Frequency-hiding Dependency-preserving Encryption for Outsourced Databases

Boxiang Dong
Department of Computer Science
Montclair State University
Montclair, NJ 07047
bxdong7@gmail.com

Wendy Wang
Department of Computer Science
Stevens Institute of Technology
Hoboken, NJ 07030
Hui.Wang@stevens.edu

***Abstract*—The cloud paradigm enables users to outsource their data to computationally powerful third-party service providers for data management. Many data management tasks rely on the data dependency in the outsourced data. This raises an important issue of how the data owner can protect the sensitive information in the outsourced data while preserving the data dependency. In this paper, we consider *functional dependency* (FD), an important type of data dependency. Although simple deterministic encryption schemes can preserve FDs, they may be vulnerable against the frequency analysis attack. We design a frequency hiding, FD-preserving probabilistic encryption scheme, named F^2, that enables the service provider to discover the FDs from the encrypted dataset. We consider two attacks, namely the *frequency analysis (FA) attack* and the *FD-preserving chosen plaintext attack (FCPA)*, and show that the F^2 encryption scheme can defend against both attacks with formal provable guarantee. Our empirical study demonstrates the efficiency and effectiveness of F^2, as well as its security against both FA and FCPA attacks.**

ID	A	B	C
r_1	a_1	b_1	c_1
r_2	a_1	b_1	c_2
r_3	a_1	b_1	c_4
r_4	a_1	b_1	c_3
r_5	a_2	b_2	c_3
r_6	a_2	b_2	c_4

(a) Base table D ($FD : A \rightarrow B$, $A \not\rightarrow C$, $B \not\rightarrow C$)

ID	A	B	C
r_1	$\hat{a}_1^1$	$\hat{b}_1^1$	$\hat{c}_1^1$
r_2	$\hat{a}_1^2$	$\hat{b}_1^2$	$\hat{c}_2^1$
r_3	$\hat{a}_1^3$	$\hat{b}_1^3$	$\hat{c}_4^2$
r_4	$\hat{a}_1^4$	$\hat{b}_1^4$	$\hat{c}_3^1$
r_5	$\hat{a}_2^1$	$\hat{b}_2^1$	$\hat{c}_3^2$
r_6	$\hat{a}_2^2$	$\hat{b}_2^2$	$\hat{c}_4^1$

(b) $\hat{D}_1$: probabilistic encryption on A, B, C individually (ciphertext frequency $= 1$) Introduce false positive FDs $A \rightarrow C$, $B \rightarrow C$

ID	A	B	C
r_1	$\hat{a}_1^1$	$\hat{b}_1^1$	$\hat{c}_1^1$
r_2	$\hat{a}_1^1$	$\hat{b}_1^2$	$\hat{c}_2^1$
r_3	$\hat{a}_1^2$	$\hat{b}_1^3$	$\hat{c}_4^2$
r_4	$\hat{a}_1^2$	$\hat{b}_1^4$	$\hat{c}_3^1$
r_5	$\hat{a}_2^1$	$\hat{b}_2^1$	$\hat{c}_3^2$
r_6	$\hat{a}_2^2$	$\hat{b}_2^2$	$\hat{c}_4^1$

(c) $\hat{D}_2$: probabilistic encryption on A, B, C individually (ciphertext frequency $>= 1$) $A \rightarrow B$ is not preserved;

ID	A	B	C
r_1	$\hat{a}_1^1$	$\hat{b}_1^1$	$\hat{c}_1^1$
r_2	$\hat{a}_1^1$	$\hat{b}_1^1$	$\hat{c}_2^1$
r_3	$\hat{a}_1^2$	$\hat{b}_1^2$	$\hat{c}_4^2$
r_4	$\hat{a}_1^2$	$\hat{b}_1^2$	$\hat{c}_3^1$
r_5	$\hat{a}_2^1$	$\hat{b}_2^1$	$\hat{c}_3^2$
r_6	$\hat{a}_2^2$	$\hat{b}_2^2$	$\hat{c}_4^1$

(d) $\hat{D}_3$: probabilistic encryption on attribute set $\{A, B\}$ and $\{C\}$ (ciphertext frequency $>= 1$) FD-preserving No false positive FDs

Fig. 1: Examples of (good and bad) probabilistic encryption schemes

I. Introduction

With the fast growth of data volume, the sheer size of today's data sets is increasingly crossing the petabyte barrier, which far exceeds the capacity of an average business computer. In-house solutions may be expensive due to the purchase of software, hardware, and staffing cost of in-house resources to administer the software. Lately, due to the advent of cloud computing and its model for IT services based on the Internet and big data centers, a new model has emerged to give companies a cheaper alternative to in-house solutions: the users outsource their data management needs to a third-party service provider. Outsourcing of data and computing services is becoming commonplace and essential.

Many outsourced data management applications rely on data dependency in the outsourced data. A typical type of data dependency is *functional dependency* (FD). Informally, a $FD : A \rightarrow B$ constraint indicates that an attribute set A uniquely determines an attribute set B. For example, the FD *Zipcode$\rightarrow$City* indicates that all tuples of the same *Zipcode* values always have the same *City* values. FDs serve a wide range of data applications, for example, improving schema quality through normalization [1], [2], and improving data quality in data cleaning [3]. Therefore, to support these applications in the outsourcing paradigm, it is vital that FDs are well preserved in the outsourced data.

Outsourcing data to a potentially untrusted third-party service provider (server) raises several security issues. One of the issues is to protect the sensitive information in the outsourced data. The data confidentiality problem is traditionally addressed by means of encryption [4]. Our goal is to design efficient *FD-preserving* data encryption methods that enable the FDs in the original dataset to be kept in the encrypted dataset. A naive method is to apply a simple deterministic encryption scheme (i.e. the same plaintext values are always encrypted as the same ciphertext for a given key) on the attributes with FDs. For instance, consider the base table D in Figure 1 (a) that has a FD: F: $A \rightarrow B$. Suppose that D is encrypted by applying a deterministic encryption scheme on each individual cell of D. Apparently F is preserved in the encrypted dataset. However, this naive method has drawbacks. One of the main drawbacks is that since the encryption preserves the frequency distribution, the deterministic encryption scheme is vulnerable against the frequency analysis attack [5]. The attacker can easily map the ciphertext to the plaintext values based on their frequency.

Challenges of designing probabilistic FD-preserving encryption. A straightforward solution to defending against the frequency analysis attack is to use the probabilistic encryption schemes (i.e., the same plaintext values are encrypted as different ciphertext values) instead of the deterministic encryption schemes, aiming to hide frequency. However, adapting the existing probabilistic encryption schemes to be FD-preserving is far more than trivial. The following example will show that applying the probabilistic encryption scheme in a careless way may introduce severe utility concerns; it may either destroy FDs completely or introduce *false positive* FDs (i.e., the FDs that do not exist in D).

Example 1.1: Consider the table D shown in Figure 1 (a). Figure 1 (b) shows an instance $\hat{D}_1$ by applying the standard *probabilistic encryption scheme on the attributes A, B and C individually. We use the notation $\hat{a}_i^j$ ($\hat{b}_i^j$, $\hat{c}_i^j$, resp.) as the j-th unique ciphertext value of a_i (b_i, c_i, resp.) by the probabilistic encryption scheme. Due to the probabilistic encryption, all ciphertext values in $\hat{D}_1$ are distinct (i.e., frequency is one). Apparently the frequency analysis attack fails on $\hat{D}_1$, since regardless of the frequency distribution of the plaintext values, the frequency distribution of the ciphertext values is always flattened. Also it is FD-preserving; the FD $A \rightarrow B$ holds on $\hat{D}_1$. However, there are quite a few false positive FDs (e.g., $A \rightarrow C$ and $B \rightarrow C$) in $\hat{D}_1$. If we apply a non-standard probabilistic encryption scheme on individual attributes that allows the ciphertext values of the same plaintext value to be identical (e.g., attribute A value of records r_1 and r_2 in $\hat{D}_2$ of Figure 1 (c)), it still can defend against the frequency analysis attack. However, the FD $A \rightarrow B$ does not hold on $\hat{D}_2$ anymore, as the same ciphertext value $\hat{a}_1^1$ of attribute A is associated with two different cipher values $\hat{b}_1^1$ and $\hat{b}_1^2$ of attribute B. A correct FD-preserving encryption scheme $\hat{D}_3$ that is free of false positive FDs is shown in Figure 1 (d). It applies the probabilistic encryption scheme on the attribute set $\{A, B\}$. The four instances of (a_1, b_1) are encrypted as $(\hat{a}_1^1, \hat{b}_1^1)$ and $(\hat{a}_1^2, \hat{b}_1^2)$. The FD $A \rightarrow B$ still holds on $\hat{D}_3$. There is no false positive FDs.*

Example 1.1 shows that the probabilistic encryption scheme, if it is applied on individual attributes, fails to be a FD-preserving solution, regardless of the frequency of the ciphertext values. This raises the first challenge of designing FD-preserving probabilistic encryption schemes: what is the appropriate encryption granularity on which the probabilistic encryption is applied? Example 1.1 shows that the if the attributes of FDs (e.g., $\{A, B\}$ in Example 1.1) are treated together as a *super attribute*, applying the probabilistic encryption scheme on the super attribute can preserve FDs. However, finding the attributes of FDs is non-trivial. In the centralized setting, the FDs may be well-defined (e.g., for data schema normalization and refinement). However, in the outsourcing setting, the data owner may not have the knowledge and expertise of FDs. Furthermore, since finding the FD attributes needs intensive computational efforts [6], the data owner may lack of computational resources to discover the FDs by herself, and has to rely on the server for FD discovery. Without the knowledge of FD attributes for encryption, the data owner has to identify the appropriate attributes (as the super attribute) for encryption by herself, while requiring that the cost of doing so should be much cheaper than FD discovery.

The second challenge of designing the FD-preserving probabilistic encryption schemes is that as shown in Example 1.1 (Figure 1 (b)), directly applying the standard probabilistic encryption algorithms may introduce a large amounts of false positive FDs, since all ciphertext values are highly likely distinct. The question is, how to adapt the standard probabilistic encryption algorithms to be FD-preserving while ensuring that the encryption does not introduce any false positive FD.

The third challenge is that FD-preserving encryption may introduce new inference attacks. FD-preserving encryption is a type of property-preserving encryption [7]. The frequency analysis attack is probably the most well-known inference attack against the property-preserving encryption [5]. Besides the frequency analysis attack, the attacker may utilize the preserved FDs to break the encryption. For example, FD-preserving encryption may not be able to provide indistinguishability under the chosen-plaintext attack (IND-CPA). The challenges include formal definition of IND-CPA with the presence of FDs, and the security guarantee analysis of the FD-preserving encryption scheme against IND-CPA. None of the existing work on the inference attacks (e.g., [8], [5]) against the property-preserving encryption have considered the FD-preserving encryption.

Contributions. To address these challenges, we design F^2, a frequency-hiding, FD-preserving encryption scheme based on probabilistic encryption. In specific, we make the following contributions.

(i) F^2 allows the data owner to encrypt the data without the awareness of any FD in the original dataset. To defend against the frequency analysis (FA) attack on the encrypted data, we apply the *probabilistic encryption* in a way that the frequency distribution of the ciphertext values is always flattened. The complexity of F^2 is much cheaper than FD discovery locally. Furthermore, F^2 guarantees to preserve all FDs in the original dataset, while avoids to introduce any false positive FDs.

(ii) We formally define *FD-preserving chosen plaintext attack* (FCPA) on F^2. We also formally define: (1) α-security model against the FA attack, which requires that the probability that any single ciphertext can be mapped to its plaintext based on frequency should not exceed α, a user-defined threshold; and (2) indistinguishability against FD-preserving chosen plaintext attack (IND-FCPA). We prove that: (1) F^2 can provide α-security against the FA attack; and (2) F^2 can provide indistinguishability against FCPA when the outsourced dataset contains large FD-based partitions.

(iii) We evaluate the performance of F^2 on both real and synthetic datasets. The experiment results show that F^2 can encrypt large datasets efficiently with high security guarantee and 100% accuracy of FD discovery. It also shows that applying F^2 and outsourcing is three orders of magnitude faster than finding FDs locally.

Applications. F^2 is the first FD-preserving algorithm that can defend against both the frequency analysis and FD-preserving chosen plaintext attacks with provable security guarantees. F^2 can benefit those applications that heavily rely on FDs as a key component, for example, data schema refinement and normalization [1], [2], data cleaning [9], [3], and schema-mapping and data exchange [10]. We believe that F^2 represents a significant contribution towards the goal of reaching the full maturity of data outsourcing systems (e.g., database-as-a-service [11] and data-cleaning-as-a-service systems [12]). For example, by preserving FDs, a number of FD-based data quality metrics, e.g., the g3 error [6], will remain intact. This enables to perform data cleaning, especially the detection and repair of inconsistencies, on the encrypted data.

The rest of this paper is organized as follows. Section II describes the preliminaries. Section III presents the security definition. Section IV discusses the details of our encryption scheme. Section V presents the security analysis. Section VI evaluates the performance of our approach. Related work is introduced in Section VII. Section VIII concludes the paper.

II. Preliminaries

A. Outsourcing Setting

We consider the outsourcing framework that contains two parties: the data owner and the service provider (server). The data owner has a private relational table D that consists of m attributes and n records. We use $r[X]$ to specify the value of record r on the attribute(s) X. To protect the private information in D, the data owner encrypts D to $\hat{D}$, and sends $\hat{D}$ to the server. The server discovers the dependency constraints of $\hat{D}$. These constraints can be utilized for data cleaning [3] or data schema refinement [1]. In principle, database encryption may be performed at various levels of granularity. However, the encryption at the coarse levels such as attribute and record levels may disable the dependency discovery at the server side. Therefore, in this paper, we consider encryption *at cell level* (i.e., each data cell is encrypted individually). Given the high complexity of FD discovery [6], we assume that the data owner is *not* aware of any FD in D before she encrypts D. As the FDs are taken as the utility, our aim is to preserve the FDs in $\hat{D}$ without the data owner's awareness of any FD.

B. Functional Dependency

In this paper, we consider *functional dependency* (FD) as the data dependency that the server aims to discover. Formally, given a relation R, there is a FD between a set of attributes X and Y in R (denoted as $X \rightarrow Y$) if for any pair of records r_1, r_2 in R, if $r_1[X] = r_2[X]$, then $r_1[Y] = r_2[Y]$. We use $LHS(F)$ ($RHS(F)$, resp.) to denote the attributes at the left-hand (right-hand, resp.) side of the FD F. For any FD $F : X \rightarrow Y$ such that $Y \subseteq X$, F is considered as *trivial*. In this paper, we only consider non-trivial FDs. It is well known that for any FD $F : X \rightarrow Y$ such that Y contains more than one attribute, F can be decomposed to multiple functional dependency rules, each having a single attribute at the right-hand side. Therefore, for the following discussions, we assume that the FDs only contain one single attribute at RHS.

III. Attacks and Security Model

In this paper, we consider the *curious-but-honest* server, i,e., it follows the outsourcing protocols honestly (i.e., no cheating on storage and computational results), but it is curious to extract additional information from the received dataset. Since the server is potentially untrusted, the data owner sends the encrypted data to the server. The data owner considers the true identity of every cipher value as the sensitive information which should be protected.

We consider two adversarial attacks: (1) the *frequency analysis attack* that is based on frequency distribution information, and (2) the *chosen plaintext attack* that is based on the FD-preserving property of F^2, that try to break the encryption on the outsourced data.

A. Frequency Analysis Attack (FA)

Frequency analysis attack is one of the most well-known example of an inference attack to break classical ciphers [5]. In this paper, we assume that the attacker may possess the frequency distribution knowledge of data values at the cell level of D. In reality, the attacker may possess approximate knowledge of the value frequency in D. However, in order to make the analysis robust, we adopt the conservative assumption that the attacker knows the exact frequency of every plain value in D, and tries to break the encryption scheme by utilizing such frequency knowledge. In particular, let $\mathcal{P}$ and $\mathcal{E}$ be the plaintext and ciphertext values. Consider a ciphertext value e that is randomly chosen from $\mathcal{E}$. Let $freq_{\mathcal{E}}(e)$ be the frequency of e, and $freq(\mathcal{P})$ be the frequency distribution of $\mathcal{P}$. Then e, $freq_{\mathcal{E}}(e)$ and $freq(\mathcal{P})$ are given to the adversary $\mathcal{A}^{freq}$. We formally define the following adversarial experiment Exp_{Π}^{FA} on an encryption scheme Π.

$$\begin{aligned} &\textbf{Experiment} \quad Exp_{\Pi}^{FA}() \\ &p' \leftarrow A^{freq_{\mathcal{E}}(e), freq(\mathcal{P})} \\ &Return\ 1\ if\ p' = Decrypt(k, e) \\ &Return\ 0\ otherwise \end{aligned}$$

Intuitively, Exp_{Π}^{FA} returns 1 if the attacker can correctly infer the plaintext of e based on the frequency distribution of plaintext and ciphertext values. Let $Exp_{\Pi}^{FA}(A)$ denote the output of the adversary by FA against the encryption scheme Π. We define the FA *advantage* as

$$Adv_{\Pi}^{FA}(A) = Prob(Exp_{\Pi}^{FA}(A) = 1).$$

Based on the FA advantage, we formally define *α-security*, the security model against FA.

Definition 3.1: **[α-Security against FA]** An encryption scheme Π is α-secure against FA if for every adversary A it holds that $Adv_{\Pi}^{FA}(A) \leq \alpha$, where $\alpha \in (0, 1]$ is user specified.

Intuitively, the smaller α is, the stronger security that Π is against the FA. In this paper, we aim at designing an encryption scheme that provides α-security. We mainly focus on the frequency distribution of individual attributes. The inter-attribute correlation-based attack [13] is out of the scope of this paper.

B. FD-preserving Chosen Plaintext Attack (FCPA)

Indistinguishability under chosen plaintext attack (IND-CPA) is a property of many encryption schemes. Intuitively, if a cryptosystem possesses IND-CPA (and thus semantically secure under chosen plaintext attack), then an adversary will be unable to distinguish pairs of ciphertexts based on the message they encrypt. Since F^2 preserves FDs, it cannot provide indistinguishability under the chosen-plaintext attack (IND-CPA). Therefore, we define a new security model named *indistinguishability against FD-preserving chosen plaintext attack* (IND-FCPA). An IND-FCPA encryption scheme reveals the FD, but everything else remains semantically secure. We first introduce the FCPA adversary. The FCPA adversary is designed in the similar fashion as the left-or-right game [14] for symmetric encryption schemes. Intuitively, the adversary chooses two plaintext datasets m_0 and m_1. We require both m_0 and m_1 have the same size and FDs. The challenger encrypts one of them at random. The adversary tries to guess which one was encrypted by making a sequence of queries $(m_0^1, m_1^1), \ldots, (m_0^q, m_1^q)$ on the encrypted dataset. Specifically, let $LR(.,.,b)$ denote the function that on inputs m_0, m_1 returns $m_b (b \in \{0,1\})$. Given a dataset D with a FD $F : X \rightarrow Y$, for a given symmetric encryption scheme Π, $b \in \{0,1\}$, and a polynomial-time adversary A, consider the following experiment:

$$\begin{aligned} &\textbf{Experiment} \quad Exp_{\Pi}^{IND-FCPA-b}() \\ &\quad d \leftarrow A^{Encrypt(k,LR(.,.,b)),F} \\ &\qquad\qquad Return\ d \end{aligned}$$

Let $Exp_{\Pi}^{IND-FCPA-1}(A)$ ($Exp_{\Pi}^{IND-FCPA-0}(A)$, resp.) denote that the adversary A outputs m_1 (m_0, resp.). We define the IND-FCPA advantage as

$$\begin{aligned} Adv_{\Pi}^{FCPA}(A) = & \ |Prob(Exp_{\Pi}^{IND-FCPA-1}(A) = 1) \\ - & \ Prob(Exp_{\Pi}^{IND-FCPA-0}(A) = 1)|. \end{aligned}$$

Now we are ready to define IND-FCPA.

Definition 3.2: **[Indistinguishability against FD-preserving Chosen Plaintext Attack (IND-FCPA)]** An encryption scheme Π is indistinguishable against FD-preserving chosen plaintext attack if for any polynomial-time adversary A, it holds that the IND-FCPA advantage is negligible in λ, i.e., $Adv_{\Pi}^{FCPA}(A) = negl(\lambda)$, where λ is a pre-defined security parameter.

Intuitively, IND-FCPA requires that for any polynomial-time adversary that has oracle access to the encryption function, it cannot differentiate the ciphertext value of a pair of plaintext values with a non-negligible probability.

IV. FD-PRESERVING ENCRYPTION

The key to designing a FD-preserving encryption algorithm is to first identify the set of attributes that is treated as the super attribute on which the probabilistic encryption scheme is applied. We have the following theorem to show that for any FD F, if the probabilistic encryption scheme is applied on the attribute set that includes both $LHS(F)$ and $RHS(F)$ (Fig 1 (d)), F is still preserved.

Theorem 4.1: Given a dataset D and any FD F of D, let the attribute set $\mathcal{A}$ be the attribute sets on which the probabilistic encryption scheme is applied on, and let $\hat{D}$ be the encryption result. Then F always holds in $\hat{D}$ if $LHS(F) \cup RHS(F) \subseteq \mathcal{A}$.

Proof: Assume that there is a FD: $A \rightarrow B$ which holds in D but not in $\hat{D}$. It must be true that there exists at least one pair of records r_1, r_2 such that $r_1[A] = r_2[A]$ and $r_1[B] = r_2[B]$ in D, while in $\hat{D}$, $\hat{r_1}[A] = \hat{r_2}[A]$ but $\hat{r_1}[B] \neq \hat{r_2}[B]$. Consider the attribute set $\mathcal{A}$ s.t. $LHS(F) \cup RHS(F) \subseteq \mathcal{A}$ for encryption, there are two cases. First, if r_1 and r_2 are taken as the same instance, then $\hat{r_1}$ and $\hat{r_2}$ have the same value in every attribute. It cannot happen as $\hat{r_1}[B] \neq \hat{r_2}[B]$. Second, if r_1 and r_2 are taken as different instances, then it must be true that $\hat{r_1}[A] \neq \hat{r_2}[A]$ and $\hat{r_1}[B] \neq \hat{r_2}[B]$. So $\hat{r_1}$ and $\hat{r_2}$ cannot break the FD: $A \rightarrow B$ in $\hat{D}$. ■

Our FD-preserving probabilistic encryption method consists of four steps: (1) identifying maximum attribute sets as the *super attribute* for encryption; (2) splitting-and-scaling; (3) conflict resolution; and (4) eliminating false positive FDs. We explain the details of these four steps in the following subsections.

A. Step 1: Identifying Maximum Attribute Sets

Theorem 4.1 states that the set of attributes on which the probabilistic encryption scheme is applied should contain all attributes in FDs. Apparently the set of all attributes of D satisfies the requirement. However, it is not a good solution. Let $|\sigma_{\mathcal{A}=r[\mathcal{A}]}(D)|$ denote the number of records in D that have the same value as $r[\mathcal{A}]$, for a specific record r and a set of attributes $\mathcal{A}$. There is no need for a probabilistic encryption scheme, due to the reason that $f = |\sigma_{\mathcal{A}=r[\mathcal{A}]}(D)|$ is more likely to be 1 when $\mathcal{A}$ contains more attributes. Now the main challenge is to decide the appropriate attribute set $\mathcal{A}$ that will be considered as the *super attribute* on which the probabilistic encryption scheme is be applied on, assuming that the data owner is not aware of the existence of any FD. To address this challenge, we define the *maximum attribute set* on which there exists at least one instance whose frequency is greater than 1. Formally,

Definition 4.1: **[Maximum Attribute Set (MAS)]** Given a dataset D, an attribute set $\mathcal{A}$ of D is a *maximum attribute set* MAS if: (1) there exists at least an instance a of $\mathcal{A}$ such that $|\sigma_{\mathcal{A}=a}(D)| > 1$; and (2) for any attribute set $\mathcal{A}'$ of D such that $\mathcal{A} \subseteq \mathcal{A}'$, there does not exist an instance a' of $\mathcal{A}'$ s.t. $|\sigma_{\mathcal{A}'=a'}(D)| > 1$.

Our goal is to design the algorithm that finds *all* $MASs$ of a given dataset D. Note that the problem of finding $MASs$ is not equivalent to finding FDs. For instance, consider the base table D in Figure 1 (a). Its $MASs$ is $\{A, B\}$ and $\{C\}$. But its FD is $A \rightarrow B$. In general, given a set of MASs $\mathcal{M}$ and a set of FDs $\mathcal{F}$, for each FD $F \in \mathcal{F}$, there always exists at least an MAS $M \in \mathcal{M}$ such that $(LHS(F) \cup RHS(F)) \subseteq M$. But it is possible that a MAS does not contain any FD.

In general, finding $MASs$ is quite challenging given the exponential number of attribute combinations to check. We found out that our MAS is equivalent to the *maximal non-unique column combination* [15]. We adapt the $Ducc$ algorithm [15] to find $MASs$. The complexity of $Ducc$ is decided by the solution set size but not the number of attributes. Due to the space limit, we omit the details of the algorithm here. For each discovered MAS, we find its *partitions* [6]. We say two tuples r and r' are *equivalent* with respect to a set of attributes X if $r[X] = r'[X]$.

Definition 4.2: **[Equivalence Class (EC) and Partitions]** [6] The *equivalence class* (EC) of a tuple r with respect to an attribute set X, denoted as r_X, is defined as $r_X = \{r' | r[A] = r'[A], \forall A \in X\}$. The *size* of r_X is defined as the number of tuples in r_X, and the *representative value* of r_X is defined as $r[X]$. The set $\pi_X = \{r_X | r \in D\}$ is defined as a *partition* of D under the attribute set X. That is, π_X is a collection of disjoint sets (ECs) of tuples, such that each set has a unique representative value of a set of attributes X, and the union of the sets equals D.

As an example, consider the dataset D in Figure 1 (a), $\pi_{\{A,B\}}$ consists of two ECs whose representative values are (a_1, b_1) and (a_2, b_2). Apparently, a MAS is a maximal attribute set whose partitions contain at least one equivalence class whose size is more than 1.

B. Step 2: Splitting-and-Scaling Encryption

After the $MASs$ and their partitions are discovered, we design two steps to apply the probabilistic encryption scheme on the partitions: (1) grouping of ECs, and (2) splitting and scaling on the EC groups. Next, we explain the details.

1) Step 2.1. Grouping of Equivalence Classes: To provide α-security, we group the ECs in the way that each equivalence class belongs to one single group, and each group contains at least $k \geq [\frac{1}{\alpha}]$ ECs, where α is the threshold for α-security. We use ECG for the *equivalence class group* in short.

We have two requirements for the construction of $ECGs$. First, we prefer to put ECs of close sizes into the same group, so that we can minimize the number of new tuples added by the next splitting & scaling step (Section IV-B2). Second, we do not allow any two ECs in the same ECG to have the same value on any attribute of MAS. Formally,

Definition 4.3: **[Collision of ECs]** Given a MAS M and two equivalence classes C_i and C_j of M, we say C_i and C_j have *collision* if there exists at least one attribute $A \in M$ such that $C_i[A] = C_j[A]$.

For security reason, we require that all ECs in the same ECG should be collision-free (more details in Section V).

To construct $ECGs$ that satisfy the aforementioned two requirements, first, we sort the equivalence classes by their sizes in ascending order. Second, we group the collision-free ECs with the closest size into the same ECG, until the number of non-collisional ECs in the ECG reaches $k = [\frac{1}{\alpha}]$. It is possible that for some ECGs, the number of ECs that can be found is less than the required $k = [\frac{1}{\alpha}]$. In this case, we add *fake* collision-free ECs to achieve the size k requirement. The fake ECs only consist of the values that do not exist in the original dataset. The size of these fake ECs is set as the minimum size of the ECs in the same ECG. We must note that the server cannot distinguish the fake values from real ones.This is because both true and fake values are encrypted before outsourcing.

EC	ID	A	B
C_1	$\{r_2, r_3\}$	a_1	b_1
C_2	$\{r_1\}$	a_2	b_1
C_3	$\{r_4\}$	a_3	b_1
C_4	$\{r_5\}$	a_4	b_2
C_5	$\{r_6\}$	a_5	b_2

(a) ECs in of $MAS = \{AB\}$ $(\varpi = 2, \alpha = 0.5)$

ECG	EC	ID	A	B
ECG_1	C_1	$\{r_2, r_3\}$	$\hat{a}_1^1$	$\hat{b}_1^1$
	C_4	$\{r_5, r_7\}$	$\hat{a}_4^1$	$\hat{b}_2^1$
ECG_2	C_2	$\{r_1\}$	$\hat{a}_2^1$	$\hat{b}_1^2$
	C_5	$\{r_6\}$	$\hat{a}_5^1$	$\hat{b}_2^2$
ECG_3	C_3	$\{r_4\}$	$\hat{a}_3^1$	$\hat{b}_1^3$
	C_6	$\{r_8\}$	$\hat{a}_6^1$	$\hat{b}_3^1$

(b) ECGs after $S\&S$

Fig. 2: An example of splitting-and-scaling

Example 4.1: Consider the MAS $M = \{A, B\}$ and its five ECs shown in Figure 2 (a). Assume it is required to meet $\frac{1}{2}$-security (i.e., $\alpha = \frac{1}{2}$). The three ECGs are $ECG_1 = \{C_1, C_4\}$, $ECG_2 = \{C_2, C_5\}$, and $ECG_3 = \{C_2, C_6\}$, where C_6 is an artificial EC. All the ECGs only have collision-free ECs, and each ECG contains at least two ECs. Note that C_1 and C_2 cannot be put into the same ECG as they share the same value b_1, similarly for C_1 and C_3.

2) Step 2.2. Splitting-and-Scaling (S&S): This step consists of two phases, *splitting* and *scaling*. In particular, consider an ECG $\overline{\mathcal{C}} = \{C_1, \dots, C_x\}$, in which each EC C_i is of size f_i $(1 \leq i \leq x)$. By *splitting*, for each C_i, its f_i (identical) plaintext values are encrypted to ϖ unique ciphertext values, each of frequency $[\frac{f_i}{\varpi}]$, where ϖ is the split factor whose value is specified by the user. The *scaling* phase is applied after splitting. By scaling, all the ciphertext values reach the same frequency by adding additional copies up to $[\frac{f_{max}}{\varpi}]$, where f_{max} is the maximum size of all ECs in $\overline{\mathcal{C}}$. After applying $S\&S$, all ciphertext values in the same ECG are of the same frequency.

We illustrate how $S\&S$ works by using Figure 2 (b). Before $S\&S$, ECG_1 contains C_1 of frequency 2, and C_4 of frequency 1. By the $S\&S$ step, the record r_5 of C_4 is duplicated (as record r_7), ensuring that C_1 and C_4 have the same frequency. Note that before $S\&S$, a_1 is the only plaintext value on attribute A whose frequency is greater than 1. But after $S\&S$, there exist two ciphertext values ($\hat{a}_1^1$ and $\hat{a}_4^1$) whose frequency is greater than 1. Thus, the adversary's probability of breaking the encryption of a_1 based on the frequency distribution is $\frac{1}{2}$.

The splitting procedure can be implemented by the probabilistic encryption. For any plaintext value p, it is encrypted as $e =< r, F_k(r) \oplus p >$, where r is a random string of length λ, F is a pseudo random function, k is the key, and $\oplus$ is the *XOR* operation. The splits of the same EC can easily be generated by using different random values r. In order to decrypt a ciphertext $e =< r, s >$, we recover p by calculating $p = F_k(r) \oplus s$. The pseudo random function can be constructed from a standard block cipher using the CBC-MAC construction [16]. For security concerns, we require that the plaintext values that appear in different $ECGs$ are never encrypted as the same ciphertext value (more details in Section V). Note that for all the records in the same EC, we can obtain its encrypted values by only encrypting the representative value of the equivalence class once.

It is not necessary that every EC must be split. Our aim is to find a subset of given ECs whose split will result in the minimal amounts of additional copies added by the scaling phase. In particular, given an ECG $\overline{\mathcal{C}} = \{C_1, \dots, C_k\}$ in which ECs are sorted by their sizes in ascending order (i.e., C_k has the largest size), we aim to find the *split point* j of $\overline{\mathcal{C}}$ such that each EC in $\{C_1, \dots, C_{j-1}\}$ is not split but each EC in $\{C_j, \dots, C_k\}$ is split. We call j the *split point*. Next, we discuss how to find the *optimal* split point that delivers the minimal amounts of additional copies by the scaling phase. There are two cases: (1) the size of C_k is still the largest after the split; and (2) the size of C_k is not the largest anymore, while the size of C_{j-1} (i.e., the EC of the largest size among all ECs that are not split) becomes the largest. We discuss these two cases below.

Case 1: $[\frac{f_k}{\varpi}] \geq f_{j-1}$, i.e., the split of C_k still has the largest frequency within the group. In this case, the total number of copies added by the scaling step is

$$R_1 = \sum_{i=1}^{j-1}([\frac{f_k}{\varpi}] - f_i) + \sum_{i=j}^{k}(f_k - f_i).$$

It can be easily inferred that when $j = max\{j | f_{j-1} \leq [\frac{f_k}{\varpi}]\}$, R_1 is minimized.

Case 2: $[\frac{f_k}{\varpi}] < f_{j-1}$, which means that C_{j-1} enjoys the largest frequency after splitting. For this case, the number of duplicates that is to be added is:

$$R_2 = \sum_{i=1}^{j-1}(f_{j-1} - f_i) + \varpi \sum_{i=j}^{k}(f_{j-1} - [\frac{f_i}{\varpi}]).$$

R_2 is not a linear function of j. Thus we define $j_{max} = max\{j|[\frac{f_k}{\varpi}] > f_{j-1}\}$. We try all $j \in [j_{max}, k]$ and return j that delivers the minimal R_2.

For any given ECG, the complexity of finding its optimal split point is $O(|ECG|)$. In practice, the optimal split point j is close to the ECs of the largest frequency (i.e., few split is needed). In general, the complexity of the $S\&S$ step is $O(t^2)$, where t is the number of equivalence classes of D. As $t = O(nq)$, the complexity is $O(n^2q^2)$, where n is the number of tuples in D, and q is the number of $MASs$.

C. Step 3: Conflict Resolution

ID	A	B
r_1	$\hat{a}_2^1$	$\hat{b}_1^2$
r_2	$\hat{a}_1^1$	$\hat{b}_1^1$
r_3	$\hat{a}_1^1$	$\hat{b}_1^1$
r_4	$\hat{a}_3^1$	$\hat{b}_1^3$
r_5	$\hat{a}_4^1$	$\hat{b}_2^1$
r_6	$\hat{a}_5^1$	$\hat{b}_2^2$

(a) $Enc_{\{A,B\}}(D)$

ID	B	C
r_1	$\hat{b}_1^2$	$\hat{c}_1^1$
r_2	$\hat{b}_1^2$	$\hat{c}_1^1$
r_3	$\hat{b}_1^3$	$\hat{c}_2^1$
r_4	$\hat{b}_1^3$	$\hat{c}_2^1$
r_5	$\hat{b}_2^1$	$\hat{c}_2^2$
r_6	$\hat{b}_2^2$	$\hat{c}_3^1$

(b) $Enc_{\{B,C\}}(D)$

ID	A	B	C
r_1	$\hat{a}_2^1$	$\hat{b}_1^2$	$\hat{c}_1^1$
r_2	$\hat{a}_1^1$	$\hat{b}_1^1$	$\hat{c}_1^2$
r_3	$\hat{a}_1^1$	$\hat{b}_1^1$	$\hat{c}_2^3$
r_4	$\hat{a}_3^1$	$\hat{b}_1^3$	$\hat{c}_2^1$
r_5	$\hat{a}_4^1$	$\hat{b}_2^1$	$\hat{c}_2^2$
r_6	$\hat{a}_5^1$	$\hat{b}_2^2$	$\hat{c}_3^1$
r_7	$\hat{a}_1^2$	$\hat{b}_1^2$	$\hat{c}_1^1$
r_8	$\hat{a}_1^3$	$\hat{b}_1^3$	$\hat{c}_2^1$

(c) $\hat{D}$: Encryption by conflict resolution

Fig. 3: An example of conflict resolution of two overlapping $MASs$

So far the grouping and splitting & scaling steps are applied on one single MAS. In practice, it is possible that there exist multiple $MASs$. The problem is that, applying grouping and splitting & scaling separately on each MAS may lead to conflicts. For instance, consider the example in Figure 3, in which Figure 3 (a) and (b) show the encryption on two MASs $\{A, B\}$ and $\{B, C\}$ individually. The conflict appears at tuples r_2 and r_3 on attribute B. For example, $r_2[B]$ is encrypted as $\hat{b}_1^1$, while it is encrypted as $\hat{b}_1^2$.

There are two possible conflicts:
(1) *Type-1. Conflicts due to scaling*: there exist tuples that are scaled by one MAS but not so by another MAS; and
(2) *Type-2. Conflicts due to shared attributes*: there exist tuples whose value on attribute(s) Z are encrypted differently by multiple $MASs$, where Z is the overlap of these $MASs$.

It initially seems true that there may exist the conflicts due to splitting too; some tuples are required to be split according to one MAS but not by another. But our further analysis shows that such conflicts only exist for overlapping $MASs$, in particular, the type-2 conflicts. Therefore, dealing with type-2 conflicts covers the conflicts due to splitting.

The aim of the conflict resolution step is to synchronize the encryption of all $MASs$. Given two $MASs$ of the attribute sets X and Y, we say these two $MASs$ *overlap* if X and Y overlap at at least one attribute. Otherwise, we say the two $MASs$ are non-overlapping. Next, we discuss the details of the conflict resolution for non-overlapping $MASs$ (Section IV-C1) and overlapping $MASs$ (Section IV-C2).

1) Non-overlapping $MASs$: Given two $MASs$ of attribute sets X and Y such that X and Y do not overlap, only the type-1 conflicts (i.e., conflicts due to scaling) are possible. WLOG we assume a tuple r is required to be scaled by applying scaling on the ECG of r_X but not on any ECG of r_Y. The challenge is that simply scaling of r makes the ECG of r_Y fails to have homogenized frequency anymore. To handle this type of conflicts, first, we execute the grouping and splitting & scaling over π_X and π_Y independently. Next, we look for any tuple r such that r is required to have ℓ ($\ell > 1$) copies to be inserted by splitting & scaling over π_X but free of split and scaling over π_Y. For such r, we modify the values of the ℓ copies of tuple r, ensuring that for each copy r', $r'[X] = r[X]$ while $r'[Y] \neq r[Y]$. To avoid collision between $ECGs$, we require that no $r'[Y]$ value exists in the original dataset D. By doing this, we ensure that the $ECGs$ of both r_X and r_Y still achieve the homogenized frequency distribution. Furthermore, by assigning new and unique values to each copy on the attribute set Y, we ensure that $MASs$ are well preserved (i.e., both $MASs$ X and Y do not change to be $X \cup Y$). The complexity of dealing with type-1 conflicts is $O(n'q)$, where n' is the number of the tuples that have conflicts due to scaling, and q is the number of $MASs$.

2) Overlapping $MASs$: When $MASs$ overlap, both types of conflicts are possible. The type-1 conflicts can be handled in the same way as for non-overlapping $MASs$ (Section IV-C1). In the following discussion, we mainly focus on how to deal with the type-2 conflicts (i.e., the conflicts due to shared attributes). We start our discussion from two overlapping $MASs$. Then we extend to the case of more than two overlapping $MASs$.

Two overlapping MASs. We say two ECs $C_i \in \pi_X$ and $C_j \in \pi_Y$ are *conflicting* if C_i and C_j share at least one tuple. We have the following theorem to show that conflicting ECs never share more than one tuple.

Theorem 4.2: Given two overlapping $MASs$ X and Y, for any pair of ECs $C_i \in \pi_X$ and $C_j \in \pi_Y$, $|C_i \cap C_j| \leq 1$.

The correctness of Theorem 4.2 is straightforward: if $|C_i \cap C_j| > 1$, there must exist at least one equivalence class of the partition $\pi_{X \cup Y}$ whose size is greater than 1. Then $X \cup Y$ should be a MAS instead of X and Y. Theorem 4.2 ensures the efficiency of the conflict resolution, as it does not need to handle a large number of tuples.

A naive method to fix type-2 conflicts is to assign the same ciphertext value to the shared attributes of the two conflicting ECs. By re-visiting the example in Figure 3 (a), if we follow the naive solution to resolve the conflict appears at tuples r_2 and r_3 on attribute B, only one value is picked for tuples r_2 and r_3 on attribute B. This scheme is incorrect as the FD $F : A \rightarrow B$ does not hold anymore.

We design a robust method to resolve the type-2 conflicts for two overlapping $MASs$. Given two overlapping $MASs$ X and Y, let $Z = X \cap Y$, for any tuple r, let $r^X[Z]$ and $r^Y[Z]$ ($r^X[Z] \neq r^Y[Z]$) be the value constructed by encryption over X and Y independently. We use $X - Z$ ($Y - Z$, resp.) to denote the attributes that appear in X (Y, resp.) but not Z. Then we construct two tuples r_1 and r_2:

- r_1: $r_1[X - Z] = r[X - Z]$, $r_1[Y - Z] = v_X$, and $r_1[Z] =$

	EC	B	C	f
ECG_1	C_1	b_1	c_1	2
	C_2	b_2	c_2	1
ECG_2	C_3	b_1	c_2	2
	C_4	b_2	c_3	1

(a) Base table D ($B \rightarrow C$ does not hold)

	B	C	f
ECG_1	$\hat{b}_1^2$	$\hat{c}_1^1$	2
	$\hat{b}_2^1$	$\hat{c}_2^2$	2
ECG_2	$\hat{b}_1^3$	$\hat{c}_2^1$	2
	$\hat{b}_2^2$	$\hat{c}_3^1$	2

(b) $\hat{D}$: encryption by Step 1 - 3 ($B \rightarrow C$ becomes false positive)

A	B	C	f
$\hat{a}_6$	$\hat{b}_3$	$\hat{c}_4$	1
$\hat{a}_7$	$\hat{b}_3$	$\hat{c}_5$	1
$\hat{a}_8$	$\hat{b}_4$	$\hat{c}_6$	1
$\hat{a}_9$	$\hat{b}_4$	$\hat{c}_7$	1

(c) Constructed ΔD to remove false positive FD $B \rightarrow C$

Fig. 4: An example of eliminating false positive FDs

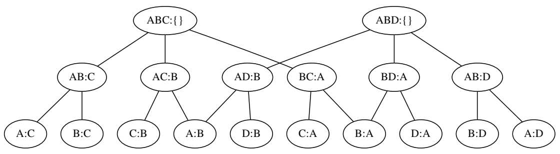


Fig. 5: An example of FD lattice

$r^X[Z]$;

- r_2: $r_2[X - Z] = v_Y$, $r_2[Y - Z] = r[Y - Z]$, and $r_2[Z] = r^Y[Z]$.

where v_X and v_Y are two values that do not exist in D. Note that both $X - Z$ and $Y - Z$ can be sets of attributes, thus v_X and v_Y can be set of values. Tuples r_1 and r_2 replace tuple r in $\hat{D}$. As an example, consider the tables in Figure 3 (a) and (b), the encryption after conflict resolution is shown in Figure 3 (c). Tuple r_2 and r_7 in Figure 3 (c) are the two records constructed for the conflict resolution of r_2 in Figure 3 (a) and (b). Our conflict resolution method guarantees that the ciphertext values of each ECG of X and Y are of homogenized frequency.

More than Two Overlapping MASs. When there are more than two overlapping $MASs$, one way is to execute the encryption scheme that deals with two overlapping $MASs$ repeatedly for every two overlapping $MASs$. This raises the question in which order the $MASs$ should be processed. We have the following theorem to show that indeed the conflict resolution is insensitive to the order of $MASs$.

Theorem 4.3: Given a dataset D that contains a set of overlapping $MASs$ M, let $\hat{D}_1$ and $\hat{D}_2$ be the datasets after executing conflict resolution in two different orders of M, then $|\hat{D}_1| = |\hat{D}_2|$.

We omit the proof of Theorem 4.3 due to the space limitation. It can be found in the full version of our paper [17].

Since the order of $MASs$ does not affect the encryption, we pick the overlapping MAS pairs randomly. For each overlapping MAS pair, we apply the encryption scheme for two overlapping $MASs$. We repeat this procedure until all $MASs$ are processed.

D. Step 4. Eliminating False Positive FDs

Before we discuss the details of this step, we first define false positive FDs. Given the dataset D and its encrypted version $\hat{D}$, we say the FD F is a *false positive* if F does not hold in D but holds in $\hat{D}$. We observe that the encryption scheme constructed by Step 1 - 3 may lead to false positive FDs. Next, we show an example of false positive FDs.

Example 4.2: Consider the table D in Figure 4 (a) generated from the dataset in Figure 3 (a). Apparently the FD $F : B \rightarrow C$ does not exist in D, as the two ECs C_1 and C_3 have collisions. However, in the encrypted dataset $\hat{D}$ constructed by Step 1 - 3 (Figure 4 (b)), since no split of any two ECs have collision anymore, $F : B \rightarrow C$ holds in $\hat{D}$.

Indeed, we have the following theorem to show that free of collision among ECs is the necessary and sufficient conditions for the existence of FDs.

Theorem 4.4: For any MAS X, there must exist a FD on X if and only if for any two ECs C_i and C_j of π_X, C_i and C_j do not have collision.

Following Theorem 4.4, the fact that Step 1 - 3 construct only collision-free ECs may lead to a large number of false positive FDs, which hurts the accuracy of dependency discovering by the server.

The key idea to eliminate the false positive FDs is to restore the collision within the ECs. Note that eliminating the false positive FD $F : X \rightarrow Y$ naturally leads to the elimination of all false positive FDs $F' : X' \rightarrow Y$ such that $X' \subseteq X$. Therefore, we only consider eliminating the *maximum false positive FDs* whose LHS is not a subset of LHS of any other false positive FDs.

We use the *FD lattice* to help restore the ECs with collision and eliminate false positive FDs. The lattice is constructed in a top-down fashion. Each MAS corresponds to a level-1 node in the lattice. We denote it in the format $M : \{\}$, where M is the MAS that the node corresponds to. The level-2 nodes in the lattice are constructed as following. For each level-1 node N, it has a set of children nodes (i.e., level-2 nodes) of format $X : Y$, where Y corresponds to a single attribute of D, and $X = M - \{Y\}$, where M is the MAS that node N corresponds to. Starting from level 2, for each node $X : Y$ at level ℓ ($\ell \geq 2$), it has a set of children nodes of format $X' : Y'$, where $Y' = Y$, and $X' \subset X$, with $|X|' = |X| - 1$ (i.e., X' is the largest subset of X). Each node at the bottom of the lattice is of format $X : Y$, where both X and Y are 1-attribute sets. An example of the FD lattice is shown in Figure 5.

Based on the FD lattice, the data owner eliminates the false positive FDs by the following procedure. Initially all nodes in the lattice are marked as "un-checked". Starting from the second level of lattice, for each node N (in the format $X : Y$), the data owner checks whether there exists at least two ECs $C_i, C_j \in \pi_M$ such that $C_i[X] = C_j[X]$ but $C_i[Y] \neq C_j[Y]$, where M is the MAS that N's parent node corresponds to. If it does, then the data owner does the following two steps. First, the data owner inserts $k = \lceil \frac{1}{\alpha} \rceil$ artificial record pairs $\{P_1, \dots, P_k\}$, where α is the given threshold for α-security. The reason why we insert k such pairs will be explained in Section V. Each P_i consists of two records r_i^1 and r_i^2:

- $r_i^1 : r_i^1[X] = x_i$, $r_i^1[Y] = a_i^1$, and $r_i^1[MAS - X - \{Y\}] = v_i^1$;
- $r_i^2 : r_i^2[X] = x_i$, $r_i^2[Y] = a_i^2$, and $r_i^2[MAS - X - \{Y\}] = v_i^2$.

where x_i, a_i^1, a_i^2, v_i^1 and v_i^2 are artificial values that do not exist in $\hat{D}$ constructed by the previous steps. We require that $a_i^1 \neq a_i^2$, and $v_i^1 \neq v_i^2$. We also require that all artificial records are of frequency one. Second, the data owner marks the current node and all of its descendants in the lattice as "checked" (i.e., the current node is identified as a maximum false positive FD). Otherwise (i.e., the ECs do not have collisions), the data owner simply marks the current node as "checked". After checking all the nodes at the level ℓ, the data owner moves to the level $\ell+1$ and applies the above operation on the "unchecked" lattice nodes. The data owner repeats the procedure until all nodes in the lattice are marked as "checked".

Let ΔD be the artificial records that are constructed by the above iterative procedure. It is easy to see that for any FD $X \rightarrow Y$ that *does not* hold in D, there always exist two ECs C'_i and C'_j in ΔD, where $C'_i[X] = C'_j[X]$ but $C'_i[Y] \neq C'_j[Y]$. This makes the FD $X \rightarrow Y$ that does not hold in D fails to hold in $\hat{D}$ too.

Example 4.3: To continue our Example 4.2, we show how to construct ΔD. It is straightforward that the MAS $\{B, C\}$ contains the ECs that have collision (e.g. C_1 and C_3 in Figure 4 (a)). Assume $\alpha = 1/2$. Then ΔD (shown in Figure 4 (c)) consists of two pairs of artificial tuples, each pair consisting of two records of the same value on attribute B but not on C. The false positive FD $F : B \rightarrow C$ does not exist in $\hat{D} + \Delta D$ any more.

The time complexity of eliminating false positive FDs for a single MAS M is $O(2^{|M|}t)$, where $|M|$ is the number of attributes in M, and t is the number of ECs of M. In our experiments, we observe $t << n$, where n is the number of records in D. For instance, on a benchmark dataset of 15 million records, the average value of t is $11,828$. Also, the experiment results on all three datasets show that at most $|M| = \frac{m}{2}$. With the existence of $q > 1$ MASs, the time complexity is $O(\sum_{i=1}^{q} 2^{|M_i|}t_i)$, where t_i is the number of equivalence classes in M_i, and $|M_i|$ is the number of attributes of MAS M_i. Considering that $t_i << n$, the total complexity is comparable to $O(nm^2)$.

We have the following theorem to show that the FDs in $\hat{D}$ and D are the same.

Theorem 4.5: Given the dataset D, let $\hat{D}$ be the dataset after applying Step 1 - 4 of F^2 on D, then: (1) any FD of D also hold on $\hat{D}$; and (2) any FD F that does not hold in D does not hold in $\hat{D}$ either.

Due to the space limit, we omit the proof of Theorem 4.5. It can be found in the full version of our paper [17].

V. Security Analysis

In this section, we analyze the security of F^2 against both the frequency analysis attack (FA) and the FD-preserving chosen plaintext attack (FCPA).

A. Frequency Analysis Attack

Frequency analysis (FA) attack is the most basic and well-known inference attack. It is formally defined in [5]. We refer the readers to [5] for more details. For any $e \in \mathcal{E}$ be a ciphertext value, let $G(e) = \{p | p \in \mathcal{P}, freq_{\mathcal{P}}(p) = freq_{\mathcal{E}}(e)\}$ be the set of distinct plaintext values having the same frequency as e. It has been shown in [18] that for any adversary A and any ciphertext value e, the advantage of the frequency analysis attack is $Adv^{FA}_{F^2}(A) = \frac{1}{|G(e)|}$, where $|G(e)|$ is the size of $G(e)$. In other words, the size of $G(e)$ determines the advantage $Adv^{FA}_{F^2}(A)$.

Apparently, the scaling step of F^2 ensures that all the equivalence classes in the same ECG have the same frequency. Hence, for any encrypted equivalence class EC', there are at least $|ECG|$ plaintext ECs having the same frequency. Recall that we do not allow any two equivalence classes in the same ECG to have the same value on any attribute. Therefore, for any attribute A, a ECG contains k distinct plaintext values on A, where k is the size of ECG. Thus for any $e \in \mathcal{E}$, it is guaranteed that $|G(e)| = k$. As $k \geq [\frac{1}{\alpha}]$, it is guaranteed that $G(e) \geq [\frac{1}{\alpha}]$. In this way, we have:

$$Adv^{FA}_{F^2}(A) \leq \alpha.$$

Thus F^2 provides α-security against the FA attack.

In our empirical study, we implemented the FA attack in [5], and compared the attack accuracy of F^2 with three different encryption schemes: (1) frequency-hiding order-preserving encryption [19]; (2) AES, a well-known symmetric encryption algorithm; and (3) Pallier, a probabilistic asymmetric algorithm. More details can be found in Section VI.

B. FD-preserving Chosen Plaintext Attack (FCPA)

Given a dataset D with a FD $X \rightarrow Y$, consider the following FCPA adversary A against F^2. Let $LR(.,.,b)$ denote the function that on inputs m_0, m_1 returns $m_b (b \in \{0,1\})$.

$$\begin{aligned}
&\textbf{Experiment } Exp^{IND-FCPA}_{F^2}(k, LR(.,.,b)) \\
&\qquad r_1, r_2, r_3, r_4 \in D \\
&\qquad r_1[X,Y] = r_3[X,Y] = (x_1, y_1) \\
&\qquad r_2[X,Y] = (x_2, y_2) \\
&\qquad r_4[X,Y] = (x_3, y_3) \\
&\qquad c_1 \leftarrow Encrypt(k, LR(r_1, r_2, b))[X,Y] \\
&\qquad c_2 \leftarrow Encrypt(k, LR(r_3, r_4, b))[X,Y] \\
&\qquad Return\ 0\ if\ c_1 = c_2 \\
&\qquad Return\ 1\ otherwise
\end{aligned}$$

Intuitively, if both r_1 and r_3 (i.e., the left side of (r_1, r_2) and (r_3, r_4)) are encrypted, the adversary can infer that $b = 0$ as F^2 is FD-preserving. Otherwise $b = 1$. Next, we analyze the security guarantee of F^2 against the FCPA adversary A. Let $Exp(A)$ denote the output of experiment.

When $b = 1$, $c_1 = Encrypt(k, r_2)[X,Y] = Encrypt(k, (x_2, y_2))$, and $c_2 = Encrypt(k, (x_3, y_3))$. Thus $Prob(Exp(A) = 1 | b = 1) = Prob(Encrypt(k, (x_2, y_2)) \neq Encrypt(k, (x_3, y_3))) = 1 - negl(\lambda)$. This is because the probability that two different plaintext values x_2, y_2 and (x_3, y_3) are encrypted to the same ciphertext value is negligible.

When $b = 0$, $c_1 = Encrypt(k, r_1)[X,Y]$, and $c_2 = Encrypt(k, r_2)[X,Y]$. According to F^2, it is possible that r_1 and r_2 have different ciphertext values on $[X,Y]$, even though they have the same plaintext values. Next, we infer the probability that $c_1 = c_2$. Let M be the MAS that include both X and Y. We define a set of equivalence classes $EQ = \{eq \in \pi_M | eq[X,Y] = (x_1, y_1)\}$, and g as the number of equivalence classes in the partition of M whose attribute values on $[X,Y]$ is (x_1, y_1). It must be true that $r_1 \in eq_i \in EQ$ and $r_2 \in eq_j \in EQ$, where $1 \leq i, j \leq g$. Based on the relationship between i and j, we have the following cases for the adversary to output 1 (i.e., $c_1 = c_2$).

Case 1: $i \neq j$**.** When r_1 and r_2 belong to different equivalence classes of π_M, their ciphertext values must be different. This is because according to Step 2.1 in Section IV, eq_i and eq_j must fall into different ECGs, as there exist collision between them. We also require that the same plaintext values that appear in different ECGs are never encrypted as the same ciphertext value. Therefore, $Prob(c_1 \neq c_2 | i \neq j) = 1$. As $Prob(i \neq j) = 1 - Prob(i = j) = 1 - \frac{1}{g}$, $Prob(Exp(A) = 1, i \neq j | b = 0) = 1 - \frac{1}{g}$.

Case 2: $i = j$**.** When r_1 and r_2 belong to the same equivalence class of π_M, it is still possible for them to have different ciphertext values. Let the equivalence class be eq. Based on our *Splitting-and-scaling* scheme, when eq is split and when r_1 and r_2 belong to different split copies, r_1 and r_2 are encrypted to be different ciphertext values. In other words, $Prob(Exp(A) = 1, i = j | b = 0) = Prob(E_1, E_2) * \frac{1}{g}$, where E_1 is the event that eq is split, and E_2 is the event that r_1 and r_2 belong to different split copies. Let y be the split point in the ECG, we have $Prob(E_1) = \frac{y}{k}$, where $k = [\frac{1}{\alpha}]$. This is because in the ECG, we pick y out of k equivalence classes to split. Let ϖ be the split factor. Then $Prob(E2|E_1) = 1 - \frac{1}{\varpi}$, since the probability that r_1 and r_2 are encrypted as the same split copy is $\frac{1}{\varpi}$. Therefore, we have $Prob(Exp(A) = 1, i = j | b = 0) = Prob(E_1, E_2) = Prob(E2|E1) * Prob(E1) = (1 - \frac{1}{\varpi}) * \frac{y}{gk}$.

Based on the two cases, $Prob(Exp(A) = 1 | b = 0) = 1 - \frac{1}{g} + \frac{(\varpi - 1)y}{gk\varpi}$. Thus, the adversary's advantage on F^2 is

$$Adv_{F^2}^{FCPA}(A) = \frac{\varpi k - (\varpi - 1)y}{g\varpi k} - negl(\lambda).$$

In practice, we find that in most cases, y is close to 0, so

$$Adv_{F^2}^{FCPA}(A) \approx \frac{1}{g}.$$

In general, when g is large (i.e., the equivalent classes are of large size), $Adv_{F^2}^{FCPA}(A)$ is negligible, and thus F^2 can provide IND-FCPA. In our experiments on real-world datasets, g can be 5,000,000 for a dataset that contains 15 million records. When g is small, we can always reach IND-FCPA by inserting artificial records to make g sufficiently large. It is possible that the number of artificial records may be significant; but that is the price we pay for the security.

VI. Experiments

In this section, we discuss our experiment results and provide the analysis of our observations.

A. Setup

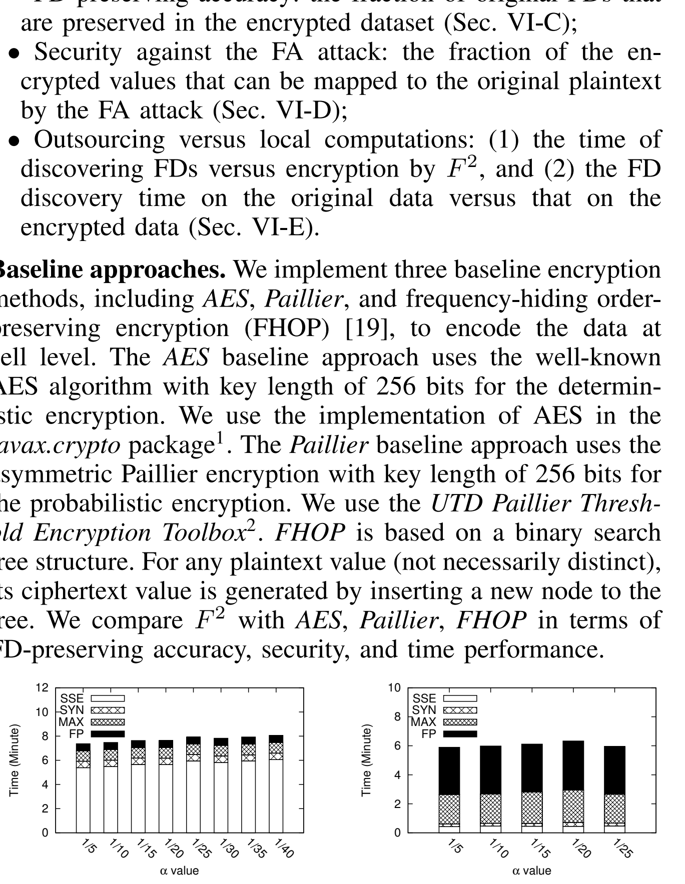
Computer environment. We implement our algorithm in Java. All the experiments are executed on a PC with 2.5GHz i7 CPU and 60GB memory running Linux.

Datasets. We execute our algorithm on the *Customer* dataset from TPC-C benchmark, the *Orders* dataset from TPC-H benchmark, and one synthetic dataset. The *Orders* dataset contains 9 attributes and 1.5 million records (size 1.67GB). *Orders* dataset contains nine maximal attribute sets $MASs$. All $MASs$ overlap pairwise. Each MAS contains either four or five attributes. The *Customer* dataset includes 21 attributes and 960K records (size 282MB). There are fifteen MASs in *Customer* dataset. The size of these $MASs$ (i.e., the number of attributes) ranges from nine to twelve. All MASs overlap pairwise. The synthetic dataset contains 7 attributes and 4 million records (size 224MB). The synthetic dataset has two MASs, one of three attributes, while the other of six attributes. The two $MASs$ overlap at one attribute.

Evaluation. We evaluate the efficiency and practicality of our encryption scheme according to the following criteria:

- Encryption time: the time for the data owner to encrypt the dataset (Sec. VI-B);
- Space overhead: the amounts of artificial records added by F^2 (In our experiments, the number of artificial records never exceeds 12% of the original data size. Due to the space limitation, we only report the statistics in Figure 9 and omit the discussion of this set of results);
- FD-preserving accuracy: the fraction of original FDs that are preserved in the encrypted dataset (Sec. VI-C);
- Security against the FA attack: the fraction of the encrypted values that can be mapped to the original plaintext by the FA attack (Sec. VI-D);
- Outsourcing versus local computations: (1) the time of discovering FDs versus encryption by F^2, and (2) the FD discovery time on the original data versus that on the encrypted data (Sec. VI-E).

Baseline approaches. We implement three baseline encryption methods, including *AES*, *Paillier*, and frequency-hiding order-preserving encryption (FHOP) [19], to encode the data at cell level. The *AES* baseline approach uses the well-known AES algorithm with key length of 256 bits for the deterministic encryption. We use the implementation of AES in the *javax.crypto* package[1]. The *Paillier* baseline approach uses the asymmetric Paillier encryption with key length of 256 bits for the probabilistic encryption. We use the *UTD Paillier Threshold Encryption Toolbox*[2]. *FHOP* is based on a binary search tree structure. For any plaintext value (not necessarily distinct), its ciphertext value is generated by inserting a new node to the tree. We compare F^2 with *AES*, *Paillier*, *FHOP* in terms of FD-preserving accuracy, security, and time performance.



(a) Synthetic dataset (53MB) (b) Orders dataset (0.325GB)

Fig. 6: Time performance for various α

B. Encryption Time

In this section, we measure the time performance of F^2 to encrypt the dataset. First, we evaluate the impact of security threshold α on the running time. We measure the time of our four steps of the algorithm: (1) finding maximal attribute sets (MAX), (2) splitting-and-scaling encryption (SSE), (3) conflict resolution (SYN), and (4) eliminating false positive FDs (FP), individually. We show our results on both *Orders* and the synthetic datasets in Figure 6. First, we observe that for both datasets, the time performance does not change much

[1]https://docs.oracle.com/javase/7/docs/api/javax/crypto/package-summary.html

[2]http://cs.utdallas.edu/dspl/cgi-bin/pailliertoolbox/.

with the decrease of α value. This is because the running time of the MAX, SYN and FP steps is independent on α. In particular, the time of finding $MASs$ stays stable with the change of α values, as its complexity relies on the data size, not α. Similarly, the time performance of FP step is stable with various α values, as its complexity is only dependent on the data size and data schema. The time performance of SYN step does not vary with α value because we only need to synchronize the encryption on the original records, without the worry about the injected records. It is worth noting that on the *Orders* dataset, the time took by the SYN step is negligible. This is because the SYN step leads to only 24 artificial records on the *Orders* dataset (of size 0.325GB). Second, the time of SSE step grows for both datasets when α decreases (i.e., tighter security guarantee). This is because smaller α value requires larger number of artificial equivalence classes to form ECGs of the desired size. This addresses the trade-off between security and time performance. We also observe that the increase in time performance is insignificant. For example, even for *Orders* dataset of size 0.325GB, when α is decreased from 0.2 to 0.04, the execution time of the SSE step only increases by 2.5 seconds. This shows that F^2 enables to achieve higher security with small additional time cost on large datasets. We also observe that different steps dominate the time performance on different datasets: the SSE step takes most of the time on the synthetic dataset, while the MAX and FP steps take the most time on the *Orders* dataset. This is because the average number of ECs of all the MASs in the synthetic dataset is much larger than that of the *Orders* dataset (128,512 v.s. 1003). Due to the quadratic complexity of the SSE step with regard to the number of ECs, the SSE step consumes most of the time on the synthetic dataset.

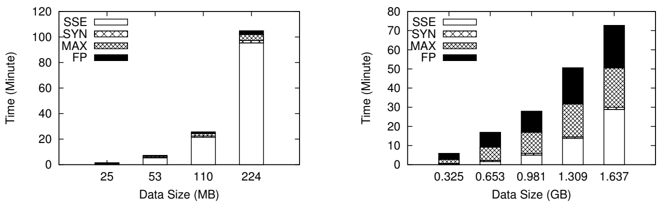

(a) Synthetic dataset (α=0.25) (b) Orders dataset (α=0.2)

Fig. 7: Time performance for various data sizes

Second, to analyze the scalability of our approach, we measure the time performance for various data sizes. The results are shown in Figure 7. It is not surprising that the time performance of all the four steps increases with the data size. We also notice that, on both datasets, the time performance of the SSE step is not linear to the data size. This is due to the fact that the time complexity of the SSE step is quadratic to the number of ECs. With the increase of the data size, the average number of ECs increases linearly on the synthetic dataset and super-linearly on the *Orders* dataset. Thus we observe the non-linear relationship between time performance of SSE step and data size. On the synthetic dataset, the dominant time factor is always the time of the SSE step. This is because of the large average number of ECs (can be as large as 1 million) and the quadratic complexity of the SSE step. In contrast, the average number of ECs is at most $11,828$ on the *Orders* dataset. So even though the synthetic dataset has fewer and smaller MASs than the *Orders* dataset, the vast difference in the number of ECs makes the SSE step takes the majority of the time performance on the synthetic dataset.

To sum up the observations above, our F^2 approach can be applied on the large datasets with high security guarantee, especially for the datasets that have few number of ECs on the MASs. For instance, it takes around 30 minutes for F^2 to encrypt the *Orders* dataset of size 1GB with security guarantee of $\alpha = 0.2$.

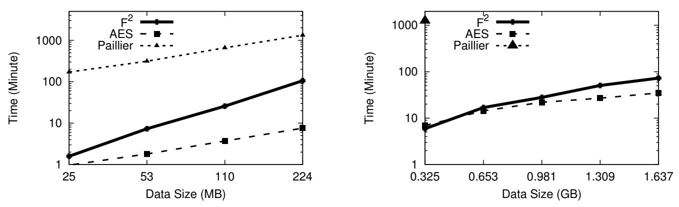
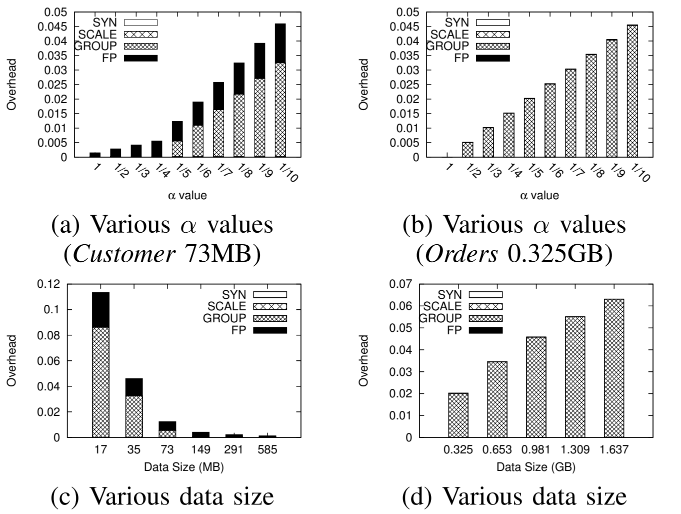

(a) Synthetic dataset (α=0.25) (b) Orders dataset (α=0.2)

Fig. 8: Time performance Comparison

We also compare the time performance of F^2 with *AES* and *Paillier*. The result is shown in Figure 8. It is not surprising that F^2 is slower than *AES*, as it has to handle with the FD-preserving requirement. On the other hand, even though F^2 has to take additional efforts to be FD-preserving (e.g., finding $MASs$, splitting and scaling, etc.), its time performance is much better than *Paillier*. This shows the efficiency of our probabilistic encryption scheme. It is worth noting that on the *Orders* dataset, *Paillier* takes 1247.27 minutes for the data of size 0.325GB, and cannot finish within one day when the data size reaches 0.653GB. Thus we only show the time performance of *Paillier* for the data size that is below 0.653GB.


(a) Various α values (*Customer* 73MB) (b) Various α values (*Orders* 0.325GB)

(c) Various data size (*Customer* $\alpha = 0.2$) (d) Various data size (*Orders* α=0.2)

Fig. 9: Amounts of Artificial Records Added by F^2

C. FD Discovery Accuracy

We evaluate the FD discovery accuracy of F^2 and the three baseline approaches on the first $30,000$ records from the *Orders* dataset. In particular, let $\mathcal{F}$ be the set of FDs discovered from the original dataset D, and $\mathcal{F}'$ be the FDs discovered from the encrypted dataset $\hat{D}$. We use *precision* and *recall* to evaluate the accuracy of FD discovery. Formally, the precision is defined as $\frac{|\mathcal{F}\cap\mathcal{F}'|}{|\mathcal{F}'|}$ (i.e., the fraction of the FDs in $\hat{D}$ that also exist in D), while recall is $\frac{|\mathcal{F}\cap\mathcal{F}'|}{|\mathcal{F}|}$ (i.e., the fraction of original FDs in D that can be discovered in $\hat{D}$). Intuitively, an encryption approach is FD-preserving only if both precision and recall are 1. To measure the impact of Step 4 of F^2 on

Approach	Precision	Recall
F^2 (before Step 4)	0.5385	0.7
F^2 (after Step 4)	1	1
FHOP	0	0
Paillier	0	0
AES	1	1

TABLE I: FD discovery accuracy

Approach	Attack accuracy
$F^2(\alpha = 0.02)$	0.01417
$F^2(\alpha = 0.05)$	0.03192
$F^2(\alpha = 0.1)$	0.0719
$F^2(\alpha = 0.25)$	0.1056
FHOP	0.1214
Paillier	0.1002
AES	0.3395

TABLE II: Security against FA attack

false positive FDs, we also measure the precision and recall before and after Step 4 of F^2.

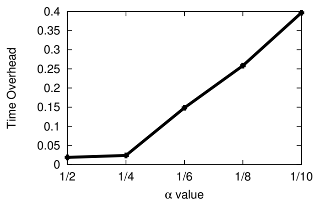
We report the precision and recall results in Table I. First, we observe that both the precision and recall of F^2 (after Step 4) are 1, which demonstrates that F^2 is FD-preserving. In particular, it shows that Step 4 is effective to remove false positive FDs and recover the real FDs that are eliminated by the false positive FDs. The other two probabilistic encryption schemes, Paillier and FHOP, cannot preserve any FD at all. AES, a deterministic encryption scheme that we applied at the cell level, is able to retain all the original FDs.

D. Security Against FA Attack

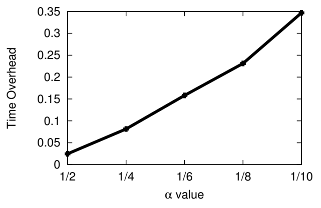
We implement FA attack defined in [5], and launch FA attack to evaluate the robustness of F^2 as well as the three baseline schemes. In particular, let h be the number of unique ciphertext values that can be successfully mapped to their plaintext values by FA attack. The *accuracy* of FA attack is defined as $acc = \frac{h}{n}$, where n is the total number of unique ciphertext values. Intuitively, the higher acc is, the weaker the encryption scheme is against FA attack. We launch FA attack on the first $30,000$ records of the *Orders* dataset that is encrypted by F^2 and three baseline schemes respectively, and report the accuracy in Table II. We have the following observations. First, the attack accuracy of F^2 in practice is always smaller than the theoretical guarantee α against FA attack. Second, F^2 provides comparable security against FA attack as FHOP and Paillier, even for the weak security guarantee as $\alpha = 0.25$. When we choose smaller values for α, F^2 can provide better security than FHOP and Paillier against FA attack. However, it has to pay the price for additional overhead for FD discovery and encryption/decryption due to the inserted artificial records. For example, the fraction of artificial records can be as large as 52.96% when $\alpha = 0.1$. It may also leak additional information to some extent due to the FCPA attack (not evaluated in this part of experiments).

E. Outsourcing VS. Local Computations

First, we compare the data owner's performance of finding FDs locally and encryption for outsourcing. We implement the TANE algorithm [6] and apply it on our datasets. First, we compare the time performance of finding FDs locally (i.e. applying TANE on the original dataset D) and outsourcing preparation (i.e., encrypting D by F^2). It turns out that finding FDs locally is significantly slower than applying F^2 on the synthetic dataset. For example, TANE takes 1,736 seconds on the synthetic dataset whose size is 25MB to discover FDs, while F^2 only takes 2 seconds.



(a) Customer dataset (73MB)



(b) Orders dataset (0.325GB)

Fig. 10: FD Discovery Time Overhead

Second, we compare the performance of discovering FDs from the original and encrypted data, for both *Customer* and *Orders* datasets. We define the *dependency discovery time overhead* $o = \frac{T'-T}{T}$, where T and T' are the time of discovering FDs from D and $\hat{D}$ respectively. The result is shown in Figure 10. For both datasets, the time overhead is small. It is at most 0.4 for the *Customers* dataset and 0.35 for the *Orders* dataset. Furthermore, the discovery time overhead increases with the decrease of α value. This is because with smaller α, the GROUP and FP steps insert more artificial records to form ECGs for higher security guarantee. This is the price to pay for higher security guarantee.

VII. Related Work

Data security is taken as a primary challenge introduced by the database-as-a-service (DaS) paradigm. To protect the sensitive data from the DaS service provider, the client may transform her data so that the server cannot read the actual content of the data outsourced to it. Hacigumus et al.'s work [20], [11] is one of the pioneering research that explores the data encryption for the DaS paradigm. They develop a framework that supports efficient query execution over encrypted data in the DaS paradigm. The key idea is to split the domain values into buckets. However, the bucket-based scheme is vulnerable against the frequency analysis attack as it uses the deterministic encryption. CryptDB [21] employs multiple encryption functions and encrypts each data item under various sequences of encryption functions in an onion approach. Alternatively, Cipherbase [22] exploits the trusted hardware (secure co-processors) to process queries on encrypted data. These cryptographic techniques are not FD-preserving.

Data encryption for outsourcing arises the challenge of performing computations over the encrypted data. A number of new encryption schemes like searchable encryption [16], order-preserving encryption [23], [24], attribute-based encryption [25], [26], and format-preserving encryption [27] are developed to support a diverse set of data features. These encryption schemes are called *property-preserving* [7], as they allow to learn the properties of the dataset, by only looking at the encrypted data elements. Most of the known practical property-preserving solutions leak some information. [8] initialized the research on the inference attacks against the property-preserving encryption. The explored inference attacks include the frequency analysis attack against the classical ciphers, the query-recovery attack [8] against searchable symmetric encryption (SSE) schemes, the ℓ_p-optimization attack [5] against the deterministic encryption, and various attacks (e.g., [5], [13]) against the order-preserving encryption. FD-preserving encryption is a type of property-preserving encryption scheme. However, none of the existing work considered the inference attack against the FD-preserving encryption.

Integrity constraints such as FDs are widely used for data cleaning. There have been very few efforts on finding data integrity constraints and cleaning of inconsistent data in private settings. Talukder et al. [28] consider a scenario where one party owns the private data quality rules (e.g., FDs) and the other party owns the private data. These two parties wish to cooperate by checking the quality of the data in one party's database with the rules discovered in the other party's database. They require that both the data and the quality rules need to remain private. They propose a cryptographic approach for FD-based inconsistency detection in private databases without the use of a third party. The quadratic algorithms in the protocol may incur high cost on large datasets [28]. Barone et al. [29] design a privacy-preserving data quality assessment that embeds data and domain look-up table values with Borugain Embedding. The protocol requires a third party to verify the (encrypted) data against the (encrypted) look-up table values for data inconsistency detection. Both work assume that the data quality rules such as FDs are pre-defined. None of these work can be directly applied to our setting due to different problem definition and possibly high computational cost.

Efficient discovery of FDs in relations is a well-known challenge in database research. Several approaches (e.g., TANE [6] and FD_MINE [30]) have been proposed. [31] classifies and compares seven FD discovery algorithms in the literature. [32] presents an excellent survey of FD discovery algorithms.

VIII. Conclusion and Discussion

In this paper, we presented F^2 algorithm that is FD-preserving and frequency-hiding. It can provide provable security guarantee against the FD-preserving chosen plaintext attack and the frequency analysis attack. Our experiment results demonstrate the efficiency of our approach.

We acknowledge that F^2 does not support efficient data updates, since it has to apply splitting and scaling from scratch if there is any data update. For the future work, we will consider how to address this important issue.

IX. Acknowledgement

This material is based upon work supported by the National Science Foundation (NSF) under Grant No. 1350324 and 1464800. We would also thank the anonymous reviewers for their comments and suggestions.

References

[1] C. Batini *et al.*, "A comparative analysis of methodologies for database schema integration," *ACM Computing Surveys*, pp. 323–364, 1986.

[2] L. Chiticariu *et al.*, "Semi-automatic schema integration in clio," in *Proceedings of the Very Large Database Endowment*, 2007, pp. 1326–1329.

[3] W. Fan *et al.*, "Discovering conditional functional dependencies," *IEEE Transactions on Knowledge and Data Engineering*, pp. 683–698, 2011.

[4] G. I. Davida *et al.*, "A database encryption system with subkeys," *ACM Transactions on Database Systems*, pp. 312–328, 1981.

[5] M. Naveed *et al.*, "Inference attacks on property-preserving encrypted databases," in *ACM Conference on Computer and Communications Security*, 2015, pp. 644–655.

[6] Y. Huhtala *et al.*, "Tane: An efficient algorithm for discovering functional and approximate dependencies," *The Computer Journal*, vol. 42, no. 2, pp. 100–111, 1999.

[7] O. Pandey and Y. Rouselakis, "Property preserving symmetric encryption," in *International Conference on the Theory and Applications of Cryptographic Techniques*, 2012, pp. 375–391.

[8] M. S. Islam *et al.*, "Access pattern disclosure on searchable encryption: Ramification, attack and mitigation." in *Network and Distributed System Security Symposium*, 2012, pp. 12–23.

[9] P. Bohannon *et al.*, "Conditional functional dependencies for data cleaning," in *IEEE International Conference on Data Engineering*, 2007, pp. 746–755.

[10] B. Marnette *et al.*, "Scalable data exchange with functional dependencies," *Proceedings of Very Large Database Endowment*, pp. 105–116, 2010.

[11] H. Hacigumus *et al.*, "Executing sql over encrypted data in the database-service-provider model," in *ACM International Conference on Management of Data*, 2002, pp. 216–227.

[12] B. Dong *et al.*, "Prada: Privacy-preserving data-deduplication-as-a-service," in *ACM International Conference on Information and Knowledge Management*, 2014, pp. 1559–1568.

[13] F. B. Durak *et al.*, "What else is revealed by order-revealing encryption?" in *ACM Conference on Computer and Communications Security*, 2016, pp. 1155–1166.

[14] M. Bellare *et al.*, "A concrete security treatment of symmetric encryption," in *Symposium on of Foundations of Computer Science*, 1997, pp. 394–403.

[15] A. Heise *et al.*, "Scalable discovery of unique column combinations," *Proceedings of Very Large Database Endowment*, pp. 301–312, 2013.

[16] D. X. Song *et al.*, "Practical techniques for searches on encrypted data," in *IEEE Symposium on Security and Privacy*, 2000, pp. 44–55.

[17] B. Dong and W. Wang, "Frequency-hiding dependency-preserving encryption for outsourced databases (full version)," http://www.cs.stevens.edu/~hwang4/papers/F2-icde17.pdf, 2017.

[18] T. Sanamrad *et al.*, "Randomly partitioned encryption for cloud databases," in *Data and Applications Security and Privacy XXVIII*, 2014, pp. 307–323.

[19] F. Kerschbaum, "Frequency-hiding order-preserving encryption," in *ACM Conference on Computer and Communications Security*, 2015, pp. 656–667.

[20] H. Hacigumus *et al.*, "Providing database as a service," in *IEEE International Conference on Data Engineering*, 2002, pp. 29–38.

[21] R. A. Popa *et al.*, "Cryptdb: Processing queries on an encrypted database," *Communications of the ACM*, pp. 103–111, 2012.

[22] A. Arasu *et al.*, "Orthogonal security with cipherbase," in *Conference on Innovative Data Systems Research*, 2013.

[23] A. Boldyreva *et al.*, "Order-preserving symmetric encryption," in *International Conference on the Theory and Applications of Cryptographic Techniques*, 2009, pp. 224–241.

[24] A. Boldyreva *et al.*, "Order-preserving encryption revisited: Improved security analysis and alternative solutions," in *Annual Cryptology Conference*, 2011, pp. 578–595.

[25] V. Goyal *et al.*, "Attribute-based encryption for fine-grained access control of encrypted data," in *Conference on Computer and Communications Security*, 2006, pp. 89–98.

[26] S. Garg *et al.*, "Attribute-based encryption for circuits from multilinear maps," in *Advances in Cryptology*, 2013, pp. 479–499.

[27] M. Bellare *et al.*, "Format-preserving encryption," in *International Workshop on Selected Areas in Cryptography*, 2009, pp. 295–312.

[28] N. Talukder *et al.*, "Detecting inconsistencies in private data with secure function evaluation," Purdue University, Tech. Rep., 2011.

[29] D. Barone *et al.*, "A privacy-preserving framework for accuracy and completeness quality assessment," *Emerging Paradigms in Informatics, Systems and Communication*, 2009.

[30] H. Yao *et al.*, "Fd_mine: discovering functional dependencies in a database using equivalences," in *IEEE International Conference on Data Mining*, 2002.

[31] T. Papenbrock *et al.*, "Functional dependency discovery: An experimental evaluation of seven algorithms," *Proceedings of the Very Large Database Endowment*, 2015.

[32] J. Liu *et al.*, "Discover dependencies from data a review," *IEEE Transactions on Knowledge and Data Engineering*, pp. 251–264, 2010.