Revisiting Security Risks of Asymmetric Scalar Product Preserving Encryption and Its Variants

Weipeng Lin, Ke Wang, Zhilin Zhang
School of Computing Science
Simon Fraser University
British Columbia, Canada
weipengl@sfu.ca, wangk@cs.sfu.ca, zhilinz@sfu.ca

Hong Chen
Department of Computer Science and Technology
Renmin University of China
P.R. China
chong@ruc.edu.cn

Abstract—**Cloud computing has emerged as a compelling vision for managing data and delivering query answering capability over the internet. This new way of computing also poses a real risk of disclosing confidential information to the cloud. Searchable encryption addresses this issue by allowing the cloud to compute the answer to a query based on the ciphertexts of data and queries. Thanks to its inner product preservation property, the *asymmetric scalar-product-preserving encryption* (ASPE) has been adopted and enhanced in a growing number of works to perform a variety of queries and tasks in the cloud computing setting. However, the security property of ASPE and its enhanced schemes has not been studied carefully. In this paper, we show a complete disclosure of ASPE and several previously unknown security risks of its enhanced schemes. Meanwhile, efficient algorithms are proposed to learn the plaintext of data and queries encrypted by these schemes with little or no knowledge beyond the ciphertexts. We demonstrate these risks on real data sets.**

I. Introduction

The current trend towards cloud based Database-as-a-Service (DaaS) as an alternative to traditional on-site RDBMSs has largely been driven by the perceived simplicity and cost effectiveness. For example, Amazon Web Services let users store personal data via its Simple Storage Service and perform computations on stored data using the Elastic Compute Cloud. For the privacy reason, the client data managed by the cloud must be encrypted while the cloud is allowed to compute a query on behalf of the clients. Sending the whole encrypted data to the user for each query obviously does not conform to the spirit of delegating the data processing to the cloud. A promising direction is *Symmetric Searchable Encryption* (SSE) [19], which enables the cloud to search directly on encrypted data. Traditionally, SSE schemes focus on the study of provable security, notably semantic security [6][9]. Most semantic security solutions require scanning the entire database for answering each query, which is not suitable for large applications.

To address the scalability requirement, the *Asymmetric Scalar-product-preserving Encryption* (ASPE) [25] was proposed as an encryption technique for efficient query processing while adopting an ad hoc security approach. A key property of ASPE is the preservation of the inner product between data and query, which makes it particularly suitable for designing a sub-linear search for distance and similarity based queries. More details are presented in Section II. For this reason, a growing number of recent works have adopted or enhanced ASPE to support a variety of important queries in cloud computing, including range queries [23], nearest neighbor queries [25], similarity based ranking queries [5][20][27][13], fuzzy multi-keyword match queries [22][8], privacy preserving computation of inner product [15], secure image search [29] and privacy-preserving biometric identification [28][24].

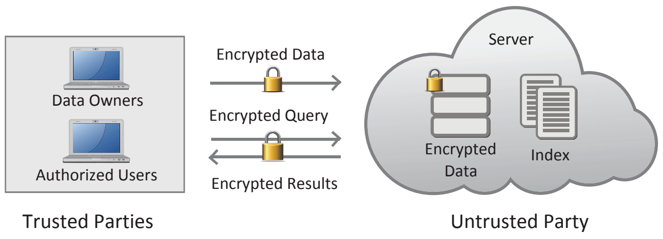


Fig. 1: A typical SSE system

On the other hand, the security property of ASPE has not been carefully studied. Until now, the most known security claim is Theorem 6 in [25], which shows that ASPE is resilient to a level-3 attack where the adversary is allowed to acquire knowledge on some plaintext-ciphertext pairs for records in the database. In this paper, we show a *complete disclosure* of ASPE under the level-3 attack, that is, the adversary can recover the entire plaintext database. To our knowledge, this is the first work systematically demonstrating this vulnerability. In particular, we pointed out that the previous attack on textsfASPE shown in [26] is not effective. Furthermore, our finding suggests that ASPE does not even resist the weakest adversary who knows nothing more than the ciphertexts. This study implies that the works based on ASPE are potentially under similar security risks.

A. Adversary Models

To explain our contributions, we adopt the SSE system in Figure 1. *Data owners* and *authorized users* are on the trusted client side, whereas the *cloud server* is "honest-but-curious". The data owner encrypts each record into a ciphertext and uploads the ciphertexts to the cloud server. Subsequently, each query request from a user is also encrypted and submitted to the cloud server in the form of trapdoor. The cloud server is responsible for computing the user query using the ciphertexts and the trapdoor. In this work, the terms "server", "cloud", and "adversary" are interchangeable. The following adversary models are based on additional information, beside ciphertexts, available to the adversary. Please refer to [7] for the motivations of these models.

Ciphertext-only attack model (COA): In this model, the adversary has only access to the ciphertexts. This is the weakest adversary model.

Known-plaintext attack model (KPA): In addition to the ciphertexts, the adversary also has the ability to acquire (or observe) pairs of plaintext-ciphertext for some number of records in the database. For example, if the adversary knows that someone joins a club and observes a new encrypted record afterward, the adversary can associate this person's plaintext record with the new ciphertext observed.

Chosen-plaintext attack model (CPA): A more powerful adversary has the ability to obtain the pairs of plaintext-ciphertext for some records of the adversary's choice. For example, the adversary may be able to influence the client side to encrypt certain plaintext records.

In the above list, the adversary is getting more powerful because of having access to more information. A security risk under a less powerful adversary is also a security risk under a more powerful adversary. In this paper, we consider COA and KPA models for adversaries and demonstrate the security risks of ASPE and its subsequent enhanced schemes. These security risks are also security risks under any more powerful adversary model.

B. Contributions

We present the following security risks of ASPE and its enhanced schemes:

• **Security risk of ASPE**: We show that ASPE is subject to the *complete disclosure* of plaintext information to a *KPA adversary*. In particular, we give an efficient attack algorithm to reconstruct the plaintext of the entire database and all processed queries, once the adversary acquires plaintext-ciphertext pairs for a small number of records under the KPA model. This finding implies that the security claim about ASPE presented in [25] does not hold. We point out that the previous attack in [26] was not effective and does not imply our finding. The details are presented in Section III.

• **Security risk of ASPE with noise enhancement**: The security of ASPE was enhanced by injecting random noises in recent works, such as MRSE [5]. Despite this additional security mechanism, we give an efficient algorithm for a *KPA adversary* to reconstruct the plaintext of queries. We focus on data over the *binary domain* because they are common in many applications such as bloom filters [4], implicit feedback, text documents. We present this attack in Section IV and demonstrate its risk on real life data in Section VI.

• **Security risk of ASPE with camouflaging enhancement**: Recent enhancement of ASPE, such as MKFSE [22], camouflages data and queries into secret binary vectors before applying ASPE, which renders the KPA adversary's knowledge on plaintext-ciphertext pairs useless. Despite this additional security mechanism, we give an efficient algorithm for a *COA adversary* to reconstruct the binary vectors without requiring any knowledge on plaintext-ciphertext pairs. We present this attack in Section V and demonstrate how it could lead to the disclosures of plaintext information in Section VI.

II. Preliminaries

We first present some notations and then describe briefly how ASPE works. In this paper, all vectors are column vectors.

- P_i – A data record in DB represented as a d-dimensional vector.
- $I_i/I_i^{'}$ – The plaintext/ciphertext of the index associated with P_i (constructed for query testing), represented as vectors.
- Q_j – A query represented as a d-dimensional vector.
- $T_j/T_j^{'}$ – The plaintext/ciphertext of the trapdoor for Q_j (constructed for query testing), represented as vectors.

Therefore, each record or query has three representations: the plaintext record and query P_i and Q_j, the plaintext index and trapdoor I_i and T_j, and the ciphertexts I_i' and T_j'.

ASPE [25] encrypts records and queries using different encryption operations (thus, the term "asymmetric"). Consider a record P_i and a query Q_j, represented by d-dimensional vectors. The index I_i and the trapdoor T_j are generated by

$$\begin{aligned} I_i &= (P_i^T, -0.5||P_i||^2)^T \\ T_j &= r_j(Q_j^T, 1)^T \end{aligned} \tag{1}$$

where $\mathbf{v}^T$ denotes the transpose of the vector $\mathbf{v}$, $||P_i||^2$ is the length of P_i, and r_j is a value randomly chosen for each Q_j. I_i and T_j are $(d+1)$-dimensional column vectors. Notice that I_i and P_i can be derived from each other and Q_j can be derived from T_j. Therefore, all of I_i, T_j, P_i, Q_j are considered sensitive. To generate the ciphertext I_i' and T_j', ASPE has two alternative schemes, the basic scheme or Scheme 1, and the enhanced scheme or Scheme 2.

Scheme 1. This scheme uses a secret $(d+1) \times (d+1)$ invertible matrix M as the secret key for encrypting the I_i and T_j as follows.

$$\begin{aligned} I_i' &= M^T I_i \\ T_j' &= M^{-1} T_j \end{aligned} \tag{2}$$

The encryption preserves the inner product between I_i and T_j, i.e.,

$$I_i'^T T_j' = I_i^T T_j = r_j(P_i^T Q_j - 0.5||P_i||^2) \tag{3}$$

Using this equality, Theorem 3 in [25] shows that a point P_1 is nearer to a query point Q_j than another point P_2 if and only if $(I_1' - I_2')^T T_j' > 0$. Thus, the server can rank points P_i by their distance to a query Q_j using the ciphertexts I_i' and T_j'. On the other hand, the asymmetric encryption does not preserve the inner product between two indexes or between two trapdoors, so the server cannot compare the distance between two records or two queries [25], a property that is considered sensitive. Theorem 4 in [25] shows that Scheme 1 does not resist the

KPA adversary: the adversary can find the $(d+1)\times(d+1)$ secret matrix M using Equation (2) if he can acquire (P_i, I'_i) pairs for $d+1$ linearly independent records P_i.

Scheme 2. To fix this problem, the authors of [25] proposed the enhanced scheme, i.e., Scheme 2, by employing two tricks. The first trick is expanding the $d+1$ dimensional I_i and T_j to d' dimensional ($d' = d+1+w$) vectors $\hat{I_i}$ and $\hat{T_j}$ by padding w artificial attributes whose inner product is equal to 0. The second trick uses a secret bit string S of length d' to split $\hat{I_i}$ into two secret shares ($\hat{I_{ia}}$, $\hat{I_{ib}}$) and split $\hat{T_j}$ into ($\hat{T_{ja}}$, $\hat{T_{jb}}$) such that

$$\hat{I_i}^T\hat{T_j} = \hat{I_{ia}}^T\hat{T_{ja}} + \hat{I_{ib}}^T\hat{T_{jb}} \tag{4}$$

where each secret share is a d' dimensional vector. See [25] for details. The split index and trapdoor are now encrypted by two $d'\times d'$ invertible matrices M_1 and M_2 as follows.

$$\begin{aligned} I'_{ia} &= M_1^T\hat{I_{ia}} \quad & I'_{ib} &= M_2^T\hat{I_{ib}} \\ T'_{ja} &= M_1^{-1}\hat{T_{ja}} \quad & T'_{jb} &= M_2^{-1}\hat{T_{jb}} \end{aligned} \tag{5}$$

Let I'_i denote the ciphertext (I'_{ia}, I'_{ib}), T'_j denote the ciphertext (T'_{ja}, T'_{jb}). We have

$$I_i'^T T'_j = I_{ia}'^T T'_{ja} + I_{ib}'^T T'_{jb} \tag{6}$$

Since $M_1M_1^{-1}$ and $M_2M_2^{-1}$ are equal to the identity matrix and the inner product over the w padded attribute values is equal to 0, Scheme 2 preserves the inner product between I_i and T_j [25]:

$$\begin{aligned} I_i'^T T'_j =& I_{ia}'^T T'_{ja} + I_{ib}'^T T'_{jb} = \hat{I_{ia}}^T\hat{T_{ja}} + \hat{I_{ib}}^T\hat{T_{jb}} \\ =& I_i^T T_j (= r_j(P_i^T Q_j - 0.5||P_i||^2)) \end{aligned} \tag{7}$$

For this reason, exactly same as for Scheme 1, Scheme 2 can be used to compare the distance of record points to a query point. Unlike Scheme 1, since the secret splitting injects randomness into the split index (I_{ia}, I_{ib}) and split trapdoor (T_{ja}, T_{jb}), it is infeasible to find M_1 and M_2 from Equation (5) like in Scheme 1. [25] proved that Scheme 2 is resilient to a KPA adversary, quoted below

Theorem 6 in [25] claims: "Scheme 2 is resilient to a level-3 attack if the attacker cannot derive the splitting configuration."

The level-3 attack in [25] is exactly our KPA adversary model and the splitting configuration is the secret bit string S. The above claim implies that a KPA adversary cannot learn the plaintext information of data and queries that are encrypted by Scheme 2. Below, we shall consider only Scheme 2 (because it enhances the security of Scheme 1) and refer it as ASPE. The rest of the paper focuses on the security risk of ASPE and enhanced schemes in the literature.

III. Security Risk of ASPE

In this section, we show that Theorem 6 in [25] does not hold, in fact, ASPE suffers from a complete disclosure for a KPA adversary. First we argue that a previously identified attack is not effect, and then we present our attack.

A. Previous Attacks

[26] shows that if the adversary knows the pairs (Q_j, T'_j) encrypted by ASPE for d queries Q_j, $1 \le j \le d$, the adversary can learn the unknown index I_i for a record P_i with d unknown variables (thus, learns the plaintext information) by solving the following "d linear equations", $1 \le j \le d$:

$$I_i'^T T'_j = I_i^T T_j \tag{8}$$

(i.e. Equation (7)) where the first d elements in I_i are d unknown variables. Refer to Equation (1) for the definition of I_i and T_j. A closer look reveals that this attack cannot be executed as suggested in [26] for two reasons: first, each T_j involves a random value r_j unknown to the adversary, hence, the above d equations actually contain $2d$ unknown variables (instead of d unknown variables): d unknown variables for the random values $r_1, \cdots, r_d$ generated to randomize $Q_1, \cdots, Q_d$, and the d unknown variables for the first d elements of I_i. Second, the $(d+1)$th element in I_i is a quadratic term $-0.5||P_i||^2$, thus, these equations are not a linear system. For these reasons, it is unclear that the above equations have a unique solution for the unknown variables.

B. Vulnerability of ASPE

In the rest of this section, we present an efficient algorithm for a KPA adversary to reconstruct the plaintext of the entire database and all processed queries. Under our KPA adversary model, we assume that the adversary acquires plaintext-ciphertext pairs $\{(P_i, I'_i)\}$, therefore $\{(I_i, I'_i)\}$, for $(d+1)$ records $O = \{P_1, \cdots, P_{d+1}\}$ in DB, where $P_1, \cdots, P_{d+1}$ are linearly independent. We further assume that the adversary has access to the ciphertext T'_j of all processed queries Q_j, denoted by $\mathcal{T}'$. Moreover, we assume that $\mathcal{T}'$ contains $d+1$ T'_j such that their plaintexts T_j are linearly independent; this assumption holds because the cloud is expected to process many queries over time.

Algorithm 1 describes the steps that the above KPA adversary may use to reconstruct the plaintext index I_i for all records $P_i \in DB - O$, and the plaintext trapdoor T_j of all processed queries in $\mathcal{T}'$. Step 1 applies Equation (9) to find T_j for $T'_j \in \mathcal{T}'$. Note that T_j is a $d+1$ dimensional vector with $d+1$ unknown variables (indicated in red), and I_i, I'_i, T'_j are known. Since $P_1, \cdots, P_{d+1}$ are linearly independent, $I_1, \cdots, I_{d+1}$ are linearly independent, thus form a $(d+1)\times(d+1)$ invertible matrix. Therefore, Equation (9) has a unique solution for T_j. The adversary can repeat this computation for $T'_j \in \mathcal{T}'$ until finding $(d+1)$ trapdoors such that $T_1, \cdots, T_{d+1}$ are linearly independent. At the end of this step, the adversary obtains the plaintext-ciphertext pairs (T_j, T'_j) for $1 \le j \le d+1$.

In the second step, with the obtained pairs (T_j, T'_j), $1 \le j \le d+1$, the adversary applies Equation (10) to learn I_i for every remaining record $P_i \in DB - O$. In this case, I_i is the only unknown vector (indicated in red) with $d+1$ unknown variables. Since $T'_1, \cdots, T'_{d+1}$ are linearly independent, as mentioned above, Equation (10) has a unique solution for I_i.

Therefore, Algorithm 1 enables a KPA adversary to learn T_j for every processed Q_j and I_i for every $P_i \in DB - O$.

Algorithm 1 Linear Equation Program (*LEP*)

- Adversary model: a KPA adversary
- Input:
  1) Acquired pairs $\{(I_i, I'_i)\}$ for $P_i \in O \subseteq DB$, $1 \leq i \leq d+1$, where $P_1, \cdots, P_{d+1}$ are linearly independent;
  2) The set of ciphertext trapdoors $\mathcal{T}'$ for queries;
  3) The ciphertext indexes I'_i for $P_i \in DB - O$;
- Output: The plaintext trapdoors T_j for $T'_j \in \mathcal{T}'$; The plaintext indexes I_i corresponding to $P_i \in DB - O$;
- Procedure:

Step 1. Given $T'_j \in \mathcal{T}'$, the adversary infers the $(d+1)$-dimensional plaintext trapdoor T_j by solving the following instances of Equation (7)

$$\begin{aligned} I'^T_1 T'_j &= I^T_1 T_j \\ &\vdots \\ I'^T_{d+1} T'_j &= I^T_{d+1} T_j \end{aligned} \tag{9}$$

where $\{(I_i, I'_i)\}$ for $P_i \in O$. For Equation (9), this is a $d+1$ linear equation system with $d+1$ unknown variables in T_j, which has a unique solution for T_j. The adversary can stop Step 1 once finding the $d+1$ linearly independent trapdoors $\{T_1, \cdots, T_{d+1}\}$.

Step 2. At the end of Step 1 , the adversary obtains plaintext-ciphertext pairs $(T_1, T'_1), \cdots, (T_{d+1}, T'_{d+1})$. With these pairs, for each $P_i \in DB - O$, the adversary infers the $(d+1)$-dimensional index I_i of P_i using the following instances of Equation (7)

$$\begin{aligned} I'^T_i T'_1 &= I_i{}^T T_1 \\ &\vdots \\ I'^T_i T'_{d+1} &= I_i{}^T T_{d+1} \end{aligned} \tag{10}$$

This system has a unique solution for I_i.

Step 3. Output T_j for $T'_j \in \mathcal{T}'$ and I_i for $P_i \in DB - O$.

According to Equation (1), the adversary learns Q_j and P_i. The following statement summarizes this security risk.

Security Risk 1. *ASPE (i.e., Scheme 2) is vulnerable to a KPA adversary: if the adversary acquires $d+1$ plaintext-ciphertext pairs (P_i, I'_i) for $P_i \in DB$ where these P_i are linearly independent, the adversary can recover the plaintext of the entire database DB and all processed queries.*

Remark 1. Security Risk 1 advances the state-of-the-art because ASPE was previously claimed to be resilient to a KPA adversary (i.e., Theorem 6 in [25]). For a d dimensional database, with $d+1$ plaintext-ciphertext pairs, the adversary can solve the $(d+1)$ linear equation systems in Algorithm 1 using Gaussian elimination with the complexity $O((d+1)^3)$ to recover the plaintext of entire database and all processed queries encrypted by ASPE. This knowledge required and complexity are a very low bar for a determined adversary. As discussed at the beginning of this section, the attack shown in [26] is ineffective. In fact, [26] assumes the adversary's knowledge on plaintext-ciphertext pairs for queries, which is different from the adversary model in [25] that assumes knowledge on plaintext-ciphertext pairs for records. To our knowledge, Algorithm 1 is the first work demonstrating a complete disclosure by assuming exactly the same adversary model as in [25].

IV. Security Risk of ASPE with Noise Enhancement

The recent work [5] enhanced ASPE to protect additional information about the query result through injecting both additive and multiplicative random noises. In this section we show that this noise enhanced scheme remains vulnerable to a KPA adversary. Again, we first discuss the work in [5] and then show the vulnerability.

A. Multi-keyword ranked search over encrypted data

[5] studied multi-keyword ranked search over encrypted cloud data, called MRSE below. Consider a record P_i and a query Q_j, both represented by d-dimensional binary vectors. The similarity of P_i with respect to Q_j is defined by the inner product $P_i^T Q_j$. The *top-k similarity search* for a query Q_j is to retrieve k records P_i in the database DB that have the highest similarity with respect to Q_j. When this querying task is outsourced to the cloud, besides plaintext information, the similarity score $P_i^T Q_j$ and similarity rank are also considered sensitive. To protect such information, unlike ASPE, MRSE generates the index I_i for P_i and the trapdoor T_j for Q_j by injecting random noises E_i^T and V_j^T:

$$\begin{aligned} I_i &= (P_i^T, E_i^T, 1)^T \\ T_j &= (r_j Q_j^T, r_j V_j^T, t_j)^T \end{aligned} \tag{11}$$

where $E_i^T = (\varepsilon_i^1, \cdots, \varepsilon_i^U)$ and each ε_i^k ($k \in [1, U]$) is a noise following the uniform distribution in the range $(\frac{2\mu}{U} - \sqrt{\frac{6}{U}}\sigma, \frac{2\mu}{U} + \sqrt{\frac{6}{U}}\sigma)$, where μ and σ are the mean and standard deviation of the normal distribution $N(\mu, \sigma^2)$; V_j^T is a random U-dimensional binary vector; r_j, t_j are random numbers. Notice that I_i and T_j are $(d+U+1)$-dimensional vectors. MRSE encrypts I_i and T_j into I'_i and T'_j using an invertible matrix, exactly as in ASPE shown in Section II. The details are omitted.

A record P_i is ranked with respect to a Q_j by the noisy similarity between P_i and Q_j computed by

$$I'^T_i T'_j = I_i^T T_j = r_j (P_i^T Q_j + E_i^T V_j) + t_j \tag{12}$$

For the purpose of ranking, r_j is positive. Therefore, the relative rank of two records P_1 and P_2 with respect to Q_j is distorted only by $E_i^T V_j$ because r_j and t_j are the same for the same query Q_j. By randomly setting $\frac{U}{2}$ bits of V_j to 1, and the resulting $E_i^T V_j$ is equal to the sum of $\frac{U}{2}$ randomly selected ε_i^k and follows the normal distribution $N(\mu, \sigma^2)$ [5]. A larger σ leads to more distortion of the relative rank of two records P_1 and P_2, thus, σ should be chosen carefully so that the "noisy" top-k answers reasonably approximate the true top-k answers.

B. *Vulnerability of MRSE*

With the U *new* unknown variables $E_i^T = (\varepsilon_i^1, \cdots, \varepsilon_i^U)$, Equation (9) has many possible solutions for T_j because the number of unknown variables is more than the number of equations. In this case, the attack algorithm proposed in Section III cannot be applied to MRSE. In the rest of this section, we describe a new algorithm to learn the plaintext information of a query Q_j from the plaintext-ciphertext pairs acquired by a KPA adversary. Suppose that the KPA adversary has acquired m plaintext-ciphertext pairs (P_i, I_i'), $1 \le i \le m$, for some m. In addition, suppose that the adversary has access to the ciphertext trapdoor T_j' of some processed query Q_j. Both P_i and Q_j are vectors over the binary domain. The adversary's goal is to learn the binary vector Q_j from the acquired pairs (P_i, I_i'), $1 \le i \le m$, and the ciphertext T_j'. We give an algorithm to learn such information.

First we rewrite Equation (12) as

$$E_i^T V_j = \widehat{r_j} I_i'^T T_j' - \widehat{t_j} - P_i^T Q_j \tag{13}$$

where $\widehat{r_j} = \frac{1}{r_j}$ and $\widehat{t_j} = \frac{t_j}{r_j}$. As discussed above, the noise $E_i^T V_j$ in Equation (13) follows the distribution $N(\mu, \sigma^2)$ [5] with the parameters μ and σ being public. The adversary's strategy is to constrain the value of $E_i^T V_j$, therefore, the value of $\widehat{r_j} I_i'^T T_j' - \widehat{t_j} - P_i^T Q_j$, to the interval $[\mu - l\sigma, \mu + l\sigma]$ with a high probability in the search for the solution to the unknown variables $\widehat{r_j}, \widehat{t_j}, Q_j$. For example, according to the 3-sigma rule, with $l = 3$, $E_i^T V_j$ falls into the interval $[\mu - l\sigma, \mu + l\sigma]$ with 99% probability. Since each pair (P_i, I_i'), $1 \le i \le m$, imposes one such constraint and since the actual Q_j satisfies all such constraints, a large enough m could provide a sufficient constraint so that a solution of Q_j is identical or similar to the actual Q_j. Algorithm 2 formulates the search of Q_j based on this idea as the mixed-integer linear program problem.

In Algorithm 2, $\widehat{r_j}, \widehat{t_j}, Q_j[k]$ $(k = 1, \cdots, d)$ are unknown variables (indicated in red) and the rest are known. We assume both $\widehat{r_j}, \widehat{t_j}$ are positive value. Constraints 3 and 4 ensure that the query is not an all-zero binary vector. Constraint 5 bounds the noise $E_i^T V_j$ to the interval $[\mu - l\sigma, \mu + l\sigma]$. Algorithm 2 will search for any solution in terms of $\widehat{r_j}, \widehat{t_j}, Q_j$ satisfying all the constraints in Equation (14). A solution may not be found, when the actual noises E_i and V_j for generating I_i and T_j are such that $E_i^T V_j$ is outside the interval $[\mu - l\sigma, \mu + l\sigma]$. Otherwise, a solution will be found. The choice of l serves a trade-off between finding a solution and confining the solution to the real answer Q_j.

Security Risk 2. *For data P_i and queries Q_j over the binary domain, Algorithm 2 enables a KPA adversary to search for the plaintext Q_j from the ciphertexts I_i' and T_j' produced by MRSE and acquired (P_i, I_i') pairs, $1 \le i \le m$. The risk of this attack depends on m and is evaluated in Section VI-A.*

Algorithm 2 Mixed Integer Linear Program (*MIP*)

- Adversary model: A KPA adversary
- Input:
  1) (P_i, I_i'), where $P_i \in O$, $1 \le i \le m$, and P_i is a d-dimensional binary vector;
  2) T_j' for a single d-dimensional binary vector Q_j;
  3) σ, μ and l;
- Output: $\widehat{r_j}$, $\widehat{t_j}$, and Q_j;
- Procedure:

```
1.  $0 < \widehat{r_j}$
2.  $0 < \widehat{t_j}$
3.  $Q_j[k] \in \{0, 1\}, k = 1 \cdots d$
4.  $\sum_{k=1}^{d} Q_j[k] \ge 1$
5.  $\mu - l\sigma \le \widehat{r_j} I_i'^T T_j' - \widehat{t_j} q - \sum_{k=1}^{d} P_i[k] Q_j[k] \le \mu + l\sigma$
    where $i = 1 \cdots m$
                                                            (14)
```

Remark 2. To our knowledge, Algorithm 2 is the first reported work to search for sensitive information from the ciphertexts encrypted by MRSE. While the worst-case complexity of MIP is $O(2^d)$, where d is the vocabulary size, the state-of-art optimization problem solvers such as Gurobi [2] can solve the nontrivial mixed integer programming problems with thousands of variables within a minute. Unlike the attack in Section III that leads to a complete disclosure, the risk of this attack depends on the similarity between the found solution for Q_j and the actual Q_j. We will evaluate this risk in Section VI.

[16] and [11] proposed methods to reconstruct the original value X from the noisy version $Y = X + R$ where R is an additive noise following a known distribution. Their methods are inadequate to break MRSE that adds *both* multiplicative and additive noises to the inner product $X = P_i^T Q_j$ as in Equation (12) where r_j, E_i, V_j, t_j all are random noises. Even if $E_i^T V_j$ follows a known distribution, the random values r_j and t_j will alter this distribution, thus, invalidate the assumption on a known distribution.

V. Security Risk ASPE with Camouflaging Enhancement

The secure multi-keyword fuzzy search over encrypted data, MKFSE, in [22] proposes to "camouflage" the index I_i and trapdoor T_j so that they cannot be derived from the original P_i and Q_j, therefore, a KPA adversary would not be able to take advantage of the acquired pairs (P_i, I_i') for learning T_j as in Section III. In the rest of this section, we show that if the camouflaged I_i and T_j are binary vectors, as in MKFSE, the adversary can still learn I_i and T_j *without* the help of acquired pairs (P_i, I_i').

A. *Multi-keyword fuzzy search over encrypted data*

First, we describe MKFSE [22]. To support fuzzy keyword search, each keyword in a record P_i is transformed to a bigram vector which is inserted into a d-dimensional binary vector (e.g., a bloom filter) using l locality-sensitive hash (LSH) functions [17]. An extra security layer is proposed to break the linkage between keywords and the bloom filter by using a pseudo-random function f. The trapdoor T_j of a query Q_j is generated in the same way. We represent this index and trapdoor generation of MKFSE by the functions

$$\begin{aligned} I_i &= f(LSH(P_i), K) \\ T_j &= f(LSH(Q_j), K) \end{aligned} \tag{15}$$

where K is the secret key for the pseudo-random function f, and I_i and T_j are both d-dimensional binary vectors. Intuitively, f can be thought as permuting the positions of the 0/1 string $LSH(P_i)$ or $LSH(Q_j)$ with the permutation determined by the secret key K. MKFSE encrypts I_i and T_j into I'_i and T'_j as in ASPE in Section II. The following equation holds

$$I_i^{'T} T'_j = I_i^T T_j \tag{16}$$

which allows the server to approximate the relevance score of P_i to Q_j using $I_i^{'T} T'_j$.

Unlike the attack on ASPE in Section III, I_i and T_j produced by Equation (15) cannot be derived from P_i and Q_j without knowing the secret key K (thus, the term "camouflaging"). Therefore, even with the acquired pairs (P_i, I'_i), without knowing I_i the equation $I_i^{'T} T'_j = I_i^T T_j$ does not help to learn T_j and Algorithm 1 cannot be used to attack MKFSE. However, since the generation of bloom filters I_i and T_j is deterministic, Equation (16) enables the adversary to infer the deterministic I_i and T_j directly without any acquired pairs (P_i, I'_i). The following discussion presents this attack.

B. Vulnerability of MKFSE

We show that an adversary can learn the information of I_i and T_j, given *only* the ciphertexts I'_i and T'_j produced by MKFSE; in other words, MKFSE is vulnerable to the weakest COA adversary. While learning I_i and T_j does not directly lead to the disclosure of plaintext P_i or Q_j, the deterministic property of LSH and f implies that the similarity between I_i or T_j reflects the similarity on plaintext. For example, similar I_1 and I_2 will be generated from similar P_1 and P_2 with a high probability; if the plaintext content of P_1 is learnt by the adversary, so is the plaintext content of P_2. In addition, the deterministic property of I_i and T_j opens a door to statistical analysis attack where the frequency distribution I_i and T_j could disclose the plaintext content of P_i and Q_j. We will evaluate these risks in Section VI. For now, we focus an efficient algorithm to learn I_i and T_j, given the ciphertexts I'_i and T'_j.

Let us suppose that the adversary observes m encrypted indexes I'_i for $1 \le i \le m$ and n encrypted trapdoors T'_j for $1 \le j \le n$. Let $\mathcal{R} \in \mathbb{Z}^{m \times n}$ denote the matrix for inner products, $\mathcal{R}[i][j] = I_i^{'T} T'_j$. From Equation (16), $\mathcal{R}[i][j] = I_i^T T_j$. $\mathcal{R}[i][j]$ is a non-negative integer because I_i and T_j are binary vectors. The adversary wants to find two matrices $\mathcal{I} \in \{0,1\}^{d \times m}$ and $\mathcal{T} \in \{0,1\}^{d \times n}$ such that each column in $\mathcal{I}$ is a reconstructed index I_i^* of I_i and each column in $\mathcal{T}$ is a reconstructed trapdoor T_j^* of T_j, and

$$\mathcal{R} \simeq \mathcal{I}^T \mathcal{T} \tag{17}$$

If I_i^* are close to I_i and T_j^* are close to T_j, I_i and T_j are disclosed. Since only the ciphertexts I'_i and T'_j are required, this attack can be launched by the weakest COA adversary.

Finding the two matrices $\mathcal{I}$ and $\mathcal{T}$ satisfying Equation (17) is in fact the factorization of an integer matrix into two binary matrices. We present this approach in Algorithm 3 based on the *sparse non-negative matrix factorization* (sparse-NMF) [12] with the following objective

$$\min_{\mathcal{I},\mathcal{T}} \frac{1}{2} ||\mathcal{R} - \mathcal{I}^T \mathcal{T}||_F^2 + \frac{\eta}{2} ||\mathcal{I}||_F^2 + \frac{\lambda}{2} \sum_{j=1}^{n} ||T_j^*||_1^2 \tag{18}$$

We choose the sparse non-negative matrix factorization for the following reason: T_j tends to be more sparse than I_i due to a small number of keywords in a query. This means that most elements of T_j^* are expected to be close to zero while only few are significantly non-zero values. This sparseness constraint is enforced through the Frobenius norm in the second term and the L1-norm in the third term. For a more detailed explanation, please refer to [12]. η and λ are constants heuristically chosen to satisfy the sparseness constraint. It is to be noted that the binary matrix factorization [18] is not suitable for our purpose because it applies to the matrix $\mathcal{R}$ that itself is over the binary domain whereas our $\mathcal{R}$ is over the integer domain.

Algorithm 3 Sparse Non-negative Matrix Factorization (*SNMF*)

• Adversary model: A COA adversary;
• Input: $\mathcal{R}$, d, θ, L, where $\mathcal{R}[i][j] = I_i^{'T} T'_j$ for $1 \le i \le m$ and $1 \le j \le n$;
• Output: $\mathcal{I} \in \{0,1\}^{d \times m}$ and $\mathcal{T} \in \{0,1\}^{d \times n}$ that satisfy Equation (17) and the sparseness constraint;
• Procedure:

```
$bestError \leftarrow \inf$
for $1 \le l \le L$ do
    $(\mathcal{I}, \mathcal{T}, error) \leftarrow sparse\_NMF(\mathcal{R}, d)$
    if $error < bestError$ then
        $best\mathcal{I} = \mathcal{I}$, $best\mathcal{T} = \mathcal{T}$, $bestError = error$
    end if
end for
$(\mathcal{I}, \mathcal{T}) \leftarrow ConvertToBinaryMatrix(best\mathcal{I}, best\mathcal{T}, \theta)$
```

Algorithm 3 presents a search algorithm for finding I_i and T_j from the ciphertexts I'_i and T'_i produced by ASPE. $\mathcal{R}$ is the ciphertext inner product matrix defined above, d is the dimensionality of indexes and trapdoors, θ is the threshold for conversion to binary values, and L is the number of runs. The $sparse\_NMF()$ function refers to the sparse non-negative matrix factorization method proposed in [12], which has $\mathcal{R}$ and d as the input parameters. $sparse\_NMF()$ is a non-deterministic function and Algorithm 3 returns the best result among L calls of $sparse\_NMF()$. The function

$ConvertToBinaryMatrix()$ converts each component in the input matrixes to a binary value: converting a value below θ to 0 and converting all other values to 1. We choose $\theta = 0.5$.

Security Risk 3. *For indexes I_i and trapdoors T_j over the binary domain, Algorithm 3 enables a COA adversary to search for I_i and T_j from the ciphertexts I'_i and T'_i produced by* ASPE*. The risk of this attack is evaluated in Section VI-B.*

Remark 3. Algorithm 3 is the strongest attack on ASPE because it applies to the weakest COA adversary that has access only to the ciphertexts. Each iteration in the non-negative matrix factorization has the complexity of $O(mnd)$ [10], thus, a polynomial time adversary can easily launch the attack of Algorithm 3. This attack highlights the vulnerability of MKFSE whose camouflaging mechanism has no effect on a COA adversary who does not need any knowledge on plaintext for the attack.

Attack Algorithm	Target Scheme	Adversary Model	Data Domain
LEP	ASPE	KPA	Real
MIP	MRSE (ASPE with random noises)	KPA	Binary
SNMF	MKFSE (ASPE with camouflaging)	COA	Binary

TABLE I: Summary of attack algorithms

VI. Empirical Analysis

Table IV summarizes the three attack algorithms, *LEP*, *MIP*, and *SNMF*, presented in the previous three sections. *LEP* always finds exactly the plaintext of all records and queries assuming that the required condition is satisfied. The effectiveness of *MIP* and *SNMF* must be evaluated empirically, which is the goal of this section. We implemented them in C++ and Matlab with the Gurobi optimization problem solver [2] and the non-negative matrix factorization toolbox [14]. All experiments were conducted on a machine with 3.07 GHz Intel Xeon W550 CPU, 12G memory, and running Windows 7.

Metrics. *MIP* and *SNMF* aim to reconstruct the plaintext indexes I_i and trapdoors T_j represented by binary vectors. We study the accuracy by *precision* and *recall* of reconstructing the 1's in a binary vector. Let $\mathbf{v}$ and $\mathbf{v}^*$ be the actual and reconstructed binary vectors respectively, and $\mathbf{v} \bigcap \mathbf{v}^*$ represent the intersection of $\mathbf{v}$ and $\mathbf{v}^*$. Let $|\mathbf{v}|$ and $|\mathbf{v}^*|$ and $|\mathbf{v} \bigcap \mathbf{v}^*|$ denote the number of 1's in $\mathbf{v}$, $\mathbf{v}^*$ and $\mathbf{v} \bigcap \mathbf{v}^*$, respectively. The *precision* of $\mathbf{v}^*$ is $\frac{|\mathbf{v} \bigcap \mathbf{v}^*|}{|\mathbf{v}^*|}$, i.e., the fraction of 1's in $\mathbf{v}^*$ which are actually 1's in $\mathbf{v}$. The *recall* of $\mathbf{v}^*$ is $\frac{|\mathbf{v} \bigcap \mathbf{v}^*|}{|\mathbf{v}|}$, i.e., the fraction of 1's in $\mathbf{v}$ that are found as 1's. If an attack algorithm reconstructs a binary vector with a high precision and recall within a reasonable time, the attack shows a high risk because the adversary can learn $\mathbf{v}$ through the reconstructed $\mathbf{v}^*$.

Parameters. We study three parameters of an attack: the dimensionality d of a record or query binary vector $\mathbf{v}$; the *density* of 1's in $\mathbf{v}$ defined as $\rho = \frac{|\mathbf{v}|}{d}$; the number m of acquired (P_i, I'_i) pairs for the *MIP* attack, or the number m of ciphertexts I'_i and the number n of ciphertexts T_j for the *SNMF* attack. For simplicity, we assume $m = n$ in the *SNMF* attack.

Data sets. Both synthetic and real life data set were used for our evaluation. m in our experiments refers to the number of the records acquired by the adversary for an attack, which is different from the full data cardinality $|DB|$. In fact, a smaller m used in the attack algorithm implies a more vulnerability of the encryption scheme. The *IBM synthetic data sets* are generated by the IBM Quest Data Generator [3] with the input of the total number of items d, the (average) density ρ of items in a transaction, and the number of transactions m. We converted each transaction into a binary vector of length d. The real life data set is the *Enron email data set* [1] with $39,861$ emails (i.e., $|DB| = 39,861$). We represented each email document by a length d bloom filter [4], as in [9] [22], by hashing each keyword in the email using h hash functions to set several positions in the bloom filter to 1 and all other positions to 0. More details of data and query generation will be discussed later.

We evaluate Security Risk 2 and Security Risk 3 through the following claims.

$\triangleright$ *Claim 1 (Security Risk 2):* If the noise injected by MRSE is such that the noisy query result remains useful, *MIP* is able to reconstruct the binary query vector T_j with a high accuracy, given the ciphertext T'_j and m pairs $\{(P_i, I'_i) \mid P_i \in O\}$ acquired by a KPA adversary, for a sufficiently large m.

$\triangleright$ *Claim 2 (Security Risk 3):* *SNMF* is able to reconstruct the binary vectors I_i and T_j with a high accuracy, given the ciphertexts I'_i and T'_j, $1 \le i \le m$ and $1 \le j \le n$, produced by MKFSE, for sufficiently large m and n. Having a large number of ciphertexts I'_i and T'_j does not impose a real constraint since ciphertexts are always available to the adversary.

A. Evaluation of Claim 1

MIP uses m acquired pairs (P_i, I'_i) for $P_i \in O$ to reconstruct the plaintext of a given query Q_j by constraining the noise injected within the interval $[\mu - l\sigma, \mu + l\sigma]$ (Equation (14)). We set $l = 3$ to ensure that the probability of the noise falling into this interval is close to 99%. The mean μ does not affect this probability. As suggested in [5], we set $\sigma = 0.5$ and $\sigma = 1$.

1) MIP on synthetic data sets: We consider $d \in \{100, 500, 1000\}$, $\rho \in \{5\%, 20\%, 35\%\}$, and $m = d$. For each (d, ρ) setting, we use the IBM data generator to produce $m = d$ records P_i as d-dimensional binary vectors, and $O_{d,\rho,m}$ denotes this set of P_i. For each d setting, we randomly generated 100 queries Q_j using the same data generator with the density being $\frac{15}{d}$, which was suggested in [5], and let $Query_d$ denote this set of Q_j. For each $O_{d,\rho,m}$, for $\sigma = 0.5$ and $\sigma = 1$, we run *MIP* with the input $\{(P_i, I'_i) \mid P_i \in O_{d,\rho,m}\}$ to find the plaintext of $Q_j \in Query_d$. The averaged precision and recall over Q_j are given in Table II.

	$d = m = 100$			$d = m = 500$			$d = m = 1000$		
	$\rho = 5\%$	$\rho = 20\%$	$\rho = 35\%$	$\rho = 5\%$	$\rho = 20\%$	$\rho = 35\%$	$\rho = 5\%$	$\rho = 20\%$	$\rho = 35\%$
$\sigma = 0.5$									
$P@query$	0.3164	0.8418	0.8923	0.8266	0.9170	0.9014	0.9119	0.9649	0.9080
$R@query$	0.2208	0.7922	0.9140	0.8818	0.9922	0.9796	0.8687	0.9903	0.9853
$Time\ (Seconds)$	0.0079	0.0126	0.1439	0.0457	0.2389	7.7684	10.198	3.781	2.485
$\sigma = 1$									
$P@query$	0.0942	0.5045	0.5206	0.2198	0.7058	0.8107	0.2957	0.7283	0.8540
$R@query$	0.0771	0.2829	0.3827	0.1374	0.6324	0.7607	0.2295	0.6734	0.8757
$Time\ (Seconds)$	0.0059	0.0091	0.0110	0.0401	2.856	3.4241	0.144	1.701	3.854

TABLE II: *MIP*'s precision (P), recall (R), and runtime for different settings on synthetic data sets

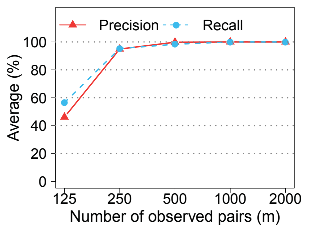
Under each d setting, there are 6 (ρ, σ) settings to run the attack for each query in $Query_d$, so there are 600 *MIP* attacks. With $d = 100$, 554 attacks find a solution. This number is 552 for $d = 500$, and 416 for $d = 1000$. One reason for not finding a solution is that the injected noise falls outside the interval $[\mu - l\sigma, \mu + l\sigma]$. Another reason is that the program terminates after some pre-set limit of time is reached. The precision and recall were collected when a solution was found. The chance of finding a solution is high (i.e., 554, 552, 416 out of 600) and when a solution is found, it took no more than 11 seconds.

With $\sigma = 0.5$, the reconstruction has precision and recall higher than 0.8 in most cases, and sometime above 0.9. With $\sigma = 1$, precision and recall are much lower. According to [5], this case is considered as "excessive noises", meaning that the similarity rank is distorted so badly that the noisy top-k answers returned are no longer a good approximation of true answers. For the noisy query result to be useful, $\sigma = 0.5$ is the more realistic and representative case.

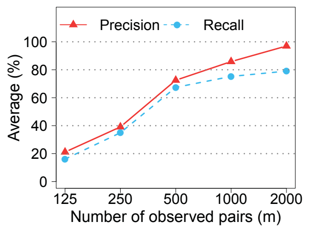
A lower density ρ (i.e., $\rho = 5\%$) weakens the power of *MIP*. In this case, fewer keywords occur in the data, the true inner product $T_i^T T_j$ is smaller and the relative noise is larger. This case is similar to the "excessive noises" case where the returned noisy top-k answers are no longer a good approximation of the true answers. For data and queries represented as bloom filters, which compresses a sparse vector, we believe that a higher density of $\rho = 20\%$ or $\rho = 30\%$ is the more common case. In this case, precision and recall are much higher, especially with $\sigma = 0.5$.

2) MIP on real life data sets: For the Enron data set, we generated five sets $O_m = \{P_i\}$ with $m \in \{125, 250, 500, 1000, 2000\}$ to simulate the records P_i whose pairs (P_i, I'_i) were acquired by a KPA adversary. O_m contains m emails P_i randomly selected from those in the Enron data set that have a density ρ in the range $[5\%, 35\%]$. We also generated 100 queries Q_j with the length $d = 500$ by the IBM data generator, exactly as in Section VI-A1, and let $Query$ denote this set of queries. Both P_i and Q_j are represented in a bloom filter of length $d = 500$. For each O_m, we run *MIP* with the input $\{(P_i, I'_i) \mid P_i \in O_m\}$ to find the plaintext of each Q_j in $Query$ and we report the averaged precision and recall of all Q_j in $Query$ where a solution was found.

As shown in Figure 2, higher precision and recall are



(a) $\sigma = 0.5$



(b) $\sigma = 1$

Fig. 2: *MIP*'s precision and recall vs the number of observed pairs m on Enron data sets ($d = 500$, $\rho \in [5\%, 35\%]$)

observed if the adversary observes more plaintext-ciphertext pairs. For example, with $m \geq 500$, the precision and recall are close to or higher than 0.8. For a small $m \in \{125, 250, 500\}$, the attacks have a high chance of finding a solution, ranging from 74% to 100%, within some pre-set limit of time. For a large $m \in \{1000, 2000\}$, the large number of constraints increases the runtime and the chance of finding a solution within the pre-set limit reduces. All runs of *MIP* terminated within 10 seconds when a solution was found.

In summary, this study supports *Claim 1*: despite injected random noises as in MRSE, the accuracy of reconstructing the plaintext of a query can be increased by acquiring more plaintext-ciphertext pairs under the KPA adversary model (i.e., a larger m). While injecting more noises can deter this attack, it also distorts the relative rank of answers, making the noisy top-k answers less useful.

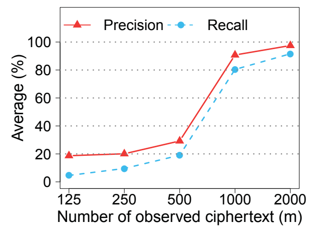
B. Evaluation of Claim 2

SNMF applies the sparse non-negative matrix factorization to the inner product result matrix $\mathcal{R}$, where $\mathcal{R}[i][j] = I_i'^T T'_j$, to reconstruct the indexes I_i and trapdoors T_j ($1 \leq i \leq m$ and $1 \leq j \leq n$) represented by d-dimensional binary vectors. In this experiment, we first study the accuracy of this attack with respect to different d, m, n, and the density ρ on synthetic data sets, and then we show the disclosures on the real Enron data set.

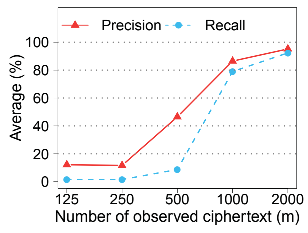
1) SNMF on synthetic data sets: To produce the input $\mathcal{R}$, we generated m binary vectors I_i and n binary vectors T_j in the same way as in Section VI-A1, but set $m = n$ to $2d$

	$d = 100, m = 200$			$d = 500, m = 1000$			$d = 1000, m = 2000$		
	$\rho = 5\%$	$\rho = 20\%$	$\rho = 35\%$	$\rho = 5\%$	$\rho = 20\%$	$\rho = 35\%$	$\rho = 5\%$	$\rho = 20\%$	$\rho = 35\%$
$P@data$	-	0.3407	0.4250	0.1718	0.9694	0.9957	0.0746	0.9246	0.9823
$R@data$	0.0000	0.6471	0.4915	0.1585	0.9593	0.9512	0.0661	0.9016	0.9013
$P@query$	-	0.6357	0.5082	0.1636	0.9621	0.9931	0.0453	0.8995	0.9452
$R@query$	0.0000	0.2069	0.1279	0.1201	0.9608	0.9774	0.0387	0.9035	0.9180
$Time\ (Seconds)$	261	264	238	28732	28649	29320	200890	188920	191749

TABLE III: *SNMF*'s precision (P), recall (R) and runtime for different settings on synthetic data sets



(a) $@data$



(b) $@query$

Fig. 3: *SNMF*'s precision and recall vs the number of ciphertexts m ($n = m$) on Enron data sets ($d = 500$, $\rho \in [5\%, 35\%]$)

(because factorizing a $m \times n$ matrix into two $m \times d$ and $d \times n$ matrices generally requires m and n to be greater than d). We encrypted I_i into I_i', T_j into T_j', and computed $\mathcal{R}[i][j] = I_i'^T T_j'$, $1 \leq i \leq m$ and $1 \leq j \leq n$. Table III presents the precision and recall of I_i^* and T_j^* reconstructed from $\mathcal{R}$.

First of all, compared to *MIP*, a *SNMF* attack took much longer time to run. However, *SNMF* tends to reconstruct the plaintext information with higher precision and recall, for example, more than 90% when $d > 500$ with the density $\rho = 20\%$ and $\rho = 35\%$. In these cases, the adversary recovers most of I_i and T_j that MKFSE intends to hide. The precision and recall deteriorate significantly for the low density $\rho = 5\%$; in this case most positions in a sparse data vector are 0's, so many solutions are feasible.

2) SNMF on real life data sets: We used the 5 sets O_m in Section VI-A2 where $d = 500$ and $m \in \{125, 250, 500, 1000, 2000\}$. For each O_m, we generated a set $Query_n$ of $n = m$ queries Q_j as binary vectors using the IBM data generator with the density being $\frac{15}{d}$. Figure 3 showed the average precision and recall of reconstructed I_i^* and T_j^*. The result is pretty consistent with those obtained on the synthetic data sets, that is, more ciphertexts, i.e., larger m and n, would help reconstruct the original indexes and trapdoors. Having more ciphertexts is not a real constraint because the adversary always has access to the ciphertext.

One may argue that the accurate reconstruction of I_i^* and T_j^* does not pose a risk because I_i and T_j are camouflaged using LSH and pseudo-random functions f (Equation 15). The risk comes from the fact that the generation of I_i and T_j is deterministic (because LSH and f are deterministic), which opens a door to learning the unknown content of other records from the acquired content of some records, and to statistical analysis attacks. Let us explain these points in details.

On average, the relative error of approximating the similarity between two documents P_i (measured by Jaccard Index [21]) by the similarity of their bloom filters I_i is $2.79*10^{-4}\%$. Such a small relative error implies that if the reconstructed I_i^* has superior precision and recall, the adversary has a good chance to learn the plaintext P_i. For example, the reconstructed I_{365}^* and I_{380}^* are identical and the precision and recall of the reconstruction are 97.26% and 93.02%, respectively. Such high precision and recall implies that I_{365} and I_{380} are highly similar. Then the small approximation error of bloom filters implies that P_{365} and P_{380} are highly similar. The original P_{365} contains the keyword "application approved". If the adversary acquires this information, he will further learn that P_{380} also contains "application approved" based on the high similarity of I_{365}^* and I_{380}^*. In this case, the adversary is right because P_{380} does contain the keyword "application approved". In the real attacking scenario, the adversary does not know the exact precision and recall of reconstruction, but he can always boost the accuracy of reconstruction by using more ciphertexts I_i' and T_j', i.e., larger m and n. This effectiveness is confirmed by the findings in Table III and Figure 3.

To explain the potential risk to statistical analysis attack, Table IV shows the frequency distribution of the five most frequent documents P_i in the set O_{2000}, as well as the frequency distribution of their plaintext indexes I_i and reconstructed indexes I_i^*. For example, P_{347} repeats 27 times in O_{2000}, so do their bloom filter I_{347} and the reconstructed I_{347}^*. The table shows that the frequency distribution of these five documents is completely preserved on I_i and I_i^*. This is because a bloom filter of the length 500 has a strong discriminative power. Thanks to this preservation, the adversary could infer the frequency information of P_i through that of I_i^*, and and use it to further infer the plaintext of P_i through background knowledge, such as "application approved" occurs 10% of the time. A similar attack applies to a query Q_j.

In summary, our study supports Claim 2: with enough data and not too low density, *SNMF* is able to recover the bloom filter content from the ciphertexts alone with a high accuracy. Unlike *MIP*, "enough data" here refers to the ciphertexts I_i' and T_j' because only the ciphertexts are required by *SNMF*, which is always available to the adversary. Also unlike *MIP*

	P_{347}	P_{1939}	P_{873}	P_{1938}	P_{731}
Frequency	27	11	9	8	7
	I_{347}	I_{1939}	I_{873}	I_{1938}	I_{731}
Frequency	27	11	9	8	7
	I^*_{347}	I^*_{1939}	I^*_{873}	I^*_{1938}	I^*_{731}
Frequency	27	11	9	8	7

TABLE IV: The frequency distribution of five most frequent documents P_i in O_{2000}. This distribution is completely preserved on the indexes I_i and the reconstructed indexes I^*_i

that may not find a solution due to the over-constrained interval for noises, the *SNMF* attack always returns a solution, whose quality can be improved using the unlimited ciphertexts. Although the *SNMF* attack took a longer runtime, this does not stop a determined adversary, not to mention that a more powerful machine can reduce the runtime. The risk of this attack is that the deterministic bloom filter generation opens a door to learning the unknown content of other records from the acquired content of some records, and to statistical analysis attacks. Finally, since *SNMF* does not need anything other than ciphertexts, it is a more general attack than *MIP* and *LEP*.

VII. Conclusion

A growing number of recent works were based on ASPE as a searchable encryption technique for performing a variety of queries and tasks in the cloud computing setting. In this work, we showed that ASPE is not resilient to a KPA adversary, by giving an efficient algorithm to reconstruct exactly the plaintext of the entire database and processed queries once the adversary acquires a small number of plaintext-ciphertext pairs under the KPA adversary model (Section III). This finding advances the state-of-the-art because ASPE was previously shown to be resilient to a KPA adversary [25] and we pointed out that the previous attack on ASPE demonstrated in [26] is not effective (Section III). We further demonstrated security risks for two enhanced schemes of ASPE, i.e., MRSE [5] (Section IV) and MKFSE [22] (Section V). The demonstrated security risk on MKFSE is particularly of concern because it is based on ciphertext alone, which holds for all adversary models. While a natural next step is to fix all identified security risks, doing so requires more work and is beyond the scope of this paper.

Acknowledgement

The work of the second author is partially supported by a Discovery Grant from Natural Sciences and Engineering Research Council of Canada, partially done during a visit to SA Center for Big Data Research hosted in Renmin University of China. The SA Center is partially funded by a Chinese National 111 Project Attracting International Talents in Data Engineering and Knowledge Engineering Research.

References

[1] Enron email data sets (bag of words). https://archive.ics.uci.edu/ml/machine-learning-databases/bag-of-words/. 2008.
[2] Gurobi. http://www.gurobi.com/academia/for-universities.
[3] Ibm quest sythetic data generator. http://ibmquestdatagen.sourceforge.net/. Accessed March 10, 2015.
[4] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 1970.
[5] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou. Privacy-preserving multi-keyword ranked search over encrypted cloud data. *IEEE Transactions on parallel and distributed systems*, 2014.
[6] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *ACM conference on Computer and Communications Security*, 2006.
[7] H. Delfs and H. Knebl. *Introduction to Cryptography: Principles and Applications*. 2002.
[8] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren. Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement. *IEEE Transactions on Information Forensics and Security*, 2016.
[9] E. J. Goh et al. Secure indexes. *IACR Cryptology ePrint Archive*, 2003.
[10] N.-D. Ho. *Nonnegative matrix factorization algorithms and applications*. PhD thesis, ÉCOLE POLYTECHNIQUE, 2008.
[11] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *ICDM*, 2003.
[12] H. Kim and H. Park. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, 2007.
[13] H. Li, Y. Yang, T. H. Luan, X. Liang, L. Zhou, and X. S. Shen. Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data. *IEEE Transactions on Dependable and Secure Computing*, 2016.
[14] Y. Li and A. Ngom. The non-negative matrix factorization toolbox for biological data mining. *Source code for biology and medicine*, 2013.
[15] K.-P. Lin and M.-S. Chen. Privacy-preserving outsourcing support vector machines with random transformation. In *SIGKDD*, 2010.
[16] K. Liu, C. Giannella, and H. Kargupta. A survey of attack techniques on privacy-preserving data perturbation methods. In *Privacy-Preserving Data Mining*. 2008.
[17] L. Paulevé, H. Jégou, and L. Amsaleg. Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern Recognition Letters*, 2010.
[18] M. Slawski, M. Hein, and P. Lutsik. Matrix factorization with binary components. In *Advances in Neural Information Processing Systems*, 2013.
[19] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *IEEE Symposium on Security and Privacy*, 2000.
[20] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li. Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. In *Proceedings of the ACM SIGSAC Symposium on Information, Computer and Communications Security*, 2013.
[21] P.-N. Tan et al. *Introduction to data mining*. Pearson Education India, 2006.
[22] B. Wang, S. Yu, W. Lou, and Y. T. Hou. Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud. In *INFOCOM*, 2014.
[23] P. Wang and C. V. Ravishankar. Secure and efficient range queries on outsourced databases using $\hat{R}$-tree. In *ICDE*, 2013.
[24] Q. Wang, S. Hu, K. Ren, M. He, M. Du, and Z. Wang. Cloudbi: Practical privacy-preserving outsourcing of biometric identification in the cloud. In *European Symposium on Research in Computer Security*, 2015.
[25] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis. Secure knn computation on encrypted databases. In *SIGMOD*, 2009.
[26] X. Xiao, F. Li, and B. Yao. Secure nearest neighbor revisited. In *ICDE*, 2013.
[27] J. Yu, P. Lu, Y. Zhu, G. Xue, and M. Li. Toward secure multikeyword top-k retrieval over encrypted cloud data. *IEEE Transactions on Dependable and Secure Computing*, 2013.
[28] J. Yuan and S. Yu. Efficient privacy-preserving biometric identification in cloud computing. In *INFOCOM*, 2013.
[29] J. Yuan, S. Yu, and L. Guo. Seisa: Secure and efficient encrypted image search with access control. In *INFOCOM*. IEEE, 2015.