Searchable Encrypted Relational Databases: Risks and Countermeasures

Mohamed Ahmed Abdelraheem[1], Tobias Andersson[1] and Christian Gehrmann[2]

[1] RISE SICS AB
[2] Lund University

Abstract. We point out the risks of protecting relational databases via Searchable Symmetric Encryption (SSE) schemes by proposing an inference attack exploiting the structural properties of relational databases. We show that record-injection attacks mounted on relational databases have worse consequences than their file-injection counterparts on unstructured databases. Moreover, we discuss some techniques to reduce the effectiveness of inference attacks exploiting the access pattern leakage existing in SSE schemes. To the best of our knowledge, this is the first work that investigates the security of relational databases protected by SSE schemes.

1 Introduction

One of the practical solutions for searching on encrypted data is provided by Searchable Symmetric Encryption (SSE) schemes. The very first such scheme was proposed by Song et al. in [19]. Later, Curtmola et al's [12] introduced two security notions for SSE schemes, namely, the non-adaptive semantic security definition and the adaptive semantic security definition. Subsequent SSE schemes [8,7,6] are all based on Curtmola et al's security model. The price of the efficiency offered by SSE schemes comes at the cost of leaking the frequency of each keyword after it has been queried. This makes them vulnerable to inference attacks [14,5] which recovers the issued queries by combining its access pattern leakage by background information about the protected dataset. Most of the proposed SSE schemes are designed to protect unstructured document datasets such as emails or a backup of any sensitive files. However, recently two SSE schemes proposed by Cash et al. [7,6] are designed to efficiently run and protect relational databases where they achieved a query speed comparable to the unprotected MySQL (release 5.5) [7,6].

In addition to SSE schemes, there are several practical solutions proposed to execute SQL queries on an encrypted database. Recently, Popa

et al. proposed CryptDB as a solution to protect confidentiality for applications using SQL databases [18]. CryptDB uses column-level encryption to encrypt the database tables. To enable equality searches, CryptDB uses deterministic encryption. Order preserving encryption (OPE) is used to enable range and comparison queries on encrypted data. OPE is the weakest encryption scheme used in CryptDB whose design concept is based on the trade-off between functionality and confidentiality. Recently, Naveed et al. [17] mounted inference attacks that recovered the plaintext from CryptDB's columns protected by deterministic encryption and order preserving encryption schemes.

Another line of research in preserving database privacy is achieved by distributing and fragmenting the database table across two or more servers [2,9,11] using vertical fragmentation where the table's columns are partitioned across the servers. Privacy in the scheme proposed in [2] is provided under the assumption that the two cloud servers are unable to communicate directly with each other. However, privacy in the work proposed in [9,11] is achieved without this assumption where encryption is used as little as possible and fragmentation is used to provide security by breaking the associations among the attributes and also to provide functionality by keeping most attributes in plaintext. In the scheme proposed in [10], privacy is preserved by using fragmentation only and no encryption is employed. This comes at the cost of saving sensitive data in the clear at the data owner, i.e. the client. All these privacy constraints schemes provide efficiency and functionality but at the cost of having plaintext fields and only encrypting sensitive data which makes them vulnerable to be attacked by an adversary with background information about the database. Also adding or modifying a record reveals the relation among the fragments to passive adversary monitoring the fragments.

Comparing the above methods for searching on encrypted data in terms of security, one can see that SSE schemes offer better security than deterministic or order preserving encryption schemes since they do not leak the frequency of a keyword before querying it. They also provide better security than the data fragmentation method via privacy constraints [9] since they encrypt all the plaintext data and they can also securely manage a dynamic database [6].

However, SSE schemes suffer from leaking the access pattern of a queried keyword which make them vulnerable to inference attacks as demonstrated by Islam et al. [14] and Cash et al. [5]. In this paper, we study the effect of these inference attacks as well as the recent file-injection attacks [21] on relational databases. We also propose a suitable inference

control to safeguard relational databases secured via SSE schemes from being completely recovered by strong adversaries with background knowledge about the relational database.

Our Contribution. We exploit the properties of relational databases and propose an inference attack [14,5] targeting relational databases. We also study the injection attacks [21] in the context of relational databases protected via SSE schemes. We propose the use of privacy constraints [2,9,11] to distribute the encrypted index of an SSE scheme into several fragments or servers to reduce the effectiveness of inference attacks exploiting the access pattern leakage [14,5] which is inherent in SSE schemes. Note that the privacy constraints as defined in [2,9,11] were mainly used to depart completely from the use of encryption or to use encryption as less as possible. However, in this paper we propose using them to strengthen the security of SSE schemes against inference attacks.

Related Work. Query recovery attacks exploiting the access pattern leakage of SSE schemes were proposed by Islam et al. [14] and recently improved by Cash et al. [5]. Both attacks assume background knowledge in the form of joint frequencies between keywords and were proposed mainly to deal with unstructured datasets. However, in this work we propose an inference attack called Relational-Count that uses only knowledge about the frequency of keywords in the target relational database. We show that this might lead to complete record-recovery attacks. Moreover, we propose the use of privacy constraints as an additional countermeasure that should be used together with padding to reduce the effectiveness of inference attacks. The recently proposed file-injection attack by Zhang et al. [21] recovers only a set of keywords in encrypted document and could be prevented by limiting the content of each injected file. However, in this paper we show that record-injection attacks have more severe consequences than file-injection attacks since they can achieve full record recovery on relational databases protected via SSE schemes. Moreover, record-injection attacks cannot be simply prevented by limiting the content of an injected record as done to prevent file-injection attacks [21] since that would hinder the addition of complete records to the protected relational databases.

Organization of the paper. Section 2 gives a brief overview about SSE schemes. In Section 3, we give a brief overview about inference attacks and propose a new inference attack targeting relational databases. In Section 4, we point out the security risks of protecting relational databases via SSE schemes where we show that inference attacks and

record-injection attacks can fully recover a significant number of database records. In Section 5, we propose the use of privacy constraints as an inference control and countermeasure to reduce the risk of inference attacks.

2 Background

Definition. An SSE scheme takes as inputs a plaintext database index together with the client's secret keys and outputs an encrypted and frequency-hiding database index where the keywords are encrypted using a deterministic encryption algorithm and the document/records identifiers are encrypted using a randomized algorithm. When the SSE-protected database is a relational database (i.e. searchable encrypted relational databases), a keyword w_i will represent an attribute-value pair which points to a cell in the relational database (i.e $w_i = (attribute_i : v_i)$ where $attribute_i$ refers to the column name or attribute name and v_i refers to the value of the attribute). All recent SSE schemes follow the adaptive security definition proposed by Curtmola et al. [12] where security is achieved against an honest-but-curious server.

Leakage Profile. An SSE scheme leaks the *access pattern*: the result size of the query and the document/record IDs corresponding to the queried keyword w_i and also leaks the *search pattern*: the fact that whether two searches are the same or not.

Passive Attacks on SSE Schemes. Two passive attacks against SSE schemes exploiting the access pattern leakage have been proposed recently by Islam et al. [14] and later developed by Cash et al. [5]. These passive attacks are inference attacks mounted by an honest-but-curious server who knows the distribution of the dataset under attack or knows a significant number of the client's plaintext documents.

Active Attacks on SSE Schemes. Another class of attacks outlined by Cash et al. [5] are the chosen-document attacks and the chosen-query attacks. Both attacks are mounted by an active adversary who is somehow capable of deceiving the client into including her own chosen-document into the documents set as well as into choosing her favorite queries respectively. Recently, Zhang et al. [21] presented a concrete description of a chosen-document attack (file-injection) where the attacker is able to recover all the queries without any prior knowledge about the client's dataset under attack. The equivalent of file-injection in the context of searchable encrypted relational databases is record-injection and it has worse consequences that go beyond query recovery, namely, full record recovery or partial record recovery (cf. Section 4.2).

3 Inference Attacks

Inference attacks are mounted on SSE schemes to recover the plaintext of encrypted keywords involved on previous queries issued by the client and observed by the attacker. This kind of attack is called *query recovery* and was proposed by Islam, Kuzu and Kantarcioglu (IKK) in [14]. Their attack, known in the literature as the IKK attack, targets the strongest kinds of SSE schemes which are those proved to be secure under the adaptive security definition. The IKK attack models the problem of recovering the unknown keywords as an optimization problem solved using a simulated annealing algorithm. Recently, Cash et al. [5] improved the IKK by proposing another inference attack, called the *Count attack*, that is simpler and more efficient than the IKK attack. The Count attack assumes knowledge about the joint frequency of any two keywords as well as the frequency of each keyword in the dataset under attack.

However, in this section, we propose an inference attack that assumes only that the attacker knows the frequency of each keyword (attribute-value pair). We call our attack, the *Relational-Count attack*. Our attack targets relational databases by exploiting their structural properties. First we describe the Count attack and then describe the Relational-Count attack.

The Count Attack. The Count attack [5] assumes knowledge about the *joint frequency (or co-occurrence count)* of any two plaintext keywords $w_i, w_j \in \mathsf{W}$ where W is the set of all unique keywords in the target database. It also assumes knowledge about the occurrence of each keyword w (attribute-value pair) over all the database documents/records, say $size(w)$. Assume that the number of unique keywords (or attribute-value pairs) indexed by an SSE scheme is m. Then the attacker uses the joint frequency knowledge to construct an $m \times m$ matrix M where its entry $M[i, j]$ holds the co-occurrence value or the joint frequency of having the ith unique keyword and jth unique keyword together in the database indexed by the SSE scheme under consideration. The constructed matrix M represents the background knowledge of the attacker about the encrypted database. Using the access pattern leakage, the attacker also constructs another matrix C that represents the observed joint frequency between any two queries intercepted by the attacker. Obviously, any keyword w with unique result size will be easily recovered when queried since the size of the result set of its query q, $size(q)$, will be known to the attacker who will use the knowledge about the frequency or occurrence of each keyword w to find the keyword whose frequency matches the result

size of q (i.e. find a keyword w such that $size(q) = size(w)$). The unique counts approach will recover all the queries corresponding to keywords with unique result sizes which can be significant in some databases. A query q with non-unique result size is recovered by creating a candidate list consisting of all keywords with the same result size as q and then discarding the wrong keywords from the candidate list by comparing the observed joint frequency between q and any previously recovered query q_k with the prior known joint frequency between c_i and w_k where c_i is the candidate keyword under consideration and w_k is the recovered keyword corresponding to q_k. If they are unequal (i.e. $C[q, q_k] \neq M[c_i, w_k]$), then c_i will be discarded from the candidate list.

The Relational-Count Attack. Similar to the Count attack, our attack assumes that the attacker has knowledge about the frequency of each attribute-value pair which allows the attacker to recover the queries with unique result sizes and to add a candidate list for each query with non-unique result size. However, unlike the Count attack, we do not assume that the attacker has any knowledge about the joint frequencies between attribute-value pairs. Instead, we use the following simple observation about the structural properties of relational databases to filter out the wrong candidates.

Observation 1. *If the joint frequency (or the co-occurrence count) between any two different queries (tokens) is non-zero, then their corresponding attribute names are different.*

In other words, the joint frequency between two queries with the same attribute name is zero. The observation should be clear from the fact that each relational database record has only one value for each column or attribute name. This observation allows us to reduce the list of candidates for a given query with non-unique result size. Assume that there is an unknown query q_i with non-unique result size, then one can see that a candidate keyword $w_j = (a_j : v_j)$, where a_j is the attribute name and v_j is its value, will be discarded if there is a previously recovered query q_k whose attribute name is a_j and whose joint frequency with q_i is non-zero. In other words, If $C[q_i, q_k] \neq 0$, then discard w_j, where C is the observed joint frequency matrix. This is because the right candidate keyword must have the same attribute name as the unknown query q_i. Next, we apply the Count attack and the Relational-Count attack on searchable encrypted relational databases.

4 Risks of Using Searchable Encrypted Relational Databases

In this section, we point out three risks that might occur when protecting relational databases via SSE schemes. The first risk is breaking query privacy by a passive adversary with knowledge about the target relational database represented in the form of joint frequencies between attribute-value pairs and the frequency of each attribute-value pair (i.e. knowledge of all the database records). This means that the attacker here is only interested in recovering queries to break query privacy. The second risk is the possibility of complete record recovery attack by a passive adversary who has knowledge about the frequency distribution of attribute-value pairs. The third risk is the possibility of complete record recovery via record-injection attacks performed by an active adversary with no prior knowledge about the target database.

4.1 Query and Record Recovery Attacks

One can see that inference attacks on searchable encrypted relational databases will probably lead to a partial or full record recovery since query recovery directly translates to partial record recovery in this case. Thus, inference attacks that do not require the knowledge of all the database records but only the frequency distribution of attribute-value pairs, such as our Relational-Count attack, are more interesting from the attacker's point's of view since they extend the attacker's goal from only breaking query privacy by performing query recovery as done in the Count attack to gaining new knowledge through partial or full record recovery.

In the following, we apply the Count and Relational-Count attacks on searchable encrypted relational databases. Our goal is to firstly break query privacy using the Count attack which employs a strong attacker who knows almost all the database records (i.e. so the attacker is able to compute joint frequencies between attribute-value pairs and frequencies of attribute-value pairs) and then secondly recover complete records using our Relational-Count attack which employs a weaker attacker who knows only the frequency distribution of the attribute-value pairs of the target database.

Data Sets. We mount the Count attack over the Census dataset [16], the Bank dataset [3] and the Adult dataset [15]. These are real world datasets from the UCI Machine Learning Repository [13]. The Census dataset consists of 299285 records, 40 columns, and 3993 distinct attribute-value

pairs when we exclude the 8 missing attribute-value pairs (in total there are 4001 attribute-value pairs but we do not consider the empty or missing values in our keywords set). The Bank dataset consists of 4521 records, 17 columns, and 3720 distinct attribute-value pairs. The Adult dataset consists of 32561 records, 14 columns, and 498 distinct attribute-value pairs.

Query Generation. We used a standard single-keyword SSE scheme where a Bitmap encrypted index is used similar to the single-keyword Bitmap index scheme described in [1,20,14] to transform each target relational database into a separate searchable encryption relational database. We conducted four experiments per each target dataset where different sets of random queries within the target protected database are issued in each experiment. The access pattern leakage (result size + retrieved record IDs) of each query is intercepted by a passive attacker who combines this knowledge with background knowledge about the dataset and then execute the Count or the Relational-Count attacks described above to recover the queries.

Query Recovery via the Count Attack. Table 1 shows the query recovery results on the three datasets. Three experiments are conducted for each dataset. One can see that when the number of issued queries increases, the rate of query recovery also increases. For example, the Census dataset contains 3993 unique attribute-value pairs and only 77.60% are recovered when 600 queries are issued. But when all queries are issued as shown in the 3rd experiment, we see that a high percentage of queries ($\approx 98.1\%$) are recovered in the Census dataset. So query privacy is completely broken by the Count attack. Note that the record recovery rate will be high but the attacker here is only concerned about query recovery and not record recovery since we assume that the attacker already knows all the database records.

Table 1: Query Recovery Results on Different Relational Databases. All queries are issued in the 3rd experiment. Results are averaged over 3 tests, where queries are chosen randomly, in the 1st and 2nd experiments. 455/600 indicates an average of 455 queries out of 600 issued queries were recovered successfully using the Count attack.

Exp. No	**Census Data Set**	**Bank Data Set**	**Adult Data Set**
1	455/600	300/600	135/150
2	1432/1500	1237/1500	230/250
3	3917/3993	3460/3720	466/498

Record Recovery via the Relational-Count Attack. We applied our attack on the three relational databases described above. Table 2 shows that our attack recovers only 757 queries out of 3993 queries whereas the Count attack recovers 3917 queries out of 3993 queries. Thus, our attack is not as effective as the Count attack. However, our attack assumes that the attacker only knows the frequency distribution of the attribute-value pairs which is a more realistic assumption that could hold in practice. Table 3 shows that the 757 queries recovered in the Census dataset allowed us to recover 447 complete records in the Census dataset whereas the 122 queries recovered in the Bank dataset allowed us to recover only recover 1 complete database record in the Bank dataset. The Bank dataset also has 989 records where 64.99-52% (9-11 attributes) of the attributes are recovered. The results on the Census displayed in Table 3 show that even recovering as few as 757/3993 queries allows us to learn a high percentage of records in the Census data set. For example 109433 records are recovered where only between 2 and 6 attributes are missing. This could enable a strong attacker that uses machine learning to predict the missing values given the 757 records that are fully recovered. We have not investigated this possibility and we leave it for future work.

Note that we can recover only 149 complete records from the Census dataset after recovering only the 531 attribute-value pairs with unique result sizes compared to recovering 447 complete records when we resolve the attribute-value pairs with non-unique result sizes using our Relational-Count attack. This shows the effect of our Relational-Count attack in recovering complete database records.

When each column in the Adult, Bank and Census datasets, is encrypted using deterministic encryption, then an attacker with frequency knowledge about the attribute-value pairs frequency distribution can only recover 272/498, 152/3720 and 821/3993 attribute-value pairs only respectively which is very close to the number of queries (236/498, 122/3720 and 757/3993 shown in Table 2) recovered by our Relational-Count attack when the same three datasets are encrypted using an SSE scheme. Therefore, we see our Relational-Count attack as one step toward closing the security gap between searchable encrypted relational databases and column-based deterministically encrypted relational databases employing systems such as CryptDB.

4.2 Record-Injection Attacks

Zhang et al.'s non-adaptive file-injection binary attack [21] cannot be applied exactly in searchable encrypted relational databases since we are

Table 2: The table shows the total number of queries recovered in three relational databases when all the queries are issued and the Relational-Count attack is used. The unique recovery column shows the number of queries recovered whose result sizes are unique and the non-unique recovery column shows the number of queries whose result sizes are non-unique which are recovered using the Relational-Count attack.

Data Set	**unique recovery**	**non-unique recovery**	**total**
Adult dataset	155	81	236/498
Bank	97	25	122/3720
Census	531	226	757/3993

Table 3: The table shows the number of records recovered with the percentage of attributes recovered (record recovery rate) in each record when all queries are executed in the SSE-protected Bank and Census datasets together. 100% Rec. means all attributes are recovered, 95-85% Rec. means all attributes are recovered except 5-15% of them have not been recovered, and so on. The entry 708/4521 indicates that around 84.99-74% attributes (i.e. 13-14 attributes out of 17 attributes) of 708 records out of 4521 records of the Bank dataset are recovered. Note that the bank dataset has 989 records where 64.99-52% (i.e. 9-11 attributes) of the attributes are recovered which is not shown in the table. 4521 $\equiv$ total number of records for the Bank dataset and 299285 $\equiv$ total number of records for the Census dataset.

Data Set	**100% Rec.**	**95-85% Rec.**	**84.99-74% Rec.**	**73.99-65% Rec.**
Bank	1/4521	39/4521	997/4521	2495/4521
Census	447/299285	109433/299285	131253/299285	58152/299285

dealing with a structured text governed by a relational database rather than unstructured text. Depending on the relational database under attack, there might be a large number of records needed to be injected in order to recover all the possible encrypted queries if the database contains attributes whose values are variables (not discrete) with a big range. However, if the attacker is concerned about a small subset of attribute-value pairs. Then the attacker can inject a number of records by focusing on some attributes whose values are discrete with a small range as noted in [21]. This will reduce the number of injected records and will lead to query recovery and consequently partial record recovery without any prior knowledge.

We focus here on non-adaptive record injections as adaptive injection attacks need background knowledge about the target encrypted database and they can be prevented by using a forward secure SSE scheme [4]. Similar to Zhang et al. [21], we assume that the attacker can identify the record ID of each injected record. Let D be a relational database with n records and m attributes or columns where each attribute is denoted by a_i and its cardinality is denoted by $|a_i|$, $1 \leq i \leq m$. Assume that the

number of records need to be injected in D in order to cover the whole attribute-value pair space $|\mathsf{W}|$ or a target subset of attribute-value pairs S is l. Suppose that $R = r_1 r_2 \cdots r_l$ is the search result on the injected records regarding an observed query q, where $r_i = 1$ iff the ith injected record is part of the result set of the query q, otherwise $r_i = 0$. Clearly $l \geq |a_i|$ for all i, otherwise the injected records will not recover all the values of the attribute a_i. Assume that there are t attributes $(a_1, \cdots, a_t)$ with the same cardinality d, then in order to cover all the $d \cdot t$ values one can construct a mapping that assigns each attribute-value pair to a unique search result string on the injected records by simply injecting $d \cdot t$ records as follows. Let the first d records contain all the values of the 1st attribute (i.e. $a_{11}, a_{12}, \cdots, a_{1d}$) and the other attributes belonging to S are empty. Also, let the second d records contain all the values of the 2nd attribute (i.e. $a_{21}, a_{22}, \cdots, a_{2d}$) and the other attributes belonging to S are empty and so on until the last and tth d records contain all the values of the tth attribute and the other attributes belonging to S are empty. Now one can see that the search result on the injected records regarding any attribute-value pair in S will yield a binary string with Hamming weight one where the location of the i-th active bit in the binary string indicates that the attribute value is located at position $i \mod d$ (or last position if $i \mod d = 0$) in the $\lceil i/d \rceil$th attribute.

However, one can inject l records, where l is much less than $d \cdot t$, by injecting t columns of length l where the search results of all the attribute-value pairs in a single column have active bits at different positions and all the possible search results representing all attribute-value pairs are disjoint. In other words, we need to separate the l-bit binary strings where each l-bit corresponds to a search result into t disjoint sets where each set S_j has d elements representing the d attribute-value pairs of the jth column (note that all the t columns have the same cardinality). Each element in S_j is an l-bit binary string representing a search result of an attribute-value pair belonging to the jth column. Assume that all the elements of S_j have the same Hamming weight w_j. Assume also that d divides l. Then, w_j must be $\leq l/d$ in order to have a valid set S_j representing unique search results. The number of l-bit binary strings of Hamming weight i is $\binom{l}{i}$. Therefore, one can look for the smallest l satisfying the inequality $\binom{l}{l/d} + \cdots + \binom{l}{1} \geq d \cdot t$ and the same time construct a one-to-one mapping between the l-bit binary strings of S_j representing the unique search results and the injected columns C_j of length l for all $1 \leq j \leq t$.

Concrete Example. The Census dataset discussed above, has 7 attributes with cardinality 3. By setting $t = 7$ and $d = 3$ in the above inequality, one can see that $l = 6$ is the smallest integer to satisfy it. Now we need to divide all the 6-bit binary vectors with Hamming weight ≤ 2 (since $l/d = 6/3 = 2$) into 7 disjoint sets where each set S_j holds three (since $d = 3$) 6-bit binary vectors that have different active positions. The elements of S_j must contain binary vectors of Hamming weight two or one as otherwise we will not be able to cover all the $d \cdot t = 3 \cdot 7 = 21$ attribute-value pairs. One can see that $S_1 = \{100000, 010000, 001000\}$ is a set whose binary strings can represent the search results corresponding to the attribute-value pairs injected in the column $C_1 = \begin{bmatrix} a_{11} \; a_{12} \; a_{13} \; ? \; ? \; ? \end{bmatrix}^T$ where '?' means an empty entry and T is the transpose operator. Let C_1 be the first column injected in our records. When one searches for a_{11}, a_{12} and a_{13} the search results on the injected records will be 100000, 010000 and 001000 respectively. Now, from S_1 we could generate another valid search results set S_2 by looking at the other possible binary strings of Hamming weight one representing valid search results. There are exactly 3 other possible binary strings, namely, 000100, 000010 and 000001. So $S_2 = \{000100, 000010, 000001\}$ and it represents the search results corresponding to the attribute-value pairs injected in the column $C_2 = \begin{bmatrix} ? \; ? \; ? \; a_{21} \; a_{22} \; a_{23} \end{bmatrix}^T$. Let C_2 be the second column injected in our records. Now we consider search results whose binary strings have Hamming weight 2 such as $S_3 = \{110000, 001100, 000011\}$ which represents the attribute-value pairs injected in the column $C_3 = \begin{bmatrix} a_{31} \; a_{31} \; a_{32} \; a_{32} \; a_{33} \; a_{33} \end{bmatrix}^T$. Let C_3 be the third column injected in our records. Thus we have three sets and we need to construct another four sets in order to obtain seven sets that cover all the search results of the 21 attribute-value pairs. One set S_3 has three binary strings of Hamming weight 2. So there remains another $\binom{6}{2} - 3 = 15 - 3 = 12$ binary strings of Hamming weight 2 and we need to divide them into 4 sets S_4, S_5, S_6 and S_7 where each set has 3 elements which have different positions for the active bits similar to S_3. Note that each binary string in S_3 have four non-active bits, so interchanging the location of an active bit with the location of a non-active bit will yield a new binary string but we need to interchange the location of active bits with the location of non-active bits for all the binary strings of S_3. So the four sets, S_4, S_5, S_6 and S_7, can be obtained by permuting the locations of active bits within each element in S_3. To do so, we need to consider the elements of S_3 as columns of a matrix M_3 where the first column is $\begin{bmatrix} 1 \; 1 \; 0 \; 0 \; 0 \; 0 \end{bmatrix}^T$, the second column is $\begin{bmatrix} 0 \; 0 \; 1 \; 1 \; 0 \; 0 \end{bmatrix}^T$ and

the last column is $\begin{bmatrix}0\,0\,0\,0\,1\,1\end{bmatrix}^T$. In each column of M_3, there are four locations to move the active bits in order to get a new valid and unique column. However, we want all the new columns to have active bits at different positions in order to form valid and unique search results (e.g. S_i $i \geq 4$). To construct the set S_4, we change the positions of the active bits in M_3 by multiplying it by a 6×6 permutation matrix, P_{π_4} where the first and third rows of the 6×6 identity matrix I_6 are permuted and also the second and fifth rows are permuted. In cyclic notation, the permutation π_4 can be written as follows $\pi_4 = (13)(25)$. Now, $M_4 = P_{\pi_4} \times M_3$. The first column of M_4 is $\begin{bmatrix}0\,0\,1\,0\,1\,0\end{bmatrix}^T$. The second column is $\begin{bmatrix}1\,0\,0\,1\,0\,0\end{bmatrix}^T$ and the last column is $\begin{bmatrix}0\,1\,0\,0\,0\,1\end{bmatrix}^T$. The columns of M_4 form valid search results and thus $S_4 = \{001010, 100100, 010001\}$. Its corresponding injected column is $C_4 = \begin{bmatrix}a_{41}\,a_{42}\,a_{43}\,a_{41}\,a_{43}\,a_{42}\end{bmatrix}^T$. Similarly, we form another 6×6 permutation matrix, P_{π_5} by interchanging the first and fourth rows and also interchanging the second and the sixth rows in the identity matrix I_6. In cyclic notation, the permutation π_5 can be written as follows $\pi_5 = (14)(26)$. The columns of $M_5 = P_{\pi_5} \times M_3$ gives us $S_5 = \{0000101, 101000, 010010\}$. Its corresponding injected column is $C_5 = \begin{bmatrix}a_{51}\,a_{52}\,a_{51}\,a_{53}\,a_{52}\,a_{53}\end{bmatrix}^T$. S_6 is obtained from the columns of $M_6 = P_{\pi_6} \times M_3$ where P_{π_6} is a permutation matrix obtained by interchanging the rows of the identity matrix corresponding to the cyclic permutation $\pi_6 = (15)(24)$. $S_6 = \{000110, 011000, 100001\}$ and its corresponding injected column is $C_6 = \begin{bmatrix}a_{61}\,a_{62}\,a_{62}\,a_{63}\,a_{63}\,a_{61}\end{bmatrix}^T$. Finally, S_7 is obtained from the columns of $M_7 = P_{\pi_7} \times M_3$ where P_{π_7} is a permutation matrix obtained by interchanging the rows of the identity matrix corresponding to the cyclic permutation $\pi_7 = (16)(23)$. $S_7 = \{001001, 010100, 100010\}$ and its corresponding injected column is $C_7 = \begin{bmatrix}a_{71}\,a_{72}\,a_{73}\,a_{72}\,a_{71}\,a_{73}\end{bmatrix}^T$. Thus, we can inject only $l = 6$ records instead of $l = d \cdot t = 3.7 = 21$ records in order to cover all the 7 attributes with the same cardinality 3 in the Census dataset. Table 4 shows the injected 6 records formed by injecting 7 columns yielding unique search results.

Discussion. It is clear that once an attacker is able to inject records then record-injection will lead to query recovery and eventually could lead to full record recovery. The above record injection attack works under the assumption that the attacker can identify the record identifiers of the injected records. This assumption was also adopted by Zhang et al. [21]. If the client updates one record at a time, then the attacker will always be able to identify the identifier of an injected records based on the time

Table 4: The table shows that for each attribute-value a_{ij} we have a unique search result on the injected records. Injecting 6 records to cover 7 attributes each with 3 attribute-value pairs. A query for the attribute-value a_{21} will yield the search result 000100 on the injected records while a query for the attribute-value a_{71} will yield the search result 100010.

No	a_1	a_2	a_3	a_4	a_5	a_6	a_7
1	a_{11}	?	a_{31}	a_{41}	a_{51}	a_{61}	a_{71}
2	a_{12}	?	a_{31}	a_{42}	a_{52}	a_{62}	a_{72}
3	a_{13}	?	a_{32}	a_{43}	a_{51}	a_{62}	a_{73}
4	?	a_{21}	a_{32}	a_{41}	a_{53}	a_{63}	a_{72}
5	?	a_{22}	a_{33}	a_{43}	a_{52}	a_{63}	a_{71}
6	?	a_{23}	a_{33}	a_{42}	a_{53}	a_{61}	a_{73}

at which it is stored in the server. However, if the client does only batch updates, then an attacker could inject records in a certain way such that each attribute-value pair has a unique number of appearances in all the injected records. The file-injection countermeasure proposed by Zhang et al. [21] which restricts the number of keywords per document to a certain threshold T (e.g. $T \ll |\mathsf{W}|/2$) cannot be applied here since a relational database record has a certain number of keywords (attribute-value pairs) equivalent to its number of attributes and any restriction would hinder the work of any application using the searchable encrypted relational database. So one needs to use a forward secure SSE scheme such as the one proposed in [4] in order to protect relational databases against adaptive injection attacks (Note that the above described attacks are non-adaptive attacks but an adaptive injection attacks similar to the one proposed in [21] can easily be realized).

5 Countermeasures Against Attacks on SSE Schemes

Countermeasures against inference attacks must be used in order to reduce the their effectiveness. A well known technique is *padding* which is proposed in [14,5] as a potential countermeasure to reduce the effectiveness of inference attacks. Basically, during the setup of the encrypted database, the client adds dummy record (or document) IDs to each attribute-value pair (or keyword) in the index in order to hide the actual frequency of the keyword. Also, the client adds an encrypted dummy record (or document) corresponding to each dummy ID added in the index. Later, during search, the client filters out the dummy records (or documents). Experiments in [5], show that a padding level that increases the index size by 15% for a real world sample dataset and 30% for an-

other real world sample dataset, does not affect the success rate of the generalized Count attack [5] which is a slight improvement of the Count attack. It basically does not depend on resolving queries with unique frequency which will not exist in a padded SSE scheme but it initially guesses these queries. The detection of a wrong guess is done during the co-occurrence testing phase which does equality matches in a window or a range of a fixed size to nullify the noise coming from the dummy records (or documents) causing false co-occurrence count values. Thus, the generalized Count attack presented in [5] suggests that padding alone does not reduce the effectiveness of inference attacks as matches in a range can be done through the observed co-occurrence matrix. The effect of our Relational-Count attack can be reduced by padding to reduce the number of attribute-value pairs with unique result sizes. So we focus here on preventing the Count attack.

Countermeasure Against the Count Attack. In addition to reducing the effectiveness of the actual query result size by padding, one might think of reducing the effectiveness of the queries' observed joint frequency matrix C by forcing the observed joint frequency between some queries to be zero. One can see that if q_i and q_j are distributed in different fragments according to a defined privacy constraints, then $C[q_i, q_j]$ will be zero when each query is executed in only one fragment and the fragments are not allowed to interact with each other to evaluate any query. Now an equality match or a window equality match with the joint frequency knowledge-matrix $M[s_i, t_j]$ as done in [5] will never happen which will significantly reduce the effectiveness of the Count attack. This can be done by applying vertical fragmentation to a relational database table according to a predefined set of *privacy constraints* on its columns.

The aim behind the privacy constraints is hiding the association among the attributes which means that there should be no joint appearance of the attributes in the privacy constraints [2,9,11]. For example, consider the relation of the following attributes about patients in a hospital: Name, Date of Birth (DOB), Disease, Medical Doctor (MD), and ZIP. Now one can define the following privacy constraints c_1 = {Name, DOB}, c_2 = {Name, Disease}, c_3 = {Name, ZIP}, c_4 = {Name, MD}, c_5 = {DOB, ZIP, Disease} and c_6 = {DOB, ZIP, MD}. Now a privacy constraint prevents some columns from being together, so the privacy constraint $c_1 = \{\mathsf{Name}, \mathsf{DoB}\}$ prevents the Name column or attribute from being together in one fragment with the DoB column since they might reveal together more information about a specific person if one of them is recovered using an inference attack. We note that the privacy constraints

should be used to produce the minimal amount of fragments possible using the heuristic algorithm proposed in [9]. For instance, a valid minimal fragmentation for the relation and privacy constraints defined above is the following $\mathcal{F}$ ={$\mathcal{F}_1$={Name}, $\mathcal{F}_2$ = {DoB,ZIP}, $\mathcal{F}_3$ ={Disease,MD}}. After applying the fragmentation, we need to ensure that each query is executed in only one fragment in order to prevent an attacker monitoring all the fragments (or collaborative honest-but-curious fragment servers) from gaining any information about the correlation of the records between any two fragments which will obviously break the pre-defined the privacy constraints set by the data owner. Moreover, we need to have different record IDs for the same original record at each fragment in order to achieve security against an attacker monitoring all fragments (or collaborative honest-but-curious fragment servers) and also apply secure random shuffling for the fragment's records. After that, we can apply the same SSE scheme in each fragment using a different key. This ensures that applying inference attacks on each fragment is not effective since the encrypted attributes within each fragment does not provide sufficient information if they are recovered. Note that the generalized Count attack is effective in each fragment and it could probably recover entire records in each fragment. However, the fragments are defined according to the privacy constraints which means that the recovered records are unlinkable and thus will not reveal useful information. If all fragments are recovered, the attacker will not be able to link or combine them to recover the original record before fragmentation. This is because each fragment is shuffled differently and each fragment's record has a different record ID secretly pointing to the same original record.

Security Gain. Vertical fragmentation using privacy constraints prevents full record recovery but the fragmented SSE scheme will still leak the access pattern inside each fragment as well as leaking the attribute of the queries in each fragment. We performed one experiment to show the security gain when we employ vertical fragmentation via privacy constraints. We split the Bank dataset into three fragments. The first fragment contains five attributes, namely, "age", "job", "marital", "education" and "duration". The second fragment contains seven attributes, namely, "default", "balance", "housing", "loan", "contact", "day" and "month". The third fragment contains five attributes, namely, "campaign", "pdays", "previous", "poutcome" and "y". The Count attack was not able to resolve 130/961 queries in the first fragment, 1264/2405 queries in the second fragment, and 105/354 queries in the third fragment. In total, there are 1499/3720 queries that have not been recovered in all the three frag-

ments compared to only, 3620-3460 = 260, queries that have not been recovered when vertical fragmentation is not employed as shown in Table 1 where 3460/3720 queries are recovered. The increased number of unresolved queries after fragmentation shows the impact of vertical fragmentation as an effective countermeasure in reducing the strength of the Count attack. We note here that in some scenarios, vertical fragmentation alone might not be enough to prevent the Count attack, for example in a small dataset such as the Adult dataset, the Count attack will always recover most of the fragments and the security gain will only be in preventing the attacker from linking the recovered records and combine them to recover one or more original records. Such a gain can be useful to reduce the effect of the Relational-Count attack where record recovery is the one of the attacker's goals. However, an attacker who is able to perform the Count attack is concerned only about query recovery and not record recovery. Therefore, we need to employ padding also as an additional countermeasure needed to reduce the effectiveness of the Count attack.

Privacy Constraints vs. Record Injection Attacks. Privacy constraints can not prevent record injection attacks but they can reduce their effectiveness since the attacker will not be able to re-construct and combine the recovered fragments to form the original plaintext records even if all the records of each fragment are recovered since each fragment has a different record ID secretly pointing to the same original record before fragmentation. Note that an attacker injecting records that do not have any prior knowledge about the target database will not be able to link and combine the fragmented records in case all fragments are recovered.

Performance Gain. The drawback with the fragmentation approach is in the extra computational work done when a multi-keyword query whose keywords or attribute-value pairs exist in different fragments. But this can be improved using a fragmentation algorithm which takes usage data into account [11] which will allow us to find a suitable minimal fragmentation providing efficient execution for multi-keyword queries and the same time meeting the security demands set by the privacy constraints.

6 Conclusion

In this paper, we pointed out that inference and record-injection attacks pose a real threat to searchable encrypted relational databases. We proposed the use of privacy constraints together with padding on top of any

SSE scheme in order to reduce the effectiveness of the inference attacks proposed in [14,5].

References

1. Mohamed Ahmed Abdelraheem, Christian Gehrmann, Malin Lindström, and Christian Nordahl. Executing boolean queries on an encrypted bitmap index. In *Proceedings of the 2016 ACM on Cloud Computing Security Workshop*, pages 11–22. ACM, 2016.
2. Gagan Aggarwal, Mayank Bawa, Prasanna Ganesan, Hector Garcia-Molina, Krishnaram Kenthapadi, Rajeev Motwani, Utkarsh Srivastava, Dilys Thomas, and Ying Xu. Two can keep a secret: A distributed architecture for secure database services. *CIDR 2005*, 2005.
3. Portuguese banking institution. Bank marketing data set. https://archive.ics.uci.edu/ml/datasets/Bank+Marketing, 2014. [Last Accessed June 2017].
4. R. Bost. $\sigma o \varphi o \varsigma$: Forward secure searchable encryption. In *CCS 2016*.
5. David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. Leakage-abuse attacks against searchable encryption. In *Proceedings of CCS 2015*.
6. David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit S Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. Dynamic searchable encryption in very-large databases: Data structures and implementation. *IACR Cryptology ePrint Archive*, 2014:853, 2014.
7. David Cash, Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel-Cătălin Roşu, and Michael Steiner. Highly-scalable searchable symmetric encryption with support for boolean queries. In *Advances in Cryptology–CRYPTO 2013*, pages 353–373. Springer, 2013.
8. Melissa Chase and Seny Kamara. Structured encryption and controlled disclosure. In *Advances in Cryptology-ASIACRYPT 2010*, pages 577–594. Springer, 2010.
9. Valentina Ciriani, Sabrina De Capitani Di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati. Fragmentation and encryption to enforce privacy in data storage. In *Computer Security–ESORICS 2007*.
10. Valentina Ciriani, Sabrina De Capitani Di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati. Keep a few: Outsourcing data while maintaining confidentiality. In *Computer Security–ESORICS 2009*, pages 440–455. Springer, 2009.
11. Valentina Ciriani, Sabrina De Capitani Di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati. Combining fragmentation and encryption to protect privacy in data storage. *ACM Transactions on Information and System Security (TISSEC)*, 2010.
12. Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 79–88. ACM, 2006.
13. Center for Machine Learning and Intelligent Systems. University of california, irvine. https://archive.ics.uci.edu/ml/datasets.html. [Last Accessed June 2017].
14. Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *NDSS*, 2012.

15. R. Kohavi and B. Becker. Adult data set. https://archive.ics.uci.edu/ml/machine-learning-databases/adult/, 1996. [Last Accessed June 2017].
16. Terran Lane and Ronny Kohavi. Census-income (kdd) data set. https://archive.ics.uci.edu/ml/machine-learning-databases/census-income-mld/, 2000. [Last Accessed June 2017].
17. Muhammad Naveed, Seny Kamara, and Charles V Wright. Inference attacks on property-preserving encrypted databases. In *Proceedings of CCS 2015*.
18. Raluca Ada Popa, Catherine Redfield, Nickolai Zeldovich, and Hari Balakrishnan. Cryptdb: protecting confidentiality with encrypted query processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, pages 85–100. ACM, 2011.
19. Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*, pages 44–55. IEEE, 2000.
20. Peter van Liesdonk, Saeed Sedghi, Jeroen Doumen, Pieter Hartel, and Willem Jonker. *Secure Data Management: 7th VLDB Workshop, SDM 2010*, chapter Computationally Efficient Searchable Symmetric Encryption.
21. Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou. All your queries are belong to us: The power of file-injection attacks on searchable encryption. Cryptology ePrint Archive, Report 2016/172.