0701 and 2018CXGC0701.

REFERENCES

[1] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, H. Shi, Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions, in: Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings, 2005, pp. 205–222.

[2] M. Bellare, M. Fischlin, A. O'Neill, T. Ristenpart, Deterministic encryption: Definitional equivalences and constructions without random oracles, in: Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings, 2008, pp. 360–378.

[3] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in: 2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA, 2007, pp. 321–334.

[4] A. Boldyreva, N. Chenette, Y. Lee, A. O'Neill, Order-preserving symmetric encryption, in: Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings, 2009, pp. 224–241.

[5] A. Boldyreva, N. Chenette, A. O'Neill, Order-preserving encryption revisited: Improved security analysis and alternative solutions, in: Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings, 2011, pp. 578–595.

[6] D. Boneh, X. Boyen, E. Goh, Hierarchical identity based encryption with constant size ciphertext, in: Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings, 2005, pp. 440–456.

[7] D. Boneh, G. D. Crescenzo, R. Ostrovsky, G. Persiano, Public key encryption with keyword search, in: Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings, 2004, pp. 506–522.

[8] D. Boneh, K. Lewi, M. Raykova, A. Sahai, M. Zhandry, J. Zimmerman, Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation, in: Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II, 2015, pp. 563–594.

[9] N. Cao, C. Wang, M. Li, K. Ren, W. J. Lou, Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data, in: INFOCOM, 2011, pp. 829–837.

[10] D. Cash, F. H. Liu, A. O'Neill, M. Zhandry, C. Zhang, Parameter-hiding order revealing encryption, in: https://eprint.iacr.org/2018/698.pdf, 2018.

[11] D. Cash, F. Liu, C. Zhang, Reducing the leakage in practical order-revealing encryption, in: http://eprint.iacr.org/2016/661.pdf, 2016.

[12] N. Chenette, K. Lewi, S. A. Weis, D. J. Wu, Practical order-revealing encryption with limited leakage, in: Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers, 2016, pp. 474–493.

[13] F. B. Durak, T. M. DuBuisson, D. Cash, What else is revealed by order-revealing encryption?, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016, 2016, pp. 1155–1166.

[14] J. Eom, D. H. Lee, K. S. Lee, Multi-Client Order-Revealing Encryption, in: IEEE Access, Volume: 6, 2018, pp. 45458 - 45472.

[15] Z. J. Fu , K. Ren, J. G. Shu, X. M. Sun, F. X. Huang, Enabling Personalized Search over Encrypted Outsourced Data with Efficiency Improvement, in: IEEE Transactions on Parallel and Distributed Systems, Volume: 27 , Issue: 9 , Sept., 2016, pp. 2546–2559.

[16] Z. J. Fu, X. L. Wu, Q. Wang, K. Ren, Enabling Central Keyword-Based Semantic Extension Search Over Encrypted Outsourced Data, in: IEEE Transactions on Information Forensics and Security, Volume: 12 , Issue: 12 , Dec., 2017, pp. 2986–2997.

[17] O. Goldreich, Foundations of cryptogrpahy, volume ii: Basic applications, 2004.

[18] P. Grubbs, K. Sekniqi, V. Bindschaedler, M. Naveed, T. Ristenpart, Leakage-abuse attacks against order-revealing encryption, in: 2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017, 2017, pp. 655–672.

[19] F. Kerschbaum, A. Schröpfer, Optimal average-complexity ideal-security order-preserving encryption, in: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014, 2014, pp. 275–286.

[20] S. Q. Lai, S. Patranabis, A. Sakzad, J. K. Liu, D. Mukhopadhyay, R. Steinfeld, S. F. Sun, D. X. Liu, C. Zuo, Result Pattern Hiding Searchable Encryption for Conjunctive Queries, in: ACM Conference on Computer and Communications Security, 2018, pp. 745-762.

[21] K. Lewi, D. J. Wu, Order-revealing encryption: New constructions, applications, and lower bounds, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016, 2016, pp. 1167–1178.

[22] K. T. Liang, X. Y. Huang, F. C. Guo, J. K. Liu, Privacy-Preserving and Regular Language Search Over Encrypted Cloud Data, in: IEEE Transactions on Information Forensics and Security, 11(10), 2016, pp. 2365-2376.

[23] J. H. Liu, J. H. Ma, W. L. Zhou, Y. Xiang, X. Y. Huang, Dissemination of Authenticated Tree-Structured Data with Privacy Protection and Fine-Grained Control in Outsourced Databases, in: ESORICS (2), 2018, pp. 167–186.

[24] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in: Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding, 1999, pp. 223–238.

[25] R. A. Popa, F. H. Liu, N. Zeldovich, An ideal-security protocol for order-preserving encoding, in: 2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013, 2013, pp. 463–477.

[26] R. A. Popa, C. M. S. Redfield, N. Zeldovich, H. Balakrishnan, Cryptdb: protecting confidentiality with encrypted query processing, in: Proceedings of the 23rd ACM Symposium on Operating Systems Principles 2011, SOSP 2011, Cascais, Portugal, October 23-26, 2011, 2011, pp. 85–100.

[27] S. C. Ramanna, S. Chatterjee, P. Sarkar, Variants of waters' dual system primitives using asymmetric pairings - (extended abstract), in: Public Key Cryptography - PKC 2012 - 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012. Proceedings, 2012, pp. 298–315.

[28] D. S. Roche, D. Apon, S. G. Choi, A. Yerukhimovich, POPE: partial order preserving encoding, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016, 2016, pp. 1131–1142.

[29] C. Schnorr, Security of blind discrete log signatures against interactive attacks, in: Information and Communications Security, Third International Conference, ICICS 2001, Xian, China, November 13-16, 2001, 2001, pp. 1–12.

[30] C. Schnorr, M. Jakobsson, Security of signed elgamal encryption, in: Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings, 2000, pp. 73–89.

[31] V. Shoup, Lower bounds for discrete logarithms and related problems, in: Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding, 1997, pp. 256–266.

[32] D. X. Song, D. Wagner, A. Perrig, Practical techniques for searches on encrypted data, in: Security and Privacy, 2000. Proceedings. 2000 IEEE Symposium on, 2002, p. 0044.

[33] S. F. Sun, X. L. Yuan, J. K. Liu, R. Steinfeld, A. Sakzad, V. Vo, S. Nepal, Practical Backward-Secure Searchable Encryption from Symmetric Puncturable Encryption, in: ACM Conference on Computer and Communications Security, 2018, pp. 763-780.

[34] C. Wang, N. Cao, K. Ren, W. J. Lou, Enabling Secure and Efficient Ranked Keyword Search over Outsourced Cloud Data, in: IEEE Trans. Parallel Distrib. Syst., 23(8), 2012, pp. 1467–1479.

[35] X. C. Wang, Y. L. Zhao, Order-Revealing Encryption: File-Injection Attack and Forward Security, in: ESORICS, 2018, pp. 101–121.

[36] L. L. Xiao, I. L. Ren, D. T. Huynh, Extending Order Preserving Encryption for Multi-User Systems, https://eprint.iacr.org/2012/192.pdf.

[37] J. Zhu, Q. Li, C. Wang, X. L. Yuan, Q. Wang, K. Ren, Enabling Generic, Verifiable, and Secure Data Search in Cloud Services, in: IEEE Trans. Parallel Distrib. Syst. 29(8), 2018, pp. 1721-1735.

A PROOFS OF LEMMAS 4.4 AND 4.5

A.1 Proof of Lemma 4.4

Proof. For a PPT adversary $\mathcal{A}$ to distinguish $\mathbf{G}_0$ and $\mathbf{G}_1$, we build a PPT algorithm $\mathcal{D}$ which uses $\mathcal{A}$ as a subroutine to break the security of DERE scheme. The construction of $\mathcal{D}$ is described as follows.

Description of $\mathcal{D}$. It starts with receiving all the global parameters from the challenger $\mathcal{C}$ in game $\mathbf{G}^{\mathsf{DERE}}_{\mathsf{ind},\mathcal{D}}$, and sends them as the global parameters for the DORE scheme to the adversary $\mathcal{A}$. At the beginning of the challenge phase, it flips a random coin $b \leftarrow \{0,1\}$. Next, it simulates the token generation oracle $\mathcal{O}^{\mathsf{tk}}_{\mathsf{dore}}$, the public key generation oracle $\mathcal{O}^{\mathsf{pk}}_{\mathsf{dore}}$ and the private key generation oracle $\mathcal{O}^{\mathsf{sk}}_{\mathsf{dore}}$ by using oracles $\mathcal{O}^{\mathsf{tk}}$, $\mathcal{O}^{\mathsf{pk}}$ and $\mathcal{O}^{\mathsf{sk}}$, respectively, where $\mathcal{O}^{\mathsf{tk}}$, $\mathcal{O}^{\mathsf{pk}}$ and $\mathcal{O}^{\mathsf{sk}}$ are oracles provided in the DERE game $\mathbf{G}^{\mathsf{DERE}}_{\mathsf{ind},\mathcal{D}}$. When $\mathcal{A}$ queries a tuple (ID, m_0, m_1) to the encryption oracle, $\mathcal{D}$ generates the ciphertext as follows:

- $m_0 = m_1$: For each bit $i \in [n]$, it normally generates the ciphertext of $m_b[i]$ by using oracle $\mathcal{O}^{\mathsf{enc}}$. In other words, it sets $y_0 = \mathcal{E}(i, m, 0)$ and $y_1 = \mathcal{E}(i, m, 1)$ if $m[i] = 0$; otherwise, it sets $y_0 = \mathcal{E}(i, m, 1)$ and $y_1 = \mathcal{E}(i, m, 2)$. Then, it successively issues the tuples (ID, y_0, y_0) and (ID, y_1, y_1) to $\mathcal{C}$ as the queries to the encoding oracle $\mathcal{O}^{\mathsf{enc}}$, and respectively gets two encodings $c_{i,0}$ and $c_{i,1}$. It sets $(c_{i,0}, c_{i,1})$ as the ciphertext corresponding to $m_b[i]$. Finally, it returns $((c_{1,0}, c_{1,1}), \cdots, (c_{n,0}, c_{n,1}))$ to $\mathcal{A}$.
- $m_0 \neq m_1$: Let $i^* = \mathsf{msdb}(m_0, m_1)$. For each $i \neq i^*$ and $i \in [n]$, it normally generates the ciphertext of $m_b[i]$. For $i = i^*$, it generates the ciphertext of $m_b[i^*]$ in the following two cases:
  - $m_b[i^*] = 0$: It sets $y_0 = \mathcal{E}(i^*, m_b, 0)$ and issues (ID, y_0, y_0) to $\mathcal{C}$ to get the $c_{i^*,0}$. Next, it sets $y_{10} = \mathcal{E}(i^*, m_b, 1)$ and $y_{11} = \mathcal{E}(i^*, m_b, 2)$, and sends $(\mathsf{ID}, y_{10}, y_{11})$ to $\mathcal{C}$ to get $c_{i^*,1}$.
  - $m_b[i^*] = 1$: It sets $y_1 = \mathcal{E}(i^*, m_b, 2)$ and issues (ID, y_1, y_1) to $\mathcal{C}$ to get the $c_{i^*,1}$. Next, it sets $y_{00} = \mathcal{E}(i^*, m_b, 1)$ and $y_{01} = \mathcal{E}(i^*, m_b, 0)$, and sends $(\mathsf{ID}, y_{00}, y_{01})$ to $\mathcal{C}$ to get $c_{i^*,0}$.

  It sets $(c_{i^*,0}, c_{i^*,1})$ as the ciphertext of $m_b[i^*]$.

Finally, the algorithm $\mathcal{D}$ returns $((c_{1,0}, c_{1,1}), \cdots, (c_{n,0}, c_{n,1}))$ to $\mathcal{A}$.

When the challenge phase is over, $\mathcal{A}$ finally outputs a guess bit $b' \in \{0,1\}$. If $b' = b$, $\mathcal{D}$ outputs 0 as its guess for the challenge bit in $\mathbf{G}^{\mathsf{DERE}}_{\mathsf{ind},\mathcal{D}}$, and 1 otherwise.

Let $\bar{b}$ be the challenge bit that is randomly picked by $\mathcal{C}$ in $\mathbf{G}^{\mathsf{DERE}}_{\mathsf{ind},\mathcal{D}}$. Next, for each tuple (ID, m_0, m_1) issued by $\mathcal{A}$ to the encryption oracle where $m_0 \neq m_1$, we analyze its ciphertext. In fact, we only need to focus on the ciphertext corresponding to $m_b[i^*]$, as only the simulation strategy on this component is different in the construction of $\mathcal{D}$.

- $\bar{b} = 0$: The ciphertext corresponding to $m_b[i^*]$ is $(\mathsf{Enc}(\mathcal{E}(i, m_b, 0), sk_{\mathsf{ID}}), \mathsf{Enc}(\mathcal{E}(i, m_b, 1), sk_{\mathsf{ID}}))$ if $m_b[i^*] = 0$, and $(\mathsf{Enc}(\mathcal{E}(i, m_b, 1), sk_{\mathsf{ID}}), \mathsf{Enc}(\mathcal{E}(i, m_b, 2), sk_{\mathsf{ID}}))$ otherwise. In this case, $\mathcal{A}$'s view is exactly identical to that in $\mathbf{G}_0$.
- $\bar{b} = 1$: The ciphertext corresponding to $m_b[i^*]$ is exactly $(\mathsf{Enc}(\mathcal{E}(i, m_b, 0), sk_{\mathsf{ID}}), \mathsf{Enc}(\mathcal{E}(i, m_b, 2), sk_{\mathsf{ID}}))$. Therefore, $\mathcal{A}$'s view is exactly identical to that in $\mathbf{G}_1$.

In order to successfully simulate $\mathbf{G}_0$ or $\mathbf{G}_1$ to $\mathcal{A}$, we claim that all the queries from $\mathcal{D}$ strictly follow the restrictions in $\mathbf{G}^{\mathsf{DERE}}_{\mathsf{ind},\mathcal{D}}$. In other words, once $\mathcal{A}$ follows the restrictions in $\mathbf{G}^{\mathsf{dore}}_{\mathsf{ind},\mathcal{A}}$, then $\mathcal{D}$'s encryption queries to $\mathcal{C}$ do not violate the restrictions in $\mathbf{G}^{\mathsf{DERE}}_{\mathsf{ind},\mathcal{D}}$. More concretely, we analyze it in the cases of Restriction-1 and Restriction-2.

The analysis with respect to Restriction-2 is simple, for which we have the following two cases:

- If $\mathcal{D}$'s queries violate Restriction-2a of DERE, then $\mathcal{D}$ has issued ID and (ID, x, y) to $\mathcal{O}^{\mathsf{sk}}$ and $\mathcal{O}^{\mathsf{enc}}$, respectively, where $x \neq y$. It implies that $\mathcal{A}$ has issued ID and (ID, m_0, m_1) to $\mathcal{O}^{\mathsf{sk}}_{\mathsf{dore}}$ and $\mathcal{O}^{\mathsf{enc}}_{\mathsf{dore}}$, respectively, where $m_0 \neq m_1$. Therefore, $\mathcal{A}$'s queries violate Restriction-2a of DORE.
- If $\mathcal{D}$'s queries violate Restriction-2b of DERE, then $\mathcal{D}$ has issued ID, (ID_j, x, y) and $(\mathsf{ID}_j, \mathsf{ID})$ to $\mathcal{O}^{\mathsf{sk}}$, $\mathcal{O}^{\mathsf{enc}}$ and $\mathcal{O}^{\mathsf{tk}}$, respectively, where $x \neq y$. It implies that $\mathcal{A}$ has issued ID, $(\mathsf{ID}_j, m_0, m_1)$ and $(\mathsf{ID}_j, \mathsf{ID})$ to $\mathcal{O}^{\mathsf{sk}}_{\mathsf{dore}}$, $\mathcal{O}^{\mathsf{enc}}_{\mathsf{dore}}$ and $\mathcal{O}^{\mathsf{tk}}_{\mathsf{dore}}$, respectively, where $m_0 \neq m_1$. Therefore, $\mathcal{A}$'s queries violate Restriction-2b of DORE.

For the analysis with respect to Restriction-1, we first define two events Event_1 and Event_2 as follows:

- Event_1: $\mathcal{D}$ issues $t_{\mathsf{ID}_\ell,*}$, $(\mathsf{ID}_\ell, x_0, y_0)$ and $(\mathsf{ID}_\ell, x_1, y_1)$ to $\mathcal{C}$.
- Event_2: $\mathcal{D}$ issues $t_{\mathsf{ID}_\ell,\mathsf{ID}_{\ell'}}$, $t_{\mathsf{ID}_{\ell'},\mathsf{ID}_\ell}$, $(\mathsf{ID}_\ell, x_0, y_0)$ and $(\mathsf{ID}_{\ell'}, x_1, y_1)$ to $\mathcal{C}$.

where $\ell, \ell' \in [N]$, $\ell \neq \ell'$, x_0, y_0, x_1, y_1 are chosen from the message space of DERE, $x_0 \neq y_0$ and $\mathcal{L}_{\mathsf{d}}(x_0, x_1) \neq \mathcal{L}_{\mathsf{d}}(y_0, y_1)$.

The possible violations of Restriction-1 of DERE only come from the occurrence of either Event_1 or Event_2. Next, we argue that both of these two events would not happen in the interaction between $\mathcal{D}$ and $\mathcal{C}$.

For each tuple $(\mathsf{ID}_\ell, m_0, m_1)$ issued by $\mathcal{A}$, where $\ell \in [N]$ and $m_0 \neq m_1$, the ciphertext generation corresponding to $m_b[i^*]$ should be focused, as only this procedure requires $\mathcal{D}$ to issues tuple (ID, x, y) to $\mathcal{O}^{\mathsf{enc}}$, where $x \neq y$. We only discuss the case that $m_b[i^*] = 0$, since the discussion for the case that $m_b[i^*] = 1$ is analogous. For the case that $m_b[i^*] = 0$, $\mathcal{D}$ should issue the tuples $(\mathsf{ID}_\ell, \mathcal{E}(i^*, m_b, 0), \mathcal{E}(i^*, m_b, 0))$ and $(\mathsf{ID}_\ell, \mathcal{E}(i^*, m_b, 1), \mathcal{E}(i^*, m_b, 2))$ to the encoding oracle $\mathcal{O}^{\mathsf{enc}}$ to obtain the DERE encodings for simulating the ciphertext of $m_b[i^*]$.

If the token $t_{\mathsf{ID}_\ell,*}$ is not queried by $\mathcal{A}$, then the above two events won't happen. If $t_{\mathsf{ID}_\ell,*}$ is queried, only the following three situations may probably cause the occurrence of Event_1:

- Cond-11: $\mathcal{D}$ queries the tuple $(\mathsf{ID}_\ell, \mathcal{E}(i^*, m_b, 1), \mathcal{E}(i^*, m_b, 1))$ or $(\mathsf{ID}_\ell, \mathcal{E}(i^*, m_b, 2), \mathcal{E}(i^*, m_b, 2))$ to $\mathcal{O}^{\mathsf{enc}}$. It implies that $\mathcal{A}$ should issue the tuple $(\mathsf{ID}, m'_0, m'_1)$ to $\mathcal{O}^{\mathsf{enc}}_{\mathsf{dore}}$ such that $\mathsf{pref}(m'_0, i^* - 1) = \mathsf{pref}(m'_1, i^* - 1) =$

$\mathsf{pref}(m_b, i^*-1)$, where $m'_0[i^*] = m'_1[i^*]$ or $m'_b[i^*] = 1, m'_{1-b}[i^*] = 0$. We have $\mathsf{msdb}(m_0, m'_0) \neq \mathsf{msdb}(m_1, m'_1)$ or $m_b < m'_b, m_{1-b} > m'_{1-b}$, which causes a violation of Restriction-1 of DORE. Therefore, this situation does not happen.

- Cond-12: $\mathcal{D}$ queries the tuple $(\mathsf{ID}_\ell, \mathcal{E}(i^*, m_b, 1), y)$ to $\mathcal{O}^{\mathsf{enc}}$, where $y \neq \mathcal{E}(i^*, m_b, 1)$ and $y \neq \mathcal{E}(i^*, m_b, 2)$. This encoding query issued by $\mathcal{D}$ is to simulate the ciphertext of the i^*-th bit corresponding to the tuple $(\mathsf{ID}, m'_0, m'_1)$ such that $\mathsf{pref}(m'_0, i^*-1) = \mathsf{pref}(m'_1, i^*-1) = \mathsf{pref}(m_b, i^*-1)$, where $m'_b[i^*] = 1$ and $m'_{1-b}[i^*] = 0$. This indicates that $m_b < m'_b$ and $m_{1-b} > m'_{1-b}$, which causes a violation of Restriction-1 of DORE. Therefore, this situation does not happen.
- Cond-13: $\mathcal{D}$ queries the tuple $(\mathsf{ID}_\ell, x, \mathcal{E}(i^*, m_b, 2))$ to $\mathcal{O}^{\mathsf{enc}}$, where $x \neq \mathcal{E}(i^*, m_b, 1)$ and $x \neq \mathcal{E}(i^*, m_b, 2)$. By the analogous analysis for Cond-12, we have that $\mathcal{A}$'s queries violate the Restriction-1 of DORE. Therefore, this situation also does not happen.

Consequently, the event Event_1 won't happen in this case.

If $t_{\mathsf{ID}_\ell, \mathsf{ID}_{\ell'}}$ and $t_{\mathsf{ID}_{\ell'}, \mathsf{ID}_\ell}$ are queried, where $\ell' \in [N]$ and $\ell \neq \ell'$, we only consider the occurrence of Event_2 since that of Event_1 has been discussed above. Only the following three situations may probably cause the occurrence of Event_2:

- Cond-21: $\mathcal{D}$ queries the tuple $(\mathsf{ID}_{\ell'}, \mathcal{E}(i^*, m_b, 1), \mathcal{E}(i^*, m_b, 1))$ or $(\mathsf{ID}_j, \mathcal{E}(i^*, m_b, 2), \mathcal{E}(i^*, m_b, 2))$ to $\mathcal{O}^{\mathsf{enc}}$.
- Cond-22: $\mathcal{D}$ queries the tuple $(\mathsf{ID}_{\ell'}, \mathcal{E}(i^*, m_b, 1), y)$ to $\mathcal{O}^{\mathsf{enc}}$, where $y \neq \mathcal{E}(i^*, m_b, 1)$ and $y \neq \mathcal{E}(i^*, m_b, 2)$.
- Cond-23: $\mathcal{D}$ queries the tuple $(\mathsf{ID}_{\ell'}, x, \mathcal{E}(i^*, m_b, 2))$ to $\mathcal{O}^{\mathsf{enc}}$, where $x \neq \mathcal{E}(i^*, m_b, 2)$ and $x \neq \mathcal{E}(i^*, m_b, 2)$.

Analogous to the discussions for Cond-11, Cond-12 and Cond-13, we have that Cond-21, Cond-22 and Cond-23 would not happen. Thus, the event Event_2 would not happen in this case.

Therefore, $\mathcal{D}$ provides a successful simulation.

Let $\mathcal{A}$'s advantage in winning $\mathbf{G}_0$ and $\mathbf{G}_1$ be $Adv_{\mathcal{A}}^{\mathbf{G}_0}$ and $Adv_{\mathcal{A}}^{\mathbf{G}_1}$, respectively. We have $Adv_{\mathcal{A}}^{\mathbf{G}_0} \geq Adv_{\mathcal{A}}^{\mathbf{G}_1}$ since the ciphertext corresponding to $m_b[i^*]$ for each ciphertext is actually independent from the bit b in $\mathbf{G}_1$, which may probably cause a degradation for $\mathcal{A}$'s advantage from the perspective of information theory. Let $Adv_{\mathcal{A}}^{\mathbf{G}_0} - Adv_{\mathcal{A}}^{\mathbf{G}_1} = \varepsilon$. We calculate $\Pr[b = \overline{b}]$ as follow

$$\begin{aligned}
&\Pr[b' = b|\overline{b} = 0] \cdot \Pr[\overline{b} = 0] + \Pr[b' \neq b|\overline{b} = 1] \cdot \Pr[\overline{b} = 1] \\
=&\Pr[b' = b|\overline{b} = 0] \cdot \Pr[\overline{b} = 0] + (1 - \Pr[b' = b|\overline{b} = 1]) \cdot \Pr[\overline{b} = 1] \\
=&(\frac{1}{2} + Adv_{\mathcal{A}}^{\mathbf{G}_0}) \cdot \frac{1}{2} + (1 - (\frac{1}{2} + Adv_{\mathcal{A}}^{\mathbf{G}_1})) \cdot \frac{1}{2} \\
=&\frac{1}{2} + \frac{\varepsilon}{2}
\end{aligned}$$

By using $\mathcal{A}$, we construct an algorithm $\mathcal{D}$ that breaks the security of DERE with advantage $\frac{\varepsilon}{2}$.

□

A.2 Proof of Lemma 4.5

Proof. For a PPT adversary $\mathcal{A}$ in $\mathbf{G}_1$ and $\mathbf{G}_2$, we build a PPT algorithm $\mathcal{D}$ which uses $\mathcal{A}$ as a subroutine to break the security of the underlying DERE scheme.

Description of $\mathcal{D}$. At the beginning of the challenge phase, it flips a random coin $b \leftarrow \{0, 1\}$. Next, it simulates the oracles $\mathcal{O}^{\mathsf{tk}}_{\mathsf{dore}}$, $\mathcal{O}^{\mathsf{pk}}_{\mathsf{dore}}$ and $\mathcal{O}^{\mathsf{sk}}_{\mathsf{dore}}$ as in Section A.1. It maintains a table $\mathbb{T}_1$ to record the information which will be used to generate the ciphertext. When $\mathcal{A}$ queries a tuple (ID, m_0, m_1) to the encryption oracle, it normally generates the ciphertext by issuing queries to the encoding oracle $\mathcal{O}^{\mathsf{enc}}$ if $m_0 = m_1$; otherwise, for each $i \in [n]$, it acts as follows (Let $i^* = \mathsf{msdb}(m_0, m_1)$):

- $1 \leq i < i^*$: It normally generates the ciphertext of $m_b[i]$.
- $i = i^*$: It sets $y_0 = \mathcal{E}(i^*, m_b, 0)$ and $y_1 = \mathcal{E}(i^*, m_b, 2)$, and then issues (ID, y_0, y_0) and (ID, y_1, y_1) to $\mathcal{O}^{\mathsf{enc}}$ to get $c_{i^*,0}$ and $c_{i^*,1}$, respectively. It sets $(c_{i^*,0}, c_{i^*,1})$ as the ciphertext of $m_b[i^*]$.
- $i^* < i \leq [n]$:
  - $m_0[i] = m_1[i]$: It normally generates the ciphertext of $m_b[i]$.
  - $m_0[i] \neq m_1[i]$: If there exists an entry $\big(\mathsf{pref}(m_0, i), \mathsf{pref}(m_1, i), Y_0, Y_1\big)$ in $\mathbb{T}_1$, it obtains Y_0 and Y_1. Otherwise, it chooses Y_0 and Y_1 uniformly at random from the DERE message space, and adds $\big(\mathsf{pref}(m_0, i), \mathsf{pref}(m_1, i), Y_0, Y_1,\big)$ into the table $\mathbb{T}_1$.
    - If $m_b[i] = 0$, it sets $y_0 = \mathcal{E}(i, m_b, 0)$ and $y_1 = \mathcal{E}(i, m_b, 1)$, and then issues the tuples (ID, y_0, Y_0) and (ID, y_1, Y_1) to $\mathcal{O}^{\mathsf{enc}}$ to obtain $c_{i,0}$ and $c_{i,1}$, respectively.
    - Otherwise, it sets $y_0 = \mathcal{E}(i, m_b, 1)$ and $y_1 = \mathcal{E}(i, m_b, 2)$, and then issues the tuples (ID, y_0, Y_0) and (ID, y_1, Y_1) to $\mathcal{O}^{\mathsf{enc}}$ to obtain $c_{i,0}$ and $c_{i,1}$, respectively.

The algorithm $\mathcal{D}$ returns $((c_{1,0}, c_{1,1}), \cdots, (c_{n,0}, c_{n,1}))$ as the ciphertext to $\mathcal{A}$'s encryption query.

Let $\overline{b}$ be the challenge bit picked by $\mathcal{C}$ in $\mathbf{G}^{\mathsf{DERE}}_{\mathsf{ind},\mathcal{D}}$. For the ciphertexts corresponding to $m_b[i]$, where $i^* < i \leq n$ and $m_0[i] \neq m_0[i]$, we have the following observations:

- $\overline{b} = 0$: The ciphertext is $\big(\mathsf{Enc}\big(\mathcal{E}(i, m_b, 0), sk_{\mathsf{ID}}\big), \mathsf{Enc}\big(\mathcal{E}(i, m_b, 1), sk_{\mathsf{ID}}\big)\big)$ if $m_b[i] = 0$, and $\big(\mathsf{Enc}\big(\mathcal{E}(i, m_b, 1), sk_{\mathsf{ID}}\big), \mathsf{Enc}\big(\mathcal{E}(i, m_b, 2), sk_{\mathsf{ID}}\big)\big)$ otherwise. In this case, $\mathcal{A}$'s view is exactly identical to that in $\mathbf{G}_1$.
- $\overline{b} = 1$: The ciphertext is $\big(\mathsf{Enc}\big(Y_0, sk_{\mathsf{ID}}\big), \mathsf{Enc}\big(Y_1, sk_{\mathsf{ID}}\big)\big)$. In this case, $\mathcal{A}$'s view is exactly identical to that in $\mathbf{G}_2$.

We claim that $\mathcal{D}$ successfully simulates game $\mathbf{G}_1$ or $\mathbf{G}_2$ to $\mathcal{A}$. The successful simulation implies that $\mathcal{D}$ does not violate the restrictions in $\mathbf{G}^{\mathsf{DERE}}_{\mathsf{ind},\mathcal{D}}$. he analysis for this is analogous to that in the proof of Lemma 4.4, we omit it for simplicity. Consequently, to successfully simulate $\mathbb{G}_1$ or $\mathbb{G}_2$, $\mathcal{D}$ does not violate the restrictions in the DERE security game.

Let $\mathcal{A}$'s advantages in winning $\mathbf{G}_1$ and $\mathbf{G}_2$ be $Adv_{\mathcal{A}}^{\mathbf{G}_1}$ and $Adv_{\mathcal{A}}^{\mathbf{G}_2}$, respectively. By analogous probability calculation as in Lemma 4.4, we have that $\mathcal{D}$'s advantage in winning $\mathbf{G}^{\mathsf{DERE}}_{\mathsf{ind},\mathcal{D}}$ is exactly $|Adv_{\mathcal{A}}^{\mathbf{G}_1} - Adv_{\mathcal{A}}^{\mathbf{G}_2}|/2$. This completes the proof. □