3-2017

Probabilistic public key encryption for controlled equijoin in relational databases

Yujue WANG
Singapore Management University, yjwang@smu.edu.sg

Hwee Hwa PANG
Singapore Management University, hhpang@smu.edu.sg

Probabilistic Public Key Encryption for Controlled Equijoin in Relational Databases

Yujue Wang and HweeHwa Pang

School of Information Systems, Singapore Management University,
80 Stamford Road, Singapore
Email: {yjwang,hhpang}@smu.edu.sg

We present a public key encryption scheme for relational databases (PKDE) that allows the owner to control the execution of cross-relation joins on an outsourced server. The scheme allows anyone to deposit encrypted records in a database on the server. Thereafter, the database owner may authorize the server to join any two relations to identify matching records across them, while preventing self-joins that would reveal information on records that are unmatched in the join. The security of our construction is formally proved in the random oracle model based on the computational bilinear Diffie-Hellman assumption. Specifically, before a relation is joined, its encrypted records enjoy IND-CCA2 security; after a join, our scheme offers One-Way CCA2 security protection on the records. Our PKDE construction is shown to outperform the only existing work, both in security guarantee and in efficiency.

Keywords: Database security; Data encryption; Controlled join; Equality test; Private set intersection; Data outsourcing

Received 22 April 2016; revised 20 June 2016

1. INTRODUCTION

Data outsourcing is gaining in adoption as it relieves users of the burden of storage and database management. The drawback is that users may lose control of their data, which gives rise to concerns whether confidentiality of the data will be safeguarded according to security provisions. To mitigate these concerns, one solution is to encrypt the data before releasing them to the outsourced server.

However, standard encryption schemes are not applicable in data outsourcing as queries cannot be executed directly on the ciphertexts. Pang and Ding [1] first studied ad hoc equijoins on encrypted relations in an outsourced database in a *private key* setting. Their scheme offers IND-CPA security on records that the database owner encrypts before sending them to the outsourced server. Subsequently, the owner may issue a query token for the server to join the encrypted records across two relations.

In this paper, we study the problem of supporting ad hoc equijoins on encrypted relations, in a *public key* setting. To exemplify the problem, consider the following application: Suppose that pharmacies are required to report the sale of certain controlled drugs to the health authority. The pharmacies encrypt each transaction with the public key of the health authority (the data owner), before submitting the encrypted records to a contractor (the server) that the owner outsources to. To pick up suspicious incidents where a patient repeatedly purchases a drug from different pharmacies, the owner issues a token for the server to perform equijoins, on the encrypted patient social security number and drug identifier, across the data collected from the pharmacies. The server should not be able to deduce the patient identities in the join results, nor the purchase patterns of the majority of patients who visit only their respective local pharmacies and hence are not in the join result. Technically, the latter implies that the server must not be allowed to perform self-joins, i.e., joining a pharmacy's data collection with itself on patients' social security number; in this sense, we say that the equijoins are *controlled*. Similar applications abound, including one where banks are required to report transfers of large cash amounts to the monetary authority or law enforcement.

1.1. Contribution

We introduce the notion of *public key encryption for relational database with controlled equijoins* (PKDE). Suppose the database includes two relations **R** and **S**. Relation **R** contains records $\{r_1, r_2, \cdots, r_m\}$; the schema of **R** is $\langle K_R, A, \cdots \rangle$ where K_R is the primary

key and A is a confidential attribute. Relation $\mathbf{S}$ contains records $\{s_1, s_2, \cdots, s_n\}$, according to schema $\langle K_S, B, \cdots \rangle$ where K_S is the primary key and B is a confidential attribute. In a PKDE scheme, with the public key of the database owner, anyone is able to encrypt records on attribute A (resp. B) before depositing them in relation $\mathbf{R}$ (resp. $\mathbf{S}$). Moreover, the database owner may generate a join token $\mathcal{T}_{AB}$ to enable the server to perform an equijoin only across attribute A of relation $\mathbf{R}$ and attribute B of relation $\mathbf{S}$, denoted as $\mathbf{R} \bowtie_{A=B} \mathbf{S}$.

Logically, the encrypted records have different confidentiality assurances before and after an equijoin, which we termed *initial confidentiality* and *post-join confidentiality* respectively. In initial confidentiality, we require that any encrypted data should achieve indistinguishability under adaptively chosen ciphertext attacks (IND-CCA2). The equijoin groups the matched records in relations $\mathbf{R}$ and $\mathbf{S}$ into equivalence classes on join attributes A and B, meaning that the records become comparable and are no longer indistinguishable. Instead, in post-join confidentiality, we require the encrypted data to possess one-way property under chosen ciphertext attacks (OW-CCA2).

Indeed, the equivalence classes produced in an equijoin may leak some statistical information on the matched records in the join result. Referring to the motivating example earlier, it implies that patients who are not reported by different pharmacies would have their privacy safeguarded, as their records would not be in the result of any equijoin authorized by the owner. Notwithstanding that, there is a need to protect the privacy of records that are unmatched in the join, just like in private set intersection protocols. In this regard, we require that the equijoin must be *controlled*; technically, a join token $\mathcal{T}_{AB}$ for $\mathbf{R} \bowtie_{A=B} \mathbf{S}$ must not be abused to carry out other join operations, including self-joins $\mathbf{R} \bowtie_{A=A} \mathbf{R}$ and $\mathbf{S} \bowtie_{B=B} \mathbf{S}$ which would reveal the equivalence classes on the unmatched records that are not in the join result.

We present a probabilistic PKDE construction on bilinear groups. The construction is formally proved to be secure with respect to initial confidentiality as well as post-join confidentiality, in the random oracle model based on the computational bilinear Diffie-Hellman (BDH) assumption. The construction is non-interactive in that, given a join token, the server can carry out the equijoin on the operand relations without any help from the database owner. An efficiency analysis shows that our scheme saves roughly 25% in storage cost compared to Pang and Ding's scheme [1], while strengthening the initial confidentiality from IND-CPA to IND-CCA2.

1.2. Related Work

Carbunar and Sion [2] first studied private join on outsourced database in a private key setting. Although their scheme supports general binary join predicates including range, equality, Hamming distance, and semantics, there is no provision to deter self-joins. Furukawa and Isshiki [3] provided a scheme where the server requires an authorization from the owner to carry out an equijoin. However, that authorization can also be used beyond the equijoin, to perform self-joins on the operand relations. Controlled equijoin for relational databases was initially investigated by Pang and Ding [1] in a private key setting. Their solution encrypts each value to 8 elements in a bilinear group, compared to 6 elements in our construction.

Yang et al. [4] introduced the notion of public key encryption with equality test (PKEET). Given two ciphertexts (that may have been produced with different public keys), anyone is able to compare and detect whether they have the same underlying message. When applied to our outsourced database problem, their scheme provides no initial confidentiality for the encrypted data. Moreover, the server could *freely* perform equijoin on any pair of relations, which does not suit our requirements. Lu, Zhang and Lin [5] showed how to achieve a stronger security model in PKEET when the message space is large and has high mininum entropy (which is not true in databases in general, and for non-unique attributes with skewed distribution in particular).

Several follow-on studies, including [6, 7, 8, 9, 10], have extended PKEET with *delegable/authorized* equality test such that only a suitably enabled server can perform equality test on the ciphertexts. All these studies do not provide a formal security model and strict proof on the initial confidentiality of data. Moreover, the schemes do not offer controlled equality test, meaning that they cannot prevent self-joins. In [6], even without any authorization, anyone is able to compare two ciphertexts generated under the same public key. The PKEET schemes proposed in [7, 8, 9, 10, 11] allow two users to authorize a tester to compare their ciphertexts. With that authorization, however, the tester is also able to compare among the ciphertexts generated by the same user. Table 1 summarizes the differentiation between the existing work and our scheme.

Private set intersection (PSI) allows two parties to compute the intersection between their respective data sets, at the same time leaking no information on the remaining data [12, 13]. Outsourced PSI [14, 15] in particular addresses how two parties may delegate set intersection operations to a server. Each input data set to a PSI protocol must contain only distinct elements; otherwise, the same element will be encoded to the same string. This implies that PSI may leak statistical information on the data when applied to databases in which different records may share the same attribute value. Furthermore, PSI does not need to provide decryption functionality for the outsourced data, unlike our PKDE scheme.

TABLE 1. Comparison with existing schemes on ciphertext equality test

Scheme	Setting	Equijoin(R, S)	Deter Self-Join (R or S)	
			Before Aut	After Aut
Yang et al. [4]	Public key	✓	×	×
Tang [6]	Public key	✓	×	×
Tang [7]	Public key	✓	✓	×
Ma et al. [8]	Public key	✓	✓	×
Huang et al. [9]	Public key	✓	✓	×
Ma et al. [10]	Public key	✓	✓	×
Lin, Qu and Zhang [11]	Public key	✓	✓	×
Carbunar and Sion [2]	Private key	✓	×	×
Furukawa and Isshiki [3]	Private key	✓	✓	×
Pang and Ding [1]	Private key	✓	✓	✓
This paper	Public key	✓	✓	✓

"Aut" denotes the authorization that issued to the server for enabling equijoin on **R** and **S**.

1.3. Paper Organization

The remainder of this paper is organized as follows. Section 2 defines the framework and security model for PKDE scheme, and covers some background on our work. Section 3 presents a construction of PKDE. We prove the security of the PKDE construction and analyze its efficiency in Section 4. Finally, Section 5 concludes the paper.

2. DEFINITIONS AND PRELIMINARIES

Our system model consists of a database owner, many users and a curious server. The database includes two relations **R** and **S** with respective confidential attributes A and B. Suppose relation **R** contains records $\{r_1, r_2, \cdots, r_m\}$ and relation **S** contains records $\{s_1, s_2, \cdots, s_n\}$. We assume that A, B are integers in $\mathcal{M} \subset [0, p)$ for some large prime number p; support for bool, float, double and string attributes can be built on top of integer operations as explained in [1].

The owner engages the server to host database. The users send encrypted records to the server, and the owner is able to request the server to execute queries on the data. The *equijoin* query, also known as *inner join*, of **R** and **S** is:

$$\mathbf{RS}_{IJ} = \mathbf{R} \bowtie_{A=B} \mathbf{S} = \{\langle r, s\rangle \mid r \in \mathbf{R}, s \in \mathbf{S}, r.A = s.B\}$$

The *outer join* of **R** and **S** is:

$$\mathbf{RS}_{OJ} = \mathbf{R} ⟗ \mathbf{S} = \mathbf{R} \bowtie_{A=B} \mathbf{S} \cup \mathbf{R}_{NM} \cup \mathbf{S}_{NM}$$

where

$$\mathbf{R}_{NM} = \{\langle r, \texttt{NULL}\rangle \mid r \in \mathbf{R} \text{ s.t. } \nexists s \in \mathbf{S} \text{ with } r.A = s.B\}$$

and

$$\mathbf{S}_{NM} = \{\langle \texttt{NULL}, s\rangle \mid s \in \mathbf{S} \text{ s.t. } \nexists r \in \mathbf{R} \text{ with } r.A = s.B\}$$

The difference between $\mathbf{RS}_{OJ}$ and $\mathbf{RS}_{IJ}$ gives the set of **R** and **S** records that found no matches (NM) in the other operand relation:

$$\mathbf{RS}_{NM} = \mathbf{RS}_{OJ} \backslash \mathbf{RS}_{IJ} = \mathbf{R}_{NM} \cup \mathbf{S}_{NM}$$

2.1. Framework

In a *public key encryption scheme for relational database with controlled equijoin* (PKDE), equality test can only be carried out on cross-relation ciphertexts after the server gets a join token from the owner and, particularly, equality test cannot be performed on unmatched ciphertexts within either relation. A PKDE scheme consists of the following procedures.

- $\texttt{KeyGen}(1^\ell) \to (PK, SK)$: Given security parameter ℓ, the database owner executes the key generation procedure to produce a pair of public and secret keys (PK, SK).
- $\texttt{Setup}(PK, SK, \mathsf{D}) \to (\mathcal{PT}, \mathcal{ST})$: On input a pair of public and private keys (PK, SK) and a description D of some relation, such as **R** or **S** with confidential attributes A and B, the database owner executes the set-up procedure to produce a pair of public and private tokens $(\mathcal{PT}, \mathcal{ST})$ for the corresponding relation.
- $\texttt{EncData}(PK, \mathcal{PT}_A, \mathbf{R}.A) \to \{\mathcal{A}_i\}_{i=1}^m$: With public key PK and public token $\mathcal{PT}_A$, any user can run the data encryption procedure on the confidential attribute A in some record r_i to produce ciphertext $\mathcal{A}_i$ for relation **R**.
- $\texttt{DecData}(SK, \mathcal{ST}_A, \{\mathcal{A}_i\}_{i=1}^m) \to \mathbf{R}.A$: The data decryption procedure, which is run by the database owner, takes as input secret key SK, private token $\mathcal{ST}_A$, and ciphertexts $\{\mathcal{A}_i\}_{i=1}^m$ for $\mathbf{R}.A$. For each record $r_i \in \mathbf{R}$, the procedure outputs plaintext $r_i.A$ if $\mathcal{A}_i$ has not been tampered with, or $\perp$ otherwise.
- $\texttt{QueryGen}(\mathcal{ST}_A, \mathcal{ST}_B) \to \mathcal{T}_{AB}$: With private tokens $\mathcal{ST}_A$ and $\mathcal{ST}_B$ for relations **R** and **S**, respectively, the query generation procedure, which is carried out by the owner of the relations, outputs query token $\mathcal{T}_{AB}$ for $\mathbf{R} \bowtie_{A=B} \mathbf{S}$.
- $\texttt{Join}(PK, \mathcal{T}_{AB}, \{\mathcal{A}_i\}_{i=1}^m, \{\mathcal{B}_j\}_{j=1}^n) \to (\mathbf{RS}_{IJ}, \mathbf{RS}_{NM})$: The join procedure, which is carried out by the server, takes as input public key PK, query token $\mathcal{T}_{AB}$, ciphertexts $\{\mathcal{A}_i\}_{i=1}^m$ for confidential attribute A in relation **R**, and ciphertexts $\{\mathcal{B}_j\}_{j=1}^n$

for confidential attribute B in relation $\mathbf{S}$. The procedure outputs $\mathbf{RS}_{IJ} = \mathbf{R} \bowtie_{A=B} \mathbf{S}$, and $\mathbf{RS}_{NM}$ which contains the unmatched $\mathbf{R}$ and $\mathbf{S}$ records.

A PKDE scheme must be able to correctly decrypt the ciphertext that it produces, as well as perform cross-relation equality test on the ciphertexts.

DEFINITION 2.1 (Correctness). *A PKDE scheme is* correct *if, for any security parameter $\ell \in \mathbb{N}$, any pair of public key and secret key $(PK, SK) \leftarrow \mathit{KeyGen}(1^\ell)$, any relation $\boldsymbol{R}$ with $r_i.A \in \mathcal{M}$ for all $r_i \in \mathbf{R}$ that is encrypted by the scheme can be decrypted, i.e., $\mathit{DecData}(SK, \mathcal{ST}_A, \{\mathcal{A}_i\}_{i=1}^m) = \mathbf{R}.A$ where $(\mathcal{PT}_A, \mathcal{ST}_A) \leftarrow \mathit{Setup}(PK, SK, \mathsf{D}_R)$ for relation $\mathbf{R}$ and $\{\mathcal{A}_i\}_{i=1}^m \leftarrow \mathit{EncData}(PK, \mathcal{PT}_A, \mathbf{R}.A)$.*

DEFINITION 2.2 (Cross-relation equality test). *A PKDE scheme possesses the* cross-relation equality test *property if, for any security parameter $\ell \in \mathbb{N}$, the following conditions hold:*

- Completeness*: This property means that every pair of matched records $\langle r_i, s_j \rangle$ will be matched by the equijoin. Specifically, for every $m \in \mathcal{M}$ and for every $r_i \in \mathbf{R}$, $s_j \in \mathbf{S}$ such that $m = r_i.A$ and $m = s_j.B$,*

$$\Pr\left[\begin{array}{l} \langle r_i, s_j \rangle \in \boldsymbol{RS}_{IJ} : \\ \quad (PK, SK) \leftarrow \mathit{KeyGen}(1^\ell), \\ \quad (\mathcal{PT}_A, \mathcal{ST}_A) \leftarrow \mathit{Setup}(PK, SK, \mathsf{D}_R), \\ \quad (\mathcal{PT}_B, \mathcal{ST}_B) \leftarrow \mathit{Setup}(PK, SK, \mathsf{D}_S), \\ \quad \{\mathcal{A}_i\}_{i=1}^m \leftarrow \mathit{EncData}(PK, \mathcal{PT}_A, \mathbf{R}.A), \\ \quad \{\mathcal{B}_j\}_{j=1}^n \leftarrow \mathit{EncData}(PK, \mathcal{PT}_B, \mathbf{S}.B), \\ \quad \mathcal{T}_{AB} \leftarrow \mathit{QueryGen}(\mathcal{ST}_A, \mathcal{ST}_B), \\ \quad (\boldsymbol{RS}_{IJ}, \boldsymbol{RS}_{NM}) \leftarrow \mathit{Join}(PK, \mathcal{T}_{AB}, \\ \qquad\qquad \{\mathcal{A}_i\}_{i=1}^m, \{\mathcal{B}_j\}_{j=1}^n) \end{array}\right] = 1$$

- Soundness*: This property means that every pair of records (r_i, s_j) with distinct encrypted values will not be matched by the equijoin. Specifically, for every polynomial time algorithm $\mathcal{P}$,*

$$\Pr\left[\begin{array}{l} \exists r_i \in \boldsymbol{R},\ \exists s_j \in \boldsymbol{S}, \\ r_i.A \neq s_j.B\ \wedge \langle r_i, s_j \rangle \in \boldsymbol{RS}_{IJ} : \\ \quad (PK, SK, \mathcal{PT}_A, \mathcal{ST}_A, \{\mathcal{A}_i\}_{i=1}^m, \\ \qquad \mathcal{PT}_B, \mathcal{ST}_B, \{\mathcal{B}_j\}_{j=1}^n) \leftarrow \mathcal{P}(1^\ell), \\ \quad \mathbf{R}.A \leftarrow \mathit{DecData}(SK, \mathcal{ST}_A, \{\mathcal{A}_i\}_{i=1}^m), \\ \quad \mathbf{S}.B \leftarrow \mathit{DecData}(SK, \mathcal{ST}_B, \{\mathcal{B}_j\}_{j=1}^n), \\ \quad \mathcal{T}_{AB} \leftarrow \mathit{QueryGen}(\mathcal{ST}_A, \mathcal{ST}_B), \\ \quad (\boldsymbol{RS}_{IJ}, \boldsymbol{RS}_{NM}) \leftarrow \mathit{Join}(PK, \mathcal{T}_{AB}, \\ \qquad\qquad \{\mathcal{A}_i\}_{i=1}^m, \{\mathcal{B}_j\}_{j=1}^n) \end{array}\right] = 0$$

- Controlled equijoin[1]*: This property means that the server cannot perform self join on $\boldsymbol{R}_{NM}$ or $\boldsymbol{S}_{NM}$ after executing equijoin on $(\boldsymbol{R}, \boldsymbol{S})$ with $\mathcal{T}_{AB}$. Specifically, for any $\ell \in \mathbb{N}$, any pair of public and secret keys $(PK, SK) \leftarrow \mathit{KeyGen}(1^\ell)$, any relations $\boldsymbol{R}$ and $\boldsymbol{S}$, after running*

$$\begin{array}{l} (\mathcal{PT}_A, \mathcal{ST}_A) \leftarrow \mathit{Setup}(PK, SK, \mathsf{D}_R), \\ (\mathcal{PT}_B, \mathcal{ST}_B) \leftarrow \mathit{Setup}(PK, SK, \mathsf{D}_S), \\ \{\mathcal{A}_i\}_{i=1}^m \leftarrow \mathit{EncData}(PK, \mathcal{PT}_A, \mathbf{R}.A), \\ \{\mathcal{B}_j\}_{j=1}^n \leftarrow \mathit{EncData}(PK, \mathcal{PT}_B, \mathbf{S}.B), \\ \mathcal{T}_{AB} \leftarrow \mathit{QueryGen}(\mathcal{ST}_A, \mathcal{ST}_B), \\ (\boldsymbol{RS}_{IJ}, \boldsymbol{RS}_{NM}) \leftarrow \mathit{Join}(PK, \mathcal{T}_{AB}, \{\mathcal{A}_i\}_{i=1}^m, \{\mathcal{B}_j\}_{j=1}^n), \end{array}$$

we have

$$(\mathit{NULL}, \boldsymbol{R}_{NM}) \leftarrow \mathit{Join}(PK, \mathcal{T}_{AB}, \boldsymbol{R}_{NM}, \boldsymbol{R}_{NM})$$

and

$$(\mathit{NULL}, \boldsymbol{S}_{NM}) \leftarrow \mathit{Join}(PK, \mathcal{T}_{AB}, \boldsymbol{S}_{NM}, \boldsymbol{S}_{NM})$$

[1]This property also means that the server can perform an equijoin only with the join token $\mathcal{T}_{AB}$. Without $\mathcal{T}_{AB}$, the server is unable to compare the encrypted records, as ensured by Initial Confidentiality (formalized in Definition 2.3).

2.2. Initial Confidentiality Requirement

Any encrypted relation that has not been involved in a join operation must satisfy *indistinguishability under adaptive chosen ciphertext attack*. The protection is captured formally by the following security game $\mathsf{Game}^{ind-cca2}_{\mathcal{E},\mathsf{PKDE}_{init}}$ which is jointly carried out by a probabilistic polynomial-time (PPT) adversary $\mathcal{E}$ and a challenger $\mathcal{C}$.

Setup: With security parameter ℓ, challenger $\mathcal{C}$ runs algorithm $\mathsf{KeyGen}(1^\ell)$ to generate a pair of public-secret keys (PK, SK). It also runs $\mathsf{Setup}(PK, SK, \mathsf{D}_R)$ to generate a pair of public and private tokens $(\mathcal{PT}_A, \mathcal{ST}_A)$ for relation $\mathbf{R}$ with confidential attribute A. Then, $\mathcal{C}$ gives PK and $\mathcal{PT}_A$ to adversary $\mathcal{E}$.

Stage 1: Adversary $\mathcal{E}$ adaptively issues queries on records of relation $\mathbf{R}$, in which A is a confidential attribute.

- *Decryption query*: $\mathcal{E}$ submits a ciphertext $\mathcal{A}_i$ to $\mathcal{C}$. The challenger invokes $\mathsf{DecData}(\cdot)$ with SK and $\mathcal{ST}_A$, and sends the result $r_i.A$ to $\mathcal{E}$ if it can be decrypted successfully; otherwise $\mathcal{C}$ sends back $\perp$.

Challenge: At the end of stage 1, the adversary randomly chooses two distinct messages $m_0, m_1 \xleftarrow{\$} \mathcal{M}$ such that $|m_0| = |m_1|$, and sends them to $\mathcal{C}$. The challenger randomly picks $d \xleftarrow{\$} \{0, 1\}$, and encrypts m_d as Y by invoking $\mathsf{EncData}(\cdot)$ with PK and $\mathcal{PT}_A$. $\mathcal{C}$ then returns challenge ciphertext Y to $\mathcal{E}$.

Stage 2: Adversary $\mathcal{E}$ poses queries in the same way as in stage 1, with the restriction that Y cannot be submitted in decryption queries.

Output: Finally, the adversary outputs a value d'. If $d' = d$, the adversary succeeds in the game, and the output of the game is defined as 1; otherwise, the output is 0.

DEFINITION 2.3 (Initial Confidentiality). *A PKDE scheme is said to offer* indistinguishability under adaptive chosen ciphertext attack *(IND-CCA2) protection*

for relations that have not been involved in join operations if, for all PPT adversary $\mathcal{E}$, there exists a negligible function $\epsilon(\cdot)$ such that:

$$\mathit{Adv}^{ind-cca2}_{\mathcal{E},PKDE_{init}}(\ell) \stackrel{def}{=} \left|\Pr\left[\mathit{Game}^{ind-cca2}_{\mathcal{E},PKDE_{init}}(\ell)=1\right]-\frac{1}{2}\right| \leq \epsilon(\ell)$$

2.3. Post-Join Confidentiality Requirement

After an equijoin, the encrypted records of PKDE scheme would not enjoy indistinguishability protection. Suppose that before the join, **R** and **S** are orthogonal relations that separately enjoy IND-CCA2 protection for their confidential attributes A and B. After the join, **R** and **S** are no longer orthogonal. Instead, logically we have joined relations $\mathbf{RS}_{IJ}$ and $\mathbf{RS}_{NM}$. Essentially, the equijoin effectively groups the matched records in **R** and **S** into equivalence classes, each containing all the matched records with the same $\mathbf{R}.A$ or $\mathbf{S}.B$ value. An adversary may be able to deduce the cleartext value corresponding to some of the classes, through frequency analysis on the class cardinalities. This is especially so with a small attribute domain $\mathcal{M}$, such as char, byte and short integer types. On the other hand, if domain $\mathcal{M}$ is large and frequency analysis is infeasible, the encrypted records enjoy *one-way* privacy, which is similar to that defined by Yang et al. [4] for PKEET scheme.

Formally, post-join confidentiality for the encrypted records in **R** (the same for **S**) is captured by the following security game $\mathtt{Game}^{ow-cca2}_{\mathcal{E},\mathsf{PKDE}_{post}}$ which is jointly carried out by a PPT adversary $\mathcal{E}$ and a challenger $\mathcal{C}$.

Setup: With security parameter ℓ, challenger $\mathcal{C}$ runs algorithm $\mathtt{KeyGen}(1^\ell)$ to generate a pair of public-secret keys (PK, SK). It also runs $\mathtt{Setup}(PK, SK)$ to generate two pairs of public and private tokens $(\mathcal{PT}_A, \mathcal{ST}_A)$, $(\mathcal{PT}_B, \mathcal{ST}_B)$ for relations **R** and **S** respectively, and query token $\mathcal{T}_{AB} \leftarrow \mathtt{QueryGen}(\mathcal{ST}_A, \mathcal{ST}_B)$. Then, $\mathcal{C}$ gives PK, $\mathcal{PT}_A$, $\mathcal{PT}_B$ and $\mathcal{T}_{AB}$ to adversary $\mathcal{E}$. This way, $\mathcal{E}$ sees relations $\mathbf{RS}_{IJ}$ and $\mathbf{RS}_{NM}$ which may be empty initially.

Stage 1: Adversary $\mathcal{E}$ adaptively poses the following queries on records of **R** and **S**. The challenger maintains all the intermediate information.[2]

- **R**-*decryption query.* $\mathcal{E}$ asks for the decryption of ciphertext $\mathcal{A}_i$ in record $r_i \in \mathbf{R}$. The challenger invokes $\mathtt{DecData}(\cdot)$ with SK and $\mathcal{ST}_A$, and sends the result $r_i.A$ to $\mathcal{E}$ if $\mathcal{A}_i$ can be decrypted successfully, or $\perp$ otherwise.
- **S**-*decryption query.* $\mathcal{E}$ asks for the decryption of ciphertext $\mathcal{B}_j$ in record $s_j \in \mathbf{S}$. The challenger invokes $\mathtt{DecData}(\cdot)$ with SK and $\mathcal{ST}_B$, and sends the result $s_j.B$ to $\mathcal{E}$ if $\mathcal{B}_j$ can be decrypted successfully, or $\perp$ otherwise.

Challenge: The challenger randomly chooses a message $m \stackrel{\$}{\leftarrow} \mathcal{M}$ for relation **R** such that m has not been involved in any of the aforementioned queries. The challenger encrypts m as Y by invoking $\mathtt{EncData}(\cdot)$ with PK and $\mathcal{PT}_A$, and sends the challenge ciphertext Y to adversary $\mathcal{E}$.

Stage 2: Adversary $\mathcal{E}$ poses queries in the same way as in stage 1, with the restriction that Y cannot be submitted in any decryption query.

Output: Finally, adversary $\mathcal{E}$ outputs a value m'. If $m' = m$, the adversary succeeds and the output of the game is defined as 1; otherwise, the output is 0.

DEFINITION 2.4 (Post-Join Confidentiality). *A P-KDE scheme is said to offer* one-way privacy under adaptive chosen ciphertext attack *(OW-CCA2) for all encrypted records after an equijoin if, for all PPT adversary $\mathcal{E}$, there exists a negligible function $\epsilon(\cdot)$ such that:*

$$\mathit{Adv}^{ow-cca2}_{\mathcal{E},PKDE_{post}}(\ell) \stackrel{def}{=} \Pr\left[\mathit{Game}^{ow-cca2}_{\mathcal{E},PKDE_{post}}(\ell)=1\right] \leq \epsilon(\ell).$$

2.4. Mathematical Assumptions

Our PKDE scheme is built on bilinear groups. Let $\mathbb{G} = \langle g \rangle$ and $\mathbb{G}_T$ be cyclic groups of prime order p. $\mathbb{G}$ is a bilinear group if there exists a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ with the following properties:

- Bilinearity: $\forall u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}^*_p$, $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$.
- Non-degeneracy: $\hat{e}(g, g) \neq 1$.
- Computability: All group operations in $\mathbb{G}, \mathbb{G}_T$ and bilinear mapping $\hat{e}(\cdot, \cdot)$ can be computed efficiently.

Our PKDE scheme will rely on the following complexity assumptions.

Discrete logarithm assumption (DL): Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order p. Given a random element $h \in_R \mathbb{G}$, any PPT algorithm $\mathcal{E}$ would have negligible probability in computing $x \in \mathbb{Z}^*_p$ such that $h = g^x$.

Computational bilinear Diffie-Hellman assumption (BDH) [16]. Let $\mathbb{G} = \langle g \rangle$ be a cyclic group with bilinear mapping $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, where $\mathbb{G}$ and $\mathbb{G}_T$ have prime order p. Given a tuple (g, g^a, g^b, g^c) for some random values $a, b, c \in_R \mathbb{Z}^*_p$, any PPT algorithm $\mathcal{E}$ would have negligible probability in computing $\hat{e}(g, g)^{abc} \in \mathbb{G}_T$.

3. A CONCRETE PKDE CONSTRUCTION

The construction of our PKDE scheme is given below. For ease of presenting the security results, we state the Setup and EncData procedures for **R** and **S** separately.

- $\mathtt{KeyGen}(1^\ell) \rightarrow (PK, SK)$: Given security parameter ℓ, construct a cyclic group $\mathbb{G} = \langle g \rangle$ with bilinear mapping $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ where $\mathbb{G}$ and

[2] A decryption query for a record in $\mathbf{RS}_{IJ}$ is equivalent to a combination of **R**- and **S**-decryption queries. Similarly, a decryption query for a record in $\mathbf{RS}_{NM}$ is equivalent to an **R**- or **S**-decryption query on the **R** or **S** fragment that is not NULL in the record. Hence we do not treat them as separate types of query.

$\mathbb{G}_T$ have order p, a large prime number such that $[0, p)$ envelops domain $\mathcal{M}$ of $\mathbf{R}.A$ and $\mathbf{S}.B$, that is, $\mathcal{M} \subset [0, p)$. Choose collision-resistant one-way hash functions $H_1 : \mathbb{G}_T \to \{0,1\}^{\log p}$ and $H_2 : \mathbb{G}^3 \to \mathbb{G}^*$. Randomly pick $\sigma, \sigma_1, \sigma_2, \sigma_3 \xleftarrow{\$} \mathbb{Z}_p^*$ such that $-\sigma_3/\sigma_1 \bmod p \notin \mathcal{M}$ (the need for this condition will become clear shortly). Compute $S = g^{\sigma}$, $h_1 = g^{\sigma_1}$, $h_2 = g^{\sigma_3}$. The owner's secret key is $SK = (\sigma, \sigma_1, \sigma_2, \sigma_3)$. The parameters $PK = (\mathbb{G}, \mathbb{G}_T, \hat{e}, p, g, S, h_1, h_2, H_1, H_2)$ are published publicly.

- $\texttt{Setup}(PK, SK, \mathsf{D}_R) \to (\mathcal{PT}_A, \mathcal{ST}_A)$: For relation $\mathbf{R}$, choose $\tau_A, \kappa_A \xleftarrow{\$} \mathbb{Z}_p^*$. Compute $\Gamma_A = g^{\sigma_2/\kappa_A}$ and $\Upsilon_A = g^{\tau_A/\sigma_2}$. The public and private tokens for relation $\mathbf{R}$ are $\mathcal{PT}_A = (\Gamma_A, \Upsilon_A)$ and $\mathcal{ST}_A = (\tau_A, \kappa_A)$, respectively.
- $\texttt{Setup}(PK, SK, \mathsf{D}_S) \to (\mathcal{PT}_B, \mathcal{ST}_B)$: Similar to the set-up procedure for relation $\mathbf{R}$.
- $\texttt{EncData}(PK, \mathcal{PT}_A, \mathbf{R}.A) \to \{\mathcal{A}_i\}_{i=1}^m$: Using PK and $\mathcal{PT}_A$, anyone may encrypt the confidential attribute A in a record for $\mathbf{R}$ before depositing it with the server. For each record $r_i \in \mathbf{R}$, let $\lambda_i, \mu_i, x_i \xleftarrow{\$} \mathbb{Z}_p^*$ and represent $r_i.A$ by encrypted tuple $\mathcal{A}_i = \langle \mathcal{A}_{i,1}, \mathcal{A}_{i,2}, \mathcal{A}_{i,3}, \mathcal{A}_{i,4}, \mathcal{A}_{i,5}, \mathcal{A}_{i,6} \rangle \in \mathbb{G}^5 \times \{0,1\}^{\log p}$ where:

$$\begin{aligned} &\mathcal{A}_{i,1} = g^{x_i}, && \mathcal{A}_{i,2} = h_1^{x_i \times r_i.A} h_2^{x_i} g^{\lambda_i}, \\ &\mathcal{A}_{i,3} = \Gamma_A^{x_i}, && \mathcal{A}_{i,4} = \Upsilon_A^{\lambda_i}, \\ &\mathcal{A}_{i,5} = g^{\mu_i}, \\ &\mathcal{A}_{i,6} = H_1(\hat{e}(S, H_2(\mathcal{A}_{i,2}, \mathcal{A}_{i,3}, \mathcal{A}_{i,4})^{\mu_i})) \oplus r_i.A. \end{aligned}$$

  Here, if $|r_i.A| < \log p$, then the binary representation of $r_i.A$ will be padded with leading "0"s. To explain the importance of the condition $-\sigma_3/\sigma_1 \bmod p \notin \mathcal{M}$ in the `Setup` procedure, suppose on the contrary that there are two records $r_1, r_2 \in \mathbf{R}$ with $r_1.A = r_2.A = -\sigma_3/\sigma_1 \bmod p$. The encryption of $r_1.A$ and $r_2.A$ will include $\mathcal{A}_{1,2} = g^{\lambda_1}$, $\mathcal{A}_{1,4} = \Upsilon_A^{\lambda_1}$ and $\mathcal{A}_{2,2} = g^{\lambda_2}$, $\mathcal{A}_{2,4} = \Upsilon_A^{\lambda_2}$, respectively. This allows an adversary to deduce that $\hat{e}(\mathcal{A}_{1,2}, \mathcal{A}_{2,4}) = \hat{e}(\mathcal{A}_{1,4}, \mathcal{A}_{2,2})$ implies $r_1.A = r_2.A$, without any query token.
- $\texttt{EncData}(PK, \mathcal{PT}_B, \mathbf{S}.B) \to \{\mathcal{B}_j\}_{j=1}^n$: Using PK and $\mathcal{PT}_B$, anyone may encrypt the confidential attribute B in a record for $\mathbf{S}$ before depositing it with the server. For each record $s_j \in \mathbf{S}$, let $\lambda_j, \mu_j, y_j \xleftarrow{\$} \mathbb{Z}_p^*$ and represent $s_j.B$ by encrypted tuple $\mathcal{B}_j = \langle \mathcal{B}_{j,1}, \mathcal{B}_{j,2}, \mathcal{B}_{j,3}, \mathcal{B}_{j,4}, \mathcal{B}_{j,5}, \mathcal{B}_{j,6} \rangle \in \mathbb{G}^5 \times \{0,1\}^{\log p}$ where:

$$\begin{aligned} &\mathcal{B}_{j,1} = g^{y_j}, && \mathcal{B}_{j,2} = h_1^{y_j \times s_j.B} h_2^{y_j} g^{\lambda_j}, \\ &\mathcal{B}_{j,3} = \Gamma_B^{y_j}, && \mathcal{B}_{j,4} = \Upsilon_B^{\lambda_j}, \\ &\mathcal{B}_{j,5} = g^{\mu_j}, \\ &\mathcal{B}_{j,6} = H_1(\hat{e}(S, H_2(\mathcal{B}_{j,2}, \mathcal{B}_{j,3}, \mathcal{B}_{j,4})^{\mu_j})) \oplus s_j.B. \end{aligned}$$

- $\texttt{DecData}(SK, \mathcal{ST}_A, \{\mathcal{A}_i\}_{i=1}^m) \to \mathbf{R}.A$: For all $r_i \in \mathbf{R}$, the owner computes

$$r_i.A = \mathcal{A}_{i,6} \oplus H_1(\hat{e}(\mathcal{A}_{i,5}, H_2(\mathcal{A}_{i,2}, \mathcal{A}_{i,3}, \mathcal{A}_{i,4})^{\sigma})),$$

  then verifies that $r_i.A \in \mathcal{M}$ and

$$\hat{e}(\mathcal{A}_{i,1}, \mathcal{A}_{i,2}) \stackrel{?}{=} \hat{e}(\mathcal{A}_{i,1}, \mathcal{A}_{i,1})^{\sigma_1 \times r_i.A + \sigma_3} \cdot \hat{e}\left(\mathcal{A}_{i,3}^{\kappa_A/\tau_A}, \mathcal{A}_{i,4}\right) \quad (1)$$

  If either of the checks fails, the user sets $r_i.A = \perp$ to signify that $\mathcal{A}_i$ is corrupted.
- $\texttt{DecData}(SK, \mathcal{ST}_B, \{\mathcal{B}_j\}_{j=1}^n) \to \mathbf{S}.B$: Similar to the decryption procedure for relation $\mathbf{R}$.
- $\texttt{QueryGen}(\mathcal{ST}_A, \mathcal{ST}_B) \to \mathcal{T}_{AB}$: For equijoin query $\mathbf{R} \bowtie_{A=B} \mathbf{S}$, the owner generates a token $\mathcal{T}_{AB} = \langle \kappa_B/\tau_A, \kappa_A/\tau_B \rangle$ and sends it to the server.
- $\texttt{Join}(PK, \mathcal{T}_{AB}, \{\mathcal{A}_i\}_{i=1}^m, \{\mathcal{B}_j\}_{j=1}^n) \to (\mathbf{RS}_{IJ}, \mathbf{RS}_{NM})$: If any pair of records $r_i \in \mathbf{R}$ with ciphertext $\mathcal{A}_i = \langle \mathcal{A}_{i,1}, \mathcal{A}_{i,2}, \mathcal{A}_{i,3}, \mathcal{A}_{i,4}, \mathcal{A}_{i,5}, \mathcal{A}_{i,6} \rangle$, and $s_j \in \mathbf{S}$ with ciphertext $\mathcal{B}_j = \langle \mathcal{B}_{j,1}, \mathcal{B}_{j,2}, \mathcal{B}_{j,3}, \mathcal{B}_{j,4}, \mathcal{B}_{j,5}, \mathcal{B}_{j,6} \rangle$, satisfy the following condition,

$$\hat{e}\left(\mathcal{A}_{i,2}, \mathcal{B}_{j,1}\right) \cdot \hat{e}\left(\mathcal{A}_{i,3}, \mathcal{B}_{j,4}^{\kappa_A/\tau_B}\right) \stackrel{?}{=} \hat{e}\left(\mathcal{B}_{j,2}, \mathcal{A}_{i,1}\right) \cdot \hat{e}\left(\mathcal{B}_{j,3}^{\kappa_B/\tau_A}, \mathcal{A}_{i,4}\right) \quad (2)$$

  then the server determines that $r_i.A = s_j.B$ and inserts $\langle r_i, s_j \rangle$ into $\mathbf{RS}_{IJ}$. $\mathbf{R}$ and $\mathbf{S}$ records that have no matching counterpart in $\mathbf{S}$ and $\mathbf{R}$, respectively, are inserted into $\mathbf{RS}_{NM}$.

THEOREM 3.1. *The PKDE scheme proposed above is correct.*

Proof. Elaborating on Formula (2), due to the properties of the bilinear group,

$$\begin{aligned} \hat{e}\left(\mathcal{B}_{j,3}^{\kappa_B/\tau_A}, \mathcal{A}_{i,4}\right) &= \hat{e}\left(\left(g^{\sigma_2/\kappa_B}\right)^{y_j \kappa_B/\tau_A}, \left(g^{\tau_A/\sigma_2}\right)^{\lambda_i}\right) \\ &= \hat{e}\left(g^{y_j}, g^{\lambda_i}\right) \end{aligned}$$

$$\begin{aligned} \hat{e}\left(\mathcal{A}_{i,3}, \mathcal{B}_{j,4}^{\kappa_A/\tau_B}\right) &= \hat{e}\left(\left(g^{\sigma_2/\kappa_A}\right)^{x_i}, \left(g^{\tau_B/\sigma_2}\right)^{\lambda_j \kappa_A/\tau_B}\right) \\ &= \hat{e}\left(g^{x_i}, g^{\lambda_j}\right) \end{aligned}$$

Moreover,

$$\begin{aligned} \hat{e}\left(\mathcal{A}_{i,2}, \mathcal{B}_{j,1}\right) &= \hat{e}\left(g^{\sigma_1 x_i \times r_i.A + x_i \sigma_3}, g^{y_j}\right) \cdot \hat{e}\left(g^{\lambda_i}, g^{y_j}\right) \\ &= \hat{e}\left(g^{x_i}, g^{y_j}\right)^{\sigma_1 \times r_i.A + \sigma_3} \cdot \hat{e}\left(\mathcal{B}_{j,3}^{\kappa_B/\tau_A}, \mathcal{A}_{i,4}\right) \end{aligned}$$

$$\begin{aligned} \Rightarrow \hat{e}\left(\mathcal{A}_{i,2}, \mathcal{B}_{j,1}\right) \cdot \hat{e}\left(\mathcal{B}_{j,3}^{\kappa_B/\tau_A}, \mathcal{A}_{i,4}\right)^{-1} \\ = \hat{e}\left(g^{x_i}, g^{y_j}\right)^{\sigma_1 \times r_i.A + \sigma_3} \end{aligned} \quad (3)$$

Similarly,

$$\begin{aligned} \hat{e}\left(\mathcal{B}_{j,2}, \mathcal{A}_{i,1}\right) &= \hat{e}\left(g^{\sigma_1 y_j \times s_j.B + y_j \sigma_3}, g^{x_i}\right) \cdot \hat{e}\left(g^{\lambda_j}, g^{x_i}\right) \\ &= \hat{e}\left(g^{x_i}, g^{y_j}\right)^{\sigma_1 \times s_j.B + \sigma_3} \cdot \hat{e}\left(\mathcal{A}_{i,3}, \mathcal{B}_{j,4}^{\kappa_A/\tau_B}\right) \end{aligned}$$

$$\Rightarrow \hat{e}\left(\mathcal{B}_{j,2}, \mathcal{A}_{i,1}\right) \cdot \hat{e}\left(\mathcal{A}_{i,3}, \mathcal{B}_{j,4}^{\kappa_A/\tau_B}\right)^{-1} = \hat{e}\left(g^{x_i}, g^{y_j}\right)^{\sigma_1 \times s_j.B + \sigma_3} \quad (4)$$

If $r_i.A = s_j.B$, the right-hand-side of Equations (3) and (4) are the same, so the left-hand-side of the equations must match. Consequently, the condition in Formula (2) must hold.

With $\mathcal{A}_{i,1}, \mathcal{A}_{i,2}, \mathcal{A}_{i,3}, \mathcal{A}_{i,4}$ as defined in the $\texttt{EncData}(\cdot)$ procedure,

$$\hat{e}(\mathcal{A}_{i,1}, \mathcal{A}_{i,2}) = \hat{e}\left(g^{x_i}, g^{x_i}\right)^{\sigma_1 \times r_i.A + \sigma_3} \cdot \hat{e}\left(g^{x_i}, g^{\lambda_i}\right)$$

and

$$\begin{aligned}
&\hat{e}(\mathcal{A}_{i,1}, \mathcal{A}_{i,1})^{\sigma_1 \times r_i.A + \sigma_3} \cdot \hat{e}\left(\mathcal{A}_{i,3}^{\kappa_A/\tau_A}, \mathcal{A}_{i,4}\right) \\
&= \hat{e}(g^{x_i}, g^{x_i})^{\sigma_1 \times r_i.A + \sigma_3} \cdot \hat{e}\left((g^{\sigma_2/\kappa_A})^{x_i \kappa_A/\tau_A}, (g^{\tau_A/\sigma_2})^{\lambda_i}\right) \\
&= \hat{e}\left(g^{x_i}, g^{x_i}\right)^{\sigma_1 \times r_i.A + \sigma_3} \cdot \hat{e}\left(g^{x_i}, g^{\lambda_i}\right)
\end{aligned}$$

Therefore, equality (1) holds and $r_i.A$ is decrypted correctly. □

4. ANALYSIS

4.1. Security Results

THEOREM 4.1. *The proposed PKDE scheme in Section 3 is complete, sound and controlled for cross-relation equijoin.*

Proof. We first show the completeness property. For any key pair $(PK, SK) \leftarrow \texttt{KeyGen}(1^\ell)$, relation tokens $(\mathcal{PT}_A, \mathcal{ST}_A) \leftarrow \texttt{Setup}(PK, SK, \mathsf{D}_R)$ and $(\mathcal{PT}_B, \mathcal{ST}_B) \leftarrow \texttt{Setup}(PK, SK, \mathsf{D}_S)$, $\{\mathcal{A}_i\}_{i=1}^m \leftarrow \texttt{EncData}(PK, \mathcal{PT}_A, \mathbf{R}.A)$, $\{\mathcal{B}_j\}_{j=1}^n \leftarrow \texttt{EncData}(PK, \mathcal{PT}_B, \mathbf{S}.B)$, $\mathcal{T}_{AB} \leftarrow \texttt{QueryGen}(\mathcal{ST}_A, \mathcal{ST}_B)$, for every $m \in \mathcal{M}$ and every $r_i \in \mathbf{R}, s_j \in \mathbf{S}$ such that $m = r_i.A$ and $m = s_j.B$, let $\mathcal{A}_i = \langle \mathcal{A}_{i,1}, \cdots, \mathcal{A}_{i,6} \rangle$ be the ciphertext of $r_i.A$ and $\mathcal{B}_j = \langle \mathcal{B}_{j,1}, \cdots, \mathcal{B}_{j,6} \rangle$ the ciphertext of $s_j.B$. Equality (2) holds because $r_i.A = s_j.B$, so $\langle r_i, s_j \rangle \in \mathbf{RS}_{IJ}$.

Next, consider the soundness property. Given $\{\mathcal{A}_i\}_{i=1}^m$ for $\mathbf{R}.A$ and $\{\mathcal{B}_j\}_{j=1}^n$ for $\mathbf{S}.B$, along with query token $\mathcal{T}_{AB} = (\kappa_B/\tau_A, \kappa_A/\tau_B)$. Equality (2) cannot hold for any $r_i \in \mathbf{R}$, $s_j \in \mathbf{S}$ if $r_i.A \neq s_j.B$. This is because $r_i.A \neq s_j.B$ implies $\hat{e}(g^{x_i}, g^{y_j})^{\sigma_1 \times r_i.A + \sigma_3} \neq \hat{e}(g^{x_i}, g^{y_j})^{\sigma_1 \times s_j.B + \sigma_3}$.

We continue to consider the property of controlled equijoin. Without loss of generality, consider two distinct ciphertexts $\mathcal{A}_i = \langle \mathcal{A}_{i,1}, \cdots, \mathcal{A}_{i,6} \rangle, \mathcal{A}_j = \langle \mathcal{A}_{j,1}, \cdots, \mathcal{A}_{j,6} \rangle \in \mathbf{R}_{NM}$. To perform equality test on them, Formula (2) can only hold in the following way:

$$\hat{e}(\mathcal{A}_{i,2}, \mathcal{A}_{j,1}) \cdot \hat{e}(\mathcal{A}_{i,3}, \mathcal{A}_{j,4})^{\kappa_A/\tau_A} = \hat{e}(\mathcal{A}_{j,2}, \mathcal{A}_{i,1}) \cdot \hat{e}(\mathcal{A}_{j,3}, \mathcal{A}_{i,4})^{\kappa_A/\tau_A}$$

which requires the server knowing κ_A/τ_A. However, it is impossible for the server to deduce κ_A/τ_A from the query token $\mathcal{T}_{AB}$. □

THEOREM 4.2. *Prior to an equijoin, the proposed PKDE scheme in Section 3 is IND-CCA2 secure in the random oracle model assuming that the computational BDH assumption holds.*

The following proof follows the standard framework established in [16, 17].

Proof. Let $\mathcal{E} = (\mathcal{E}_1, \mathcal{E}_2)$ be a PPT adversary that has advantage ϵ in attacking the IND-CCA2 security of the PKDE scheme in the initial phase. Suppose $\mathcal{E}$ issues at most q_D decryption queries, at most q_{H_1} hash queries of H_1, and at most q_{H_2} hash queries of H_2 (here, q_D, q_{H_1} and q_{H_2} are positive). Using $\mathcal{E}$, we construct an algorithm $\mathcal{I}$ to solve the computational BDH problem with non-negligible probability.

At first, algorithm $\mathcal{I}$ is given BDH parameters $(\mathbb{G}, \mathbb{G}_T, p, \hat{e})$ and a BDH instance (g, g^a, g^b, g^c). The goal of $\mathcal{I}$ is to compute $\Lambda = \hat{e}(g, g)^{abc}$. Algorithm $\mathcal{I}$ simulates the challenger and interacts with adversary $\mathcal{E}$ as follows:

Setup. Algorithm $\mathcal{I}$ randomly picks $\sigma_1, \sigma_2, \sigma_3, \kappa_A, \tau_A \xleftarrow{\$} \mathbb{Z}_p^*$, sets $S = g^a$, and computes the elements in PK and $\mathcal{PT}_A$ according to the proposed scheme.

Adversary $\mathcal{E}$ is able to issue hash queries to H_1 and H_2 at any time as follows. $\mathcal{O}_{H_1}$ and $\mathcal{O}_{H_2}$ are random oracles controlled by $\mathcal{I}$. To respond to these two types of hash queries, $\mathcal{I}$ maintains two lists $\mathcal{L}_1$ and $\mathcal{L}_2$ of tuples. These lists are initially empty.

$\mathbf{H_1}$-queries. For input element $T_i \in \mathbb{G}_T$, if there is a pair $(T_i, \delta_i) \in \mathcal{L}_1$, then return δ_i; if not, return a random value $\delta_i \xleftarrow{\$} \{0,1\}^{\log p}$, and append the tuple (T_i, δ_i) to $\mathcal{L}_1$.

$\mathbf{H_2}$-queries. For input tuple $(T_{i,1}, T_{i,2}, T_{i,3}) \in \mathbb{G}^3$, algorithm $\mathcal{I}$ responds as follows:

1. If there is a tuple $(T_{i,1}, T_{i,2}, T_{i,3}, \omega_i, \Omega_i, cn_i) \in \mathcal{L}_2$, then return Ω_i.
2. If not, randomly pick a coin $cn_i \in \{0,1\}$ such that $\Pr[cn_i = 0] = \rho$ (ρ will be determined later). Randomly pick a value $\omega_i \xleftarrow{\$} \mathbb{Z}_p^*$.
   - If $cn_i = 0$, compute $\Omega_i = g^b \cdot g^{\omega_i} \in \mathbb{G}$.
   - If $cn_i = 1$, compute $\Omega_i = g^{\omega_i} \in \mathbb{G}$.

   Return Ω_i, and append $(T_{i,1}, T_{i,2}, T_{i,3}, \omega_i, \Omega_i, cn_i)$ to $\mathcal{L}_2$.

Decryption queries. For input ciphertext $\langle \mathcal{A}_{i,1}, \cdots, \mathcal{A}_{i,6} \rangle$, algorithm $\mathcal{I}$ queries $\mathcal{O}_{H_2}$ to get $H_2(\mathcal{A}_{i,2}, \mathcal{A}_{i,3}, \mathcal{A}_{i,4}) = g^{\omega_i} \in \mathbb{G}$. Let $(T_{i,1}, T_{i,2}, T_{i,3}, \omega_i, \Omega_i, cn_i)$ be the corresponding tuple on the query list $\mathcal{L}_2$.

- If $cn_i = 0$, then algorithm $\mathcal{I}$ reports failure and terminates.
- Otherwise, we know $cn_i = 1$ and $\Omega_i = g^{\omega_i} \in \mathbb{G}$. Algorithm $\mathcal{I}$ queries $\mathcal{O}_{H_1}$ and gets $\delta_i \leftarrow \mathcal{O}_{H_1}(\hat{e}(\mathcal{A}_{i,5}, S^{\omega_i}))$. Following that, $\mathcal{I}$ returns $m_i' \leftarrow \delta_i \oplus \mathcal{A}_{i,6}$ if equality (1) is satisfied, or $\perp$ otherwise.

Challenge. With public key PK and public token $\mathcal{PT}_A$, $\mathcal{E}$ outputs m_0, m_1. Algorithm $\mathcal{I}$ chooses random

values $d \xleftarrow{\$} \{0,1\}$, $\lambda, x \xleftarrow{\$} \mathbb{Z}_p^*$, $\Theta \xleftarrow{\$} \{0,1\}^{\log p}$, and computes

$$\mathcal{A}_{d,1} = g^x, \qquad \mathcal{A}_{d,2} = h_1^{x \times m_d} h_2^x g^\lambda,$$
$$\mathcal{A}_{d,3} = \Gamma_A^x, \qquad \mathcal{A}_{d,4} = \Upsilon_A^\lambda.$$

Next, $\mathcal{I}$ runs $\mathcal{O}_{H_2}$ with $(\mathcal{A}_{d,2}, \mathcal{A}_{d,3}, \mathcal{A}_{d,4})$. If $cn_d = 1$, then algorithm $\mathcal{I}$ reports failure and terminates. If not, $\mathcal{I}$ sets the other two components in the challenge ciphertext as follows:

$$\mathcal{A}_{d,5} = g^c, \qquad \mathcal{A}_{d,6} = \Theta.$$

Algorithm $\mathcal{I}$ gives the challenge ciphertext $Y = \langle \mathcal{A}_{d,1}, \cdots, \mathcal{A}_{d,6} \rangle$ to $\mathcal{E}$.

Note that challenge ciphertext Y implicitly defines $H_1(\hat{e}(S^c, H_2(\mathcal{A}_{d,2}, \mathcal{A}_{d,3}, \mathcal{A}_{d,4}))) = \Theta \oplus m_d$. That is,

$$\Theta \oplus m_d = H_1(\hat{e}(g^{ac}, g^{b+\omega_d})) = H_1(\hat{e}(g,g)^{ac(b+\omega_d)})$$

Thus, Y is a valid ciphertext for message m_d.

More decryption queries. Adversary $\mathcal{E}$ can issue more decryption queries for ciphertexts $\langle \mathcal{A}_{i,1}, \cdots, \mathcal{A}_{i,6} \rangle$, with the restriction that Y cannot be submitted in decryption queries.

Guess. Eventually, $\mathcal{E}$ outputs a guess $d' \in \{0,1\}$ to indicate whether Y is an encryption of m_0 or m_1. Algorithm $\mathcal{I}$ randomly picks a tuple (T_o, δ_o) from list $\mathcal{L}_1$ and outputs $T_o / \hat{e}(g^a, g^c)^{\omega_d}$ as the solution $\hat{e}(g,g)^{abc}$ to the given BDH instance, where ω_d is generated in the challenge phase.

Note that the responses to H_1 and H_2 queries are uniformly and independently distributed in $\mathbb{Z}_p^*$ and $\mathbb{G}^*$, respectively, which means they perfectly simulate a real attack. Thus, if algorithm $\mathcal{I}$ does not abort in the simulation, then adversary $\mathcal{E}$'s view is identical to that in the real attack. That means if $\mathcal{I}$ does not abort, then $|\Pr[d = d'] - \frac{1}{2}| \geq \epsilon$.

We continue to show the probability that algorithm $\mathcal{I}$ aborts during the simulation. Suppose adversary $\mathcal{E}$ issues q_D decryption queries in total. Thus, the probability that algorithm $\mathcal{I}$ does not abort in the decryption queries in the two phases is:

$$P_D = \Pr[\neg \textit{abort} \text{ in } \mathcal{O}_D] = \rho^{q_D}$$

Also, the probability that $\mathcal{I}$ does not abort in the challenge phase is:

$$P_C = \Pr[\neg \textit{abort} \text{ in challenge}] = 1 - \rho$$

Therefore, the probability that algorithm $\mathcal{I}$ does not abort in the simulation is:

$$P_N = P_D \cdot P_C = \rho^{q_D}(1-\rho)$$

P_N is maximum when $\rho = \frac{q_D}{q_D+1}$. Accordingly, the probability that $\mathcal{I}$ does not abort in the simulation is at least $\frac{1}{e(q_D+1)}$, which implies that algorithm $\mathcal{I}$'s advantage is at least $\frac{\epsilon}{e(q_D+1)}$. Here, e is the base of the natural logarithm.

We then show that algorithm $\mathcal{I}$ outputs a correct answer $\Lambda = \hat{e}(g,g)^{abc}$ with probability at least $\frac{2\epsilon}{eq_{H_1}(q_D+1)}$. Let $\mathbb{E}$ be the event that $\hat{e}(S^c, H_2(\mathcal{A}_{d,2}, \mathcal{A}_{d,3}, \mathcal{A}_{d,4}))$ has been submitted to $\mathcal{O}_{H_1}$ at some point. We prove the following lemma, in an identical way as [16, Claims 1 and 2].

LEMMA 4.1. $\Pr[\mathbb{E}] \geq \frac{2\epsilon}{e(q_D+1)}$.

Proof. If event $\mathbb{E}$ happened, $\hat{e}(S^c, H_2(\mathcal{A}_{d,2}, \mathcal{A}_{d,3}, \mathcal{A}_{d,4}))$ would appear in some tuple of $\mathcal{L}_1$ at the end of the security game. It is easy to show that $\Pr[\mathbb{E}]$ in the above security game is the same as in a real attack. Let $\mathbb{E}_i$ be the event that $\hat{e}(S^c, H_2(\mathcal{A}_{d,2}, \mathcal{A}_{d,3}, \mathcal{A}_{d,4}))$ is submitted to $\mathcal{O}_{H_1}$ in the first i H_1 queries by adversary $\mathcal{E}$. By induction, we have $\Pr[\mathbb{E}_0] = 0$ in both cases of the simulated security game and the real attack. Assume that for $i > 0$, $\Pr[\mathbb{E}_{i-1}]$ in the simulation equals to that in the real attack. Thus, we have

$$\begin{aligned}\Pr[\mathbb{E}_i] &= \Pr[\mathbb{E}_i|\mathbb{E}_{i-1}]\Pr[\mathbb{E}_{i-1}] + \Pr[\mathbb{E}_i|\neg\mathbb{E}_{i-1}]\Pr[\neg\mathbb{E}_{i-1}] \\ &= \Pr[\mathbb{E}_{i-1}] + \Pr[\mathbb{E}_i|\neg\mathbb{E}_{i-1}]\Pr[\neg\mathbb{E}_{i-1}]\end{aligned}$$

Due to the perfectness of the simulation, $\Pr[\mathbb{E}_i|\neg\mathbb{E}_{i-1}]$ in the simulation is equal to that in the real attack, which in turn means that $\Pr[\mathbb{E}]$ in the two cases are identical.

In the real attack, if event $\mathbb{E}$ does not happen, then the decryption of the challenge ciphertext Y is independent of adversary $\mathcal{E}$'s view, which means $\Pr[d' = d|\neg\mathbb{E}] = 1/2$. Hence, the following facts hold.

$$\begin{aligned}\Pr[d' = d] &= \Pr[d' = d|\mathbb{E}]\Pr[\mathbb{E}] + \Pr[d' = d|\neg\mathbb{E}]\Pr[\neg\mathbb{E}] \\ &\leq \Pr[\mathbb{E}] + \frac{1}{2}\Pr[\neg\mathbb{E}] = \frac{1}{2} + \frac{1}{2}\Pr[\mathbb{E}], \\ \Pr[d' = d] &\geq \Pr[d' = d|\neg\mathbb{E}]\Pr[\neg\mathbb{E}] = \frac{1}{2}\Pr[\neg\mathbb{E}] \\ &= \frac{1}{2} - \frac{1}{2}\Pr[\mathbb{E}],\end{aligned}$$

Together, they imply that

$$\left|\Pr[d' = d] - \frac{1}{2}\right| \leq \frac{1}{2}\Pr[\mathbb{E}]. \tag{5}$$

On the other hand, $|\Pr[d' = d] - 1/2| \geq \frac{\epsilon}{e(q_D+1)}$. Combine with Equality (5), we get the relationship

$$\frac{1}{2}\Pr[\mathbb{E}] \geq \frac{\epsilon}{e(q_D+1)}.$$

Therefore, $\Pr[\mathbb{E}] \geq \frac{2\epsilon}{e(q_D+1)}$ holds in the real attack as well as in the simulation. □

Thus, the success probability of algorithm $\mathcal{I}$ picking a correct answer is at least $\frac{2\epsilon}{eq_{H_1}(q_D+1)}$. That is,

$$\mathtt{Adv}_{\mathcal{I}}^{BDH}(\ell) = \Pr[T_o = \Lambda \cdot \hat{e}(g^a, g^c)^{\omega_d}] \geq \frac{2\epsilon}{eq_{H_1}(q_D+1)}$$

If ϵ is not negligible, neither is $\mathtt{Adv}_{\mathcal{I}}^{BDH}(\ell)$, which contradicts the computational BDH assumption. This concludes Theorem 4.2. □

THEOREM 4.3. *After an equijoin, the proposed PKDE scheme in Section 3 is OW-CCA2 secure in the random oracle model assuming that the computational BDH assumption holds.*

The following proof follows the standard framework established in [4].

Proof. Let $\mathcal{E}$ be a PPT adversary attacking the OW-CCA2 security of the PKDE scheme after an equijoin. Suppose $\mathcal{E}$ issues at most q_D decryption queries (including both **R**- and **S**-decryption queries), at most q_{H_1} hash queries of H_1, and at most q_{H_2} hash queries of H_2 (here, q_D, q_{H_1} and q_{H_2} are positive). Let $\mathtt{Adv}^{ow-cca2}_{\mathcal{E},\mathsf{PKDE}_{post}}(\ell)$ denote the advantage of $\mathcal{E}$'s attacks in the security game. Using $\mathcal{E}$, we then construct an algorithm $\mathcal{I}$ to solve the computational BDH problem with non-negligible probability. Algorithm $\mathcal{I}$ simulates the challenger and answers the queries of adversary $\mathcal{E}$.

More specifically, we prove the theorem using hybrid games $\mathtt{Game}_{post}G_i$. The first one $\mathtt{Game}_{post}G_0$ models the original OW-CCA2 security game, the second shows that the two-phase OW-CCA2 security game $\mathtt{Game}_{post}G_0$ can be simplified with a negligible security sacrifice, and the following games are perfectly simulated in the random oracle model. Let Ψ_i denote the event that $m' = m$ in $\mathtt{Game}_{post}G_i$. Accordingly, $\mathcal{E}$'s success probability in $\mathtt{Game}_{post}G_i$ is $\Pr[\Psi_i]$. We first consider the original security game.

$\mathtt{Game}_{post}G_0$

1 Pick $\sigma, \sigma_1, \sigma_2, \sigma_3, \kappa_A, \tau_A, \kappa_B, \tau_B \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$. Compute $S = g^{\sigma}$, $h_1 = g^{\sigma_1}$, $h_2 = g^{\sigma_3}$, $\Gamma_A = g^{\sigma_2/\kappa_A}$, $\Upsilon_A = g^{\tau_A/\sigma_2}$, $\Gamma_B = g^{\sigma_2/\kappa_B}$, $\Upsilon_B = g^{\tau_B/\sigma_2}$, and $\mathcal{T}_{AB} = \langle \kappa_B/\tau_A, \kappa_A/\tau_B \rangle$.

2 Generate $\mathtt{state} \leftarrow \mathcal{E}_1^{\mathcal{O}_{H_1},\mathcal{O}_{H_2},\mathcal{O}_{D_1},\mathcal{O}_{D_2}}(PK, \mathcal{PT}_A, \mathcal{PT}_B, \mathcal{T}_{AB}, *)$ with the following oracles.

- $\mathcal{O}_{H_1}$: For an input element T, $\mathcal{O}_{H_1}$ responds with a random value in a consistent way, meaning that the same value will be returned for the same input. When composing a ciphertext, the adversary should invoke this oracle; similarly, this oracle is invoked in answering decryption queries.
- $\mathcal{O}_{H_2}$: Similar to $\mathcal{O}_{H_1}$, for an input tuple (T_1, T_2, T_3), $\mathcal{O}_{H_2}$ responds with a random value in a consistent way. When composing a ciphertext, the adversary should invoke this oracle; similarly, this oracle is invoked in answering decryption queries.
- $\mathcal{O}_{D_1}$: For a decryption query on $\mathcal{A}_i = \langle \mathcal{A}_{i,1}, \cdots, \mathcal{A}_{i,6} \rangle$, $\mathcal{O}_{D_1}$ runs the decryption procedure of the PKDE scheme using private parameters $\sigma, \sigma_1, \sigma_2, \sigma_3, \kappa_A, \tau_A$.
- $\mathcal{O}_{D_2}$: For a decryption query on $\mathcal{B}_j = \langle \mathcal{B}_{j,1}, \cdots, \mathcal{B}_{j,6} \rangle$, $\mathcal{O}_{D_2}$ runs the decryption procedure of the PKDE scheme using private parameters $\sigma, \sigma_1, \sigma_2, \sigma_3, \kappa_B, \tau_B$.

3 $m \stackrel{\$}{\leftarrow} \mathcal{M}$, $Y \leftarrow \mathtt{EncData}(PK, \mathcal{PT}_A, m)$.

4 $m' \leftarrow \mathcal{E}_2^{\mathcal{O}_{H_1},\mathcal{O}_{H_2},\mathcal{O}_{D_1},\mathcal{O}_{D_2}}(PK, \mathcal{PT}_A, \mathcal{PT}_B, \mathcal{T}_{AB}, \mathtt{state}, Y, *)$, where the oracles work in the same way as in step 2 except that decryption queries on Y are not allowed.

5 If $m' = m$, $\mathcal{E}$ wins the game.

Next, we define a simplified security game and show its relationship with $\mathtt{Game}_{post}G_0$.

$\mathtt{Game}_{post}G_1$

1 Pick $\sigma, \sigma_1, \sigma_2, \sigma_3, \kappa_A, \tau_A, \kappa_B, \tau_B \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$. Compute $S = g^{\sigma}$, $h_1 = g^{\sigma_1}$, $h_2 = g^{\sigma_3}$, $\Gamma_A = g^{\sigma_2/\kappa_A}$, $\Upsilon_A = g^{\tau_A/\sigma_2}$, $\Gamma_B = g^{\sigma_2/\kappa_B}$, $\Upsilon_B = g^{\tau_B/\sigma_2}$, and $\mathcal{T}_{AB} = \langle \kappa_B/\tau_A, \kappa_A/\tau_B \rangle$.

2 $m \stackrel{\$}{\leftarrow} \mathcal{M}$, $Y \leftarrow \mathtt{EncData}(PK, \mathcal{PT}_A, m)$.

3 $m' \leftarrow \mathcal{E}^{\mathcal{O}_{H_1},\mathcal{O}_{H_2},\mathcal{O}_{D_1},\mathcal{O}_{D_2}}(PK, \mathcal{PT}_A, \mathcal{PT}_B, \mathcal{T}_{AB}, Y, *)$ with the following oracles.

- $\mathcal{O}_{H_1}$: The same as in $\mathtt{Game}_{post}G_0$.
- $\mathcal{O}_{H_2}$: The same as in $\mathtt{Game}_{post}G_0$.
- $\mathcal{O}_{D_1}$: The same as in $\mathtt{Game}_{post}G_0$.
- $\mathcal{O}_{D_2}$: The same as in $\mathtt{Game}_{post}G_0$.

4 If $m' = m$, $\mathcal{E}$ wins the game.

LEMMA 4.2. $\Pr[\Psi_0] \leq \Pr[\Psi_1] + \frac{q_{D'}}{|\mathcal{M}|}$, *where $q_{D'}$ denotes the total number of the two types of decryption queries in step 2 of $\mathtt{Game}_{post}G_0$.*

Proof. For ease of presentation, let $\mathcal{E} = (\mathcal{E}_1, \mathcal{E}_2)$ and $\mathcal{F}$ denote the adversaries in $\mathtt{Game}_{post}G_0$ and $\mathtt{Game}_{post}G_1$, respectively. Adversary $\mathcal{F}$ initiates $\mathtt{Game}_{post}G_1$ and receives challenge ciphertext Y from the security game, before running $\mathcal{E}$. For $\mathcal{E}_1$'s hash queries to $\mathcal{O}_{H_1}$ and $\mathcal{O}_{H_2}$, $\mathcal{F}$ responds using its corresponding oracles. For decryption queries, $\mathcal{F}$ uses its own $\mathcal{O}_{D_1}$ and $\mathcal{O}_{D_2}$; the exception is that $\mathcal{F}$ aborts the security game if some Y' such that $\mathtt{DecData}(SK, *, Y') = m$ is submitted, which happens with probability at most $\frac{1}{|\mathcal{M}|}$. Eventually, $\mathcal{E}_1$ terminates and outputs a state $\mathtt{state}$. Next, $\mathcal{F}$ runs $\mathcal{E}_2$ with $(\mathtt{state}, Y)$, and uses its corresponding oracles to answer $\mathcal{E}_2$'s queries. Eventually, $\mathcal{E}_2$ terminates and $\mathcal{F}$ outputs whatever m' that $\mathcal{E}_2$ outputs. □

$\mathtt{Game}_{post}G_2$

1 Pick $\sigma, \sigma_1, \sigma_2, \sigma_3, \kappa_A, \tau_A, \kappa_B, \tau_B \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$. Compute $S = g^{\sigma}$, $h_1 = g^{\sigma_1}$, $h_2 = g^{\sigma_3}$, $\Gamma_A = g^{\sigma_2/\kappa_A}$, $\Upsilon_A = g^{\tau_A/\sigma_2}$, $\Gamma_B = g^{\sigma_2/\kappa_B}$, $\Upsilon_B = g^{\tau_B/\sigma_2}$, and $\mathcal{T}_{AB} = \langle \kappa_B/\tau_A, \kappa_A/\tau_B \rangle$. Set $\mathcal{L}_1 = \varnothing$ and $\mathcal{L}_2 = \varnothing$.

2 $m \xleftarrow{\$} \mathcal{M}$, $\lambda \xleftarrow{\$} \mathbb{Z}_p^*$, $x \xleftarrow{\$} \mathbb{Z}_p^*$, $\mu \xleftarrow{\$} \mathbb{Z}_p^*$, $\omega \xleftarrow{\$} \mathbb{Z}_p^*$, $\Omega = g^\omega$, $\Delta \xleftarrow{\$} \{0,1\}^{\log p}$, generate the challenge ciphertext $Y = \langle \mathcal{A}'_1, \cdots, \mathcal{A}'_6 \rangle$ as follows:

$$\begin{array}{ll} \mathcal{A}'_1 = g^x, & \mathcal{A}'_2 = h_1^{x \times m} h_2^x g^\lambda, \\ \mathcal{A}'_3 = \Gamma_A^x, & \mathcal{A}'_4 = \Upsilon_A^\lambda, \\ \mathcal{A}'_5 = g^\mu, & \mathcal{A}'_6 = \Delta \oplus m. \end{array}$$

Update $\mathcal{L}_1 = \mathcal{L}_1 \cup \{(\hat{e}(S, \Omega^\mu), \Delta)\}$ and $\mathcal{L}_2 = \mathcal{L}_2 \cup \{(\mathcal{A}'_1, \mathcal{A}'_2, \mathcal{A}'_3, \omega, \Omega)\}$.

3 $m' \leftarrow \mathcal{E}^{\mathcal{O}_{H_1}, \mathcal{O}_{H_2}, \mathcal{O}_{D_1}, \mathcal{O}_{D_2}}(PK, \mathcal{PT}_A, \mathcal{PT}_B, \mathcal{T}_{AB}, Y, *)$ with the following oracles.

- $\mathcal{O}_{H_1}$: For input element $T_i \in \mathbb{G}_T$, if there is a pair $(T_i, \delta_i) \in \mathcal{L}_1$, then return δ_i; if not, return a random value $\delta_i \xleftarrow{\$} \{0,1\}^{\log p}$, and append the tuple (T_i, δ_i) to $\mathcal{L}_1$.
- $\mathcal{O}_{H_2}$: For input tuple $(T_{i,1}, T_{i,2}, T_{i,3}) \in \mathbb{G}^3$, if there is an entry $(T_{i,1}, T_{i,2}, T_{i,3}, \omega_i, \Omega_i) \in \mathcal{L}_2$, then return Ω_i; if not, pick a random value $\omega_i \xleftarrow{\$} \mathbb{Z}_p^*$, return $\Omega_i = g^{\omega_i} \in \mathbb{G}$, and append $(T_{i,1}, T_{i,2}, T_{i,3}, \omega_i, \Omega_i)$ to $\mathcal{L}_2$.
- $\mathcal{O}_{D_1}$: For input ciphertext $\mathcal{A}_i = \langle \mathcal{A}_{i,1}, \cdots, \mathcal{A}_{i,6} \rangle$, invoke hash queries $\Omega_i \leftarrow \mathcal{O}_{H_2}(\mathcal{A}_{i,2}, \mathcal{A}_{i,3}, \mathcal{A}_{i,4})$ and $\delta_i \leftarrow \mathcal{O}_{H_1}(\hat{e}(\mathcal{A}_{i,5}, \Omega_i^\sigma))$, compute $r_i.A = \delta_i \oplus \mathcal{A}_{i,6}$. If $r_i.A$ satisfies Equality (1), then return $r_i.A$; otherwise return $\perp$.
- $\mathcal{O}_{D_2}$: For input ciphertext $\mathcal{B}_j = \langle \mathcal{B}_{j,1}, \cdots, \mathcal{B}_{j,6} \rangle$, invoke hash queries $\Omega_j \leftarrow \mathcal{O}_{H_2}(\mathcal{B}_{j,2}, \mathcal{B}_{j,3}, \mathcal{B}_{j,4})$ and $\delta_j \leftarrow \mathcal{O}_{H_1}(\hat{e}(\mathcal{B}_{j,5}, \Omega_j^\sigma))$, and compute $s_j.B = \delta_j \oplus \mathcal{B}_{j,6}$. If $s_j.B$ satisfies Equality (1), then return $s_j.B$; otherwise return $\perp$.

4 If $m' = m$, $\mathcal{E}$ wins the game.

Due to the idealness of random oracles $\mathcal{O}_{H_1}$ and $\mathcal{O}_{H_2}$, $\texttt{Game}_{post}G_2$ is identical to $\texttt{Game}_{post}G_1$. Thus, we have

LEMMA 4.3. $\Pr[\Psi_2] = \Pr[\Psi_1]$.

$\texttt{Game}_{post}G_3$

1 Pick $\sigma, \sigma_1, \sigma_2, \sigma_3, \kappa_A, \tau_A, \kappa_B, \tau_B \xleftarrow{\$} \mathbb{Z}_p^*$. Compute $S = g^\sigma$, $h_1 = g^{\sigma_1}$, $h_2 = g^{\sigma_3}$, $\Gamma_A = g^{\sigma_2/\kappa_A}$, $\Upsilon_A = g^{\tau_A/\sigma_2}$, $\Gamma_B = g^{\sigma_2/\kappa_B}$, $\Upsilon_B = g^{\tau_B/\sigma_2}$, and $\mathcal{T}_{AB} = \langle \kappa_B/\tau_A, \kappa_A/\tau_B \rangle$. Set $\mathcal{L}_1 = \varnothing$ and $\mathcal{L}_2 = \varnothing$.

2 $m \xleftarrow{\$} \mathcal{M}$, $\lambda \xleftarrow{\$} \mathbb{Z}_p^*$, $x \xleftarrow{\$} \mathbb{Z}_p^*$, $\mu \xleftarrow{\$} \mathbb{Z}_p^*$, $\omega \xleftarrow{\$} \mathbb{Z}_p^*$, $\Omega = g^\omega$, $\Theta \xleftarrow{\$} \{0,1\}^{\log p}$, generate the challenge ciphertext $Y = \langle \mathcal{A}'_1, \cdots, \mathcal{A}'_6 \rangle$ as follows:

$$\begin{array}{ll} \mathcal{A}'_1 = g^x, & \mathcal{A}'_2 = h_1^{x \times m} h_2^x g^\lambda, \\ \mathcal{A}'_3 = \Gamma_A^x, & \mathcal{A}'_4 = \Upsilon_A^\lambda, \\ \mathcal{A}'_5 = g^\mu, & \mathcal{A}'_6 = \Theta. \end{array}$$

Update $\mathcal{L}_1 = \mathcal{L}_1 \cup \{(\hat{e}(S, \Omega^\mu), \Theta \oplus m)\}$ and $\mathcal{L}_2 = \mathcal{L}_2 \cup \{(\mathcal{A}'_1, \mathcal{A}'_2, \mathcal{A}'_3, \omega, \Omega)\}$.

3 $m' \leftarrow \mathcal{E}^{\mathcal{O}_{H_1}, \mathcal{O}_{H_2}, \mathcal{O}_{D_1}, \mathcal{O}_{D_2}}(PK, \mathcal{PT}_A, \mathcal{PT}_B, \mathcal{T}_{AB}, Y, *)$ with the following oracles.

- $\mathcal{O}_{H_1}$: The same as in $\texttt{Game}_{post}G_2$ except that if the adversary $\mathcal{E}$ submits $\hat{e}(S, \Omega^\mu)$, then the security game aborts. Let this abort event be $\mathbb{E}_1$.
- $\mathcal{O}_{H_2}$: The same as in $\texttt{Game}_{post}G_2$.
- $\mathcal{O}_{D_1}$: The same as in $\texttt{Game}_{post}G_2$ except that if the adversary $\mathcal{E}$ submits $\langle \mathcal{A}'_1, \cdots, \mathcal{A}'_5, \mathcal{A}''_6 \rangle$ with $\mathcal{A}''_6 \neq \mathcal{A}'_6$, then return $\perp$.
- $\mathcal{O}_{D_2}$: The same as in $\texttt{Game}_{post}G_2$.

4 If $m' = m$, $\mathcal{E}$ wins the game.

In both $\texttt{Game}_{post}G_2$ and $\texttt{Game}_{post}G_3$, the element $\mathcal{A}'_6$ in the challenge ciphertext is a random value. Thus, the challenge ciphertext Y generated in $\texttt{Game}_{post}G_3$ is identically distributed to that in $\texttt{Game}_{post}G_2$. Hence, if the event $\mathbb{E}_1$ does not occur, then $\texttt{Game}_{post}G_3$ would be identical to $\texttt{Game}_{post}G_2$, that is,

$$|\Pr[\Psi_2] - \Pr[\Psi_3]| \leq \Pr[\mathbb{E}_1]. \tag{6}$$

In fact, event $\mathbb{E}_1$ implies the adversary is able to correctly decrypt Y to get m. We show that event $\mathbb{E}_1$ can only happen with negligible probability if the computational BDH assumption holds.

LEMMA 4.4. $\Pr[\mathbb{E}_1] \leq q_{H_1} \texttt{Adv}^{BDH} + \frac{q_{D_1}}{p}$, *where* q_{H_1} *denotes the number of hash queries to* H_1, *and* q_{D_1} *denotes the number of decryption queries for the records in* $\mathbf{R}$.

Proof. Suppose that event $\mathbb{E}_1$ happens with non-negligible probability, we can construct a PPT algorithm $\mathcal{I}$ that breaks the BDH assumption. At first, $\mathcal{I}$ is given BDH parameters $(\mathbb{G}, \mathbb{G}_T, p, \hat{e})$ and a BDH instance (g, g^a, g^b, g^c), with the goal of computing $\Lambda = \hat{e}(g,g)^{abc}$. $\mathcal{I}$ randomly picks $\sigma_1, \sigma_2, \sigma_3, \kappa_A, \tau_A, \kappa_B, \tau_B \xleftarrow{\$} \mathbb{Z}_p^*$, sets $S = g^a$, and computes $h_1 = g^{\sigma_1}$, $h_2 = g^{\sigma_3}$, $\Gamma_A = g^{\sigma_2/\kappa_A}$, $\Upsilon_A = g^{\tau_A/\sigma_2}$, $\Gamma_B = g^{\sigma_2/\kappa_B}$, $\Upsilon_B = g^{\tau_B/\sigma_2}$, and $\mathcal{T}_{AB} = \langle \kappa_B/\tau_A, \kappa_A/\tau_B \rangle$. Further, algorithm $\mathcal{I}$ randomly picks $m \xleftarrow{\$} \mathcal{M}$, $\lambda \xleftarrow{\$} \mathbb{Z}_p^*$, $x \xleftarrow{\$} \mathbb{Z}_p^*$, and $\Theta \xleftarrow{\$} \{0,1\}^{\log p}$, and generates the challenge ciphertext $Y = \langle \mathcal{A}'_1, \cdots, \mathcal{A}'_6 \rangle$ as follows:

$$\begin{array}{ll} \mathcal{A}'_1 = g^x, & \mathcal{A}'_2 = h_1^{x \times m} h_2^x g^\lambda, \\ \mathcal{A}'_3 = \Gamma_A^x, & \mathcal{A}'_4 = \Upsilon_A^\lambda, \\ \mathcal{A}'_5 = g^c, & \mathcal{A}'_6 = \Theta. \end{array}$$

Algorithm $\mathcal{I}$ initiates $\mathcal{L}_1$ as empty and adds the tuple $(\mathcal{A}'_1, \mathcal{A}'_2, \mathcal{A}'_3, \top, g^b)$ into $\mathcal{L}_2$, where $\top$ denotes an 'unknown' value. The challenge ciphertext Y has the same distribution as that in $\texttt{Game}_{post}G_3$. Algorithm $\mathcal{I}$ simulates the challenger of $\texttt{Game}_{post}G_3$ and invokes adversary $\mathcal{E}$ with inputs PK, $\mathcal{PT}_A$, $\mathcal{PT}_B$, $\mathcal{T}_{AB}$, Y, and the following oracles:

- $\mathcal{O}_{H_1}$: The same as in $\texttt{Game}_{post}G_3$.
- $\mathcal{O}_{H_2}$: The same as in $\texttt{Game}_{post}G_3$.

- $\mathcal{O}_{D_1}$: On input a ciphertext $\langle \mathcal{A}_{i,1}, \cdots, \mathcal{A}_{i,6} \rangle$, if $\mathcal{A}_{i,j} = \mathcal{A}'_j$ holds for $1 \le j \le 5$ yet $\mathcal{A}_{i,6} \neq \mathcal{A}'_6$, $\mathcal{I}$ returns $\perp$. Otherwise, algorithm $\mathcal{I}$ performs the following steps:
  - Query $\mathcal{O}_{H_2}$ with input $(\mathcal{A}_{i,2}, \mathcal{A}_{i,3}, \mathcal{A}_{i,4})$ and get (ω_i, Ω_i).
  - Search for an entry $(\hat{e}(\mathcal{A}_{i,5}, S^{\omega_i}), *)$ in $\mathcal{L}_1$. If some entry $(\hat{e}(\mathcal{A}_{i,5}, S^{\omega_i}), \Delta_i)$ exists, then compute $\Delta_i \oplus \mathcal{A}_{i,6}$. The decryption $\Delta_i \oplus \mathcal{A}_{i,6}$ is returned only when Equality (1) is satisfied, otherwise return $\perp$.
- $\mathcal{O}_{D_2}$: The same as in $\mathtt{Game}_{post}G_3$.

Let $\mathbb{E}_2$ denote the event that $\mathcal{E}$ queries $\mathcal{O}_{H_1}$ with $\hat{e}(g,g)^{abc}$. If event $\mathbb{E}_2$ does not happen by the end of the simulation, then $\mathcal{I}$ aborts with failure. To show that the decryption queries to $\mathcal{O}_{D_1}$ are simulated indistinguishably from $\mathtt{Game}_{post}G_3$, we analyze the decryption queries as follows:

- Case 1: $\hat{e}(\mathcal{A}_{i,5}, H_2(\mathcal{A}_{i,2}, \mathcal{A}_{i,3}, \mathcal{A}_{i,4})^a)$ has been queried to $\mathcal{O}_{H_1}$ before a decryption query for $\mathcal{A}_i = (\mathcal{A}_{i,1}, \cdots, \mathcal{A}_{i,6})$ is issued. In this case, $\mathcal{A}_{i,6}$ is uniquely determined. Thus, the decryption oracle $\mathcal{O}_{D_1}$ is perfectly simulated.
- Case 2: $\hat{e}(\mathcal{A}_{i,5}, H_2(\mathcal{A}_{i,2}, \mathcal{A}_{i,3}, \mathcal{A}_{i,4})^a)$ has not been queried to $\mathcal{O}_{H_1}$ before a decryption query for $\mathcal{A}_i = (\mathcal{A}_{i,1}, \cdots, \mathcal{A}_{i,6})$ is issued. In this case, $\mathcal{O}_{D_1}$ will output $\perp$. Thus, the simulations would fail if $\mathcal{A}_i$ is a valid ciphertext. Due to the idealness of $\mathcal{O}_{H_1}$, this happens with probability $1/p$.

Letting $\mathbb{E}_3$ denote the event that a valid ciphertext is rejected in the simulation, we have

$$\Pr[\mathbb{E}_3] \le \frac{q_{D_1}}{p}.$$

Thus, if event $\mathbb{E}_3$ does not happen, the simulations are identical to $\mathtt{Game}_{post}G_3$, which implies $\Pr[\mathbb{E}_2|\neg\mathbb{E}_3] = (1/q_{H_1})\Pr[\mathbb{E}_1]$. Therefore,

$$\begin{aligned}\Pr[\mathbb{E}_2] &= \Pr[\mathbb{E}_2|\mathbb{E}_3]\Pr[\mathbb{E}_3] + \Pr[\mathbb{E}_2|\neg\mathbb{E}_3]\Pr[\neg\mathbb{E}_3] \\ &\ge \Pr[\mathbb{E}_2|\neg\mathbb{E}_3]\Pr[\neg\mathbb{E}_3] \\ &= \frac{1}{q_{H_1}}\Pr[\mathbb{E}_1](1-\Pr[\mathbb{E}_3]) \\ &\ge \frac{1}{q_{H_1}}(\Pr[\mathbb{E}_1] - \Pr[\mathbb{E}_3]) \\ &\ge \frac{1}{q_{H_1}}\left(\Pr[\mathbb{E}_1] - \frac{q_{D_1}}{p}\right).\end{aligned}$$

Thus, we have

$$\mathtt{Adv}^{BDH}_{\mathcal{I}} \ge \frac{1}{q_{H_1}}\left(\Pr[\mathbb{E}_1] - \frac{q_{D_1}}{p}\right).$$

This completes the proof of Lemma 4.4. □

We continue to show that event Ψ_3 can only happen with negligible probability if the discrete logarithm problem is hard.

LEMMA 4.5. $\Pr[\Psi_3] \le \mathtt{Adv}^{DL}$.

Proof. Suppose that event Ψ_3 happens with non-negligible probability, we can construct a PPT algorithm $\mathcal{I}$ that breaks the DL assumption.

At first, $\mathcal{I}$ is given a DL instance $(g, g^a) \in \mathbb{G}^2$, with the goal of computing $a \in \mathbb{Z}_p^*$. Algorithm $\mathcal{I}$ randomly picks $\sigma, \sigma_1, \sigma_2, \sigma_3, \kappa_A, \tau_A, \kappa_B, \tau_B \xleftarrow{\$} \mathbb{Z}_p^*$, and computes $S = g^\sigma$, $h_1 = g^{\sigma_1}$, $h_2 = g^{\sigma_3}$, $\Gamma_A = g^{\sigma_2/\kappa_A}$, $\Upsilon_A = g^{\tau_A/\sigma_2}$, $\Gamma_B = g^{\sigma_2/\kappa_B}$, $\Upsilon_B = g^{\tau_B/\sigma_2}$, and $\mathcal{T}_{AB} = \langle \kappa_B/\tau_A, \kappa_A/\tau_B \rangle$. Further, algorithm $\mathcal{I}$ randomly picks $\lambda \xleftarrow{\$} \mathbb{Z}_p^*$, $x \xleftarrow{\$} \mathbb{Z}_p^*$, $\mu \xleftarrow{\$} \mathbb{Z}_p^*$, $\Theta \xleftarrow{\$} \{0,1\}^{\log p}$, and generates the challenge ciphertext $Y = \langle \mathcal{A}'_1, \cdots, \mathcal{A}'_6 \rangle$ as follows:

$$\begin{aligned}&\mathcal{A}'_1 = g^x, && \mathcal{A}'_2 = (g^a)^{\sigma_1 x} h_2^x g^\lambda = h_1^{ax} h_2^x g^\lambda, \\ &\mathcal{A}'_3 = \Gamma_A^x, && \mathcal{A}'_4 = \Upsilon_A^\lambda, \\ &\mathcal{A}'_5 = g^\mu, && \mathcal{A}'_6 = \Theta.\end{aligned}$$

Algorithm $\mathcal{I}$ also randomly picks $\omega \xleftarrow{\$} \mathbb{Z}_p^*$ and $\Omega = g^\omega$, and updates two initially empty query lists, that is, $\mathcal{I}$ inserts $(\hat{e}(S, \Omega^\mu), \top)$ into $\mathcal{L}_1$ and $(\mathcal{A}'_1, \mathcal{A}'_2, \mathcal{A}'_3, \omega, \Omega)$ into $\mathcal{L}_2$, where $\top$ denotes an 'unknown' value. Algorithm $\mathcal{I}$ simulates the challenger of $\mathtt{Game}_{post}G_3$ and invokes adversary $\mathcal{E}$ with inputs PK, $\mathcal{PT}_A$, $\mathcal{PT}_B$, $\mathcal{T}_{AB}$, and Y. At last, algorithm $\mathcal{I}$ outputs whatever $\mathcal{E}$ outputs. Thus, we have $\Pr[\Psi_3] \le \mathtt{Adv}^{DL}$. □

Combining Lemma 4.2 through Lemma 4.5, we have

$$\begin{aligned}\mathtt{Adv}^{ow-cca2}_{\mathcal{E},\mathsf{PKDE}_{post}}(\ell) &= \Pr[\Psi_0] \\ &\le \Pr[\Psi_1] + \frac{q_{D'}}{|\mathcal{M}|} \\ &= \Pr[\Psi_2] + \frac{q_{D'}}{|\mathcal{M}|} \\ &\le \Pr[\Psi_3] + \Pr[\mathbb{E}_1] + \frac{q_{D'}}{|\mathcal{M}|} \\ &\le \mathtt{Adv}^{DL} + q_{H_1}\mathtt{Adv}^{BDH} + \frac{q_{D_1}}{p} + \frac{q_{D'}}{|\mathcal{M}|}\end{aligned}$$

Since the DL assumption implies the BDH assumption, this concludes Theorem 4.3. □

4.2. Efficiency Analysis

We continue to analyze the computational complexity of the procedures in the proposed PKDE scheme. The analysis focuses on the most time-consuming operations, that is, exponentiations in cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$, and bilinear pairing $\hat{e}$. Their evaluation times are denoted by $\chi_{\mathbb{G}}$, $\chi_{\mathbb{G}_T}$, $\chi_{\hat{e}}$, respectively. Table 2 summarizes the computation costs of each procedure and makes a comparison with the most relevant existing work in [1]. In the table, the computation costs are analyzed for one evaluation, that is, one encryption, one decryption, and one join. In an encryption, both schemes require roughly the same computations – 8 exponentiations and 1 bilinear mapping. More specifically, in our PKDE scheme, the exponentiation for the last element (e.g., $\mathcal{A}_{i,6}$ or $\mathcal{B}_{j,6}$) of a ciphertext

TABLE 2. Computation cost for each procedure

	Pang and Ding [1]	This paper
`EncData`	$7\chi_{\mathbb{G}} + 1\chi_{\mathbb{G}_T} + 1\chi_{\hat{e}}$	$8\chi_{\mathbb{G}} + 1\chi_{\hat{e}}$
`DecData`	–	$2\chi_{\mathbb{G}} + 1\chi_{\mathbb{G}_T} + 4\chi_{\hat{e}}$
`Join`	$2\chi_{\mathbb{G}} + 5\chi_{\hat{e}}$	$2\chi_{\mathbb{G}} + 4\chi_{\hat{e}}$

TABLE 3. Comparison of element size

	Pang and Ding [1]	This paper
Ciphertext $\mathcal{A}_i$ or $\mathcal{B}_j$	$8\xi_{\mathbb{G}}$	$5\xi_{\mathbb{G}} + \log p$
Join query token $\mathcal{T}_{AB}$	$2\xi_{\mathbb{Z}}$	$2\xi_{\mathbb{Z}}$

is taken on $\mathbb{G}$. In fact, this exponentiation can also be moved to be performed on $\mathbb{G}_T$. In this way, both schemes enjoy the same encryption efficiency. For decryption, our construction requires 3 exponentiations and 4 bilinear mapping operations, whereas there is no efficient decryption procedure in [1]. Also, our scheme has a more efficient join mechanism that incurs one less bilinear mapping than that in [1]. Moreover, query token generation in both our PKDE scheme and [1] take only two multiplications, involving no time-consuming computation. As for storage cost, our PKDE scheme also outperforms [1] in that the latter one has larger ciphertext size as shown in Table 3, where $\xi_{\mathbb{G}}$ and $\xi_{\mathbb{Z}}$ denote the element size of $\mathbb{G}$ and $\mathbb{Z}_p$, respectively.

5. CONCLUSION

In this paper, we study the problem of privacy-preserving and controlled equijoin on databases that are hosted by an untrusted server. With the database owner's authorization, the server is able to perform an equijoin of two operand relations without decrypting their data or interacting with the owner. Moreover, the authorization cannot be abused to carry out any other join, including self-joins on the operand relations that would disclose statistical information even on records that are not in the equijoin result; in this sense, the equjoin is *controlled*. We introduce the notion of public key encryption for relational database with controlled equijoin (PKDE), and formalize its security model to capture two phases of data confidentiality. In the initial phase before any equijoin is performed, encrypted data should enjoy IND-CCA2 security. After an equijoin, the data would satisfy OW-CCA2 security. We propose a construction for our PKDE scheme and formally prove its security in our security model under the computational Bilinear Diffie-Hellman assumption. An efficiency analysis shows that our construction outperforms the only existing scheme (that however is formulated for a private key setting).

ACKNOWLEDGEMENTS

This article is based on research work supported by the Singapore National Research Foundation under NCR Award Number NRF2014NCR-NCR001-012.

REFERENCES

[1] Pang, H. and Ding, X. (2014) Privacy-preserving ad-hoc equi-join on outsourced data. *ACM Trans. Database Syst.*, **39**, 23:1–23:40.

[2] Carbunar, B. and Sion, R. (2012) Toward private joins on outsourced data. *IEEE Transactions on Knowledge and Data Engineering*, **24**, 1699–1710.

[3] Furukawa, J. and Isshiki, T. (2013) Controlled joining on encrypted relational database. *Pairing, Cologne, Germany*, pp. 46–64. Springer.

[4] Yang, G., Tan, C. H., Huang, Q., and Wong, D. S. (2010) Probabilistic public key encryption with equality test. *CT-RSA, San Francisco, CA, USA, March 1-5*, pp. 119–131. Springer.

[5] Lu, Y., Zhang, R., and Lin, D. (2012) Stronger security model for public-key encryption with equality test. *Pairing, Cologne, Germany*, pp. 65–82. Springer.

[6] Tang, Q. (2011) Towards public key encryption scheme supporting equality test with fine-grained authorization. *ACISP, Melbourne, Australia, July 11-13*, pp. 389–406. Springer.

[7] Tang, Q. (2012) Public key encryption supporting plaintext equality test and user-specified authorization. *Sec. and Commun. Netw.*, **5**, 1351–1362.

[8] Ma, S., Zhang, M., Huang, Q., and Yang, B. (2015) Public key encryption with delegated equality test in a multi-user setting. *The Computer Journal*, **58**, 986–1002.

[9] Huang, K., Tso, R., Chen, Y.-C., Rahman, S. M. M., Almogren, A., and Alamri, A. (2015) PKE-AET: Public key encryption with authorized equality test. *The Computer Journal*, **58**, 2686–2697.

[10] Ma, S., Huang, Q., Zhang, M., and Yang, B. (2015) Efficient public key encryption with equality test supporting flexible authorization. *IEEE Transactions on Information Forensics and Security*, **10**, 458–470.

[11] Lin, X.-J., Qu, H., and Zhang, X. (2016). Public Key Encryption Supporting Equality Test and Flexible Authorization without Bilinear Pairings. Cryptology ePrint Archive, Report 2016/277. http://eprint.iacr.org/2016/277.pdf.

[12] Cristofaro, E., Kim, J., and Tsudik, G. (2010) Linear-complexity private set intersection protocols secure in malicious model. *ASIACRYPT, Singapore, December 5-9*, pp. 213–231. Springer.

[13] Stefanov, E., Shi, E., and Song, D. (2012) Policy-enhanced private set intersection: Sharing information while enforcing privacy policies. *PKC, Darmstadt, Germany, May 21-23*, pp. 413–430. Springer.

[14] Kerschbaum, F. (2012) Collusion-resistant outsourcing of private set intersection. *SAC, Trento, Italy*, pp. 1451–1456. ACM.

[15] Kerschbaum, F. (2012) Outsourced private set intersection using homomorphic encryption. *ASIACCS, Seoul, Korea*, pp. 85–86. ACM.

[16] Boneh, D. and Franklin, M. (2003) Identity-based encryption from the weil pairing. *SIAM Journal on Computing*, **32**, 586–615.

[17] Boneh, D., Di Crescenzo, G., Ostrovsky, R., and Persiano, G. (2004) Public key encryption with keyword search. *EUROCRYPT, Interlaken, Switzerland, May 2-6*, pp. 506–522. Springer.