Interpreting and Mitigating Leakage-abuse Attacks in Searchable Symmetric Encryption

Lei Xu, Huayi Duan, Anxin Zhou, Xingliang Yuan, Cong Wang

***Abstract*—Searchable symmetric encryption (SSE) enables users to make confidential queries over always encrypted data while confining information disclosure to pre-defined leakage profiles. Despite the well-understood performance and potentially broad applications of SSE, recent leakage-abuse attacks (LAAs) are questioning its real-world security implications. They show that a passive adversary with certain prior information of a database can recover queries by exploiting the legitimately admitted leakage. While several countermeasures have been proposed, they are insufficient for either security, i.e., handling only specific leakage like query volume, or efficiency, i.e., incurring large storage and bandwidth overhead.**

We aim to fill this gap by advancing the understanding of LAAs from a fundamental algebraic perspective. Our investigation starts by revealing that the index matrices of a plaintext database and its encrypted image can be linked by linear transformation. The invariant characteristics preserved under the transformation encompass and surpass the information exploited by previous LAAs. They allow one to unambiguously link encrypted queries with corresponding keywords, even with only partial knowledge of the database. Accordingly, we devise a new powerful attack and conduct a series of experiments to show its effectiveness. In response, we propose a new security notion to thwart LAAs in general, inspired by the principle of local differential privacy (LDP). Under the notion, we further develop a practical countermeasure with tunable privacy and efficiency guarantee. Experiment results on representative real-world datasets show that our countermeasure can reduce the query recovery rate of LAAs, including our own.

***Index Terms*—Encrypted Search, Cryptographic databases, Leakage abuse attack, Linear Algebraic.**

I. Introduction

UNDER active research for years, searchable symmetric encryption (SSE) [1] is considered as a promising building block to enable practical keyword search in encrypted databases [2]. It allows clients to make confidential queries directly over encrypted data, confining information disclosure to reasonably defined leakage profiles. Despite fruitful progress on enriched query types [3]–[5] and improved efficiency [6], it is concerned that SSE is still not ready for deployment, because the security implications of its leakage have not been adequately understood.

L. Xu is with the School of Mathematics and Statistics, Nanjing University of Science and Technology, Nanjing, 210094, China. E-mail: xulei-crypto@gmail.com.

H. Duan, A. Zhou, C. Wang are with the Department of Computer Science, City University of Hong Kong, Hong Kong SAR, China, and are also with the City University of Hong Kong Shenzhen Research Institute, Shenzhen 518057, China.E-mail: anxizhou-c@my.cityu.edu.hk, hduan2-c@my.cityu.edu.hk, congwang@cityu.edu.hk

X. Yuan is with the Faculty of Information Technology, Monash University, Melbourne, VIC3800, Australia. E-mail: xingling.yuan@monash.edu.

Manuscript received April 10, 2021; revised August 3, 2021.

Recent studies [7]–[11] warn that those legitimately admitted leakage profiles, while seemingly innocent, can be exploited to devastate the security guarantee of SSE. With certain prior knowledge of a dataset, which is often a reasonable assumption in practice, an adversary can recover the content (i.e., searched keyword) of the query from the revealed access pattern. The most known leakage-abuse attack (LAA) against SSE (i.e., count attack) and its recent variants demonstrate that various leakage profiles, such as query result volumes and co-occurrence counts (derived from the access patterns of a sequence of queries), can be used to match the prior knowledge of the database [8], [9] for query recovery.

While several countermeasures have been suggested, we found they are still with certain limits in defeating LAAs. Database padding [8], [12] or specialized data structures with volume-hiding properties [9], [13], [14] only deal with a certain type of leakage profiles, e.g., the length of each query response. Ignoring other information derived from leakage like query co-occurrence patterns or individual document volume patterns, particularly in SSE schemes supporting document retrieval, may fail to defend against sophisticated adversaries. Most recently, response-hiding multi-map encryption schemes have received wide attentions [5], [15]. These constructions suggest to introduce multiplicities to hide the co-occurrence of different values. However, in the context of document search, this structure usually refers to the inverted index [5], [15], where labels are the keywords, values are the document identifiers, and index entries are label-identifier pairs [16]. As a result, after the query, the user needs to fetch the documents from the server. And if no countermeasure is enforced on the documents, such a direct application will still leak the individual document volume, which might lead to possible LAAs. In short, we believe there is still some gap in literature about how the pre-defined leakage can be linked to the broader privacy risk in SSE, which has motivated our work.

A. Our Motivation and Contributions

This work aims to advance the understanding of LAAs from a fundamental linear algebraic perspective and develop a more effective and efficient countermeasure to mitigate such attacks.

Refined Leakage and New LAA. We begin with the observation that, the impact of LAAs is far from been reported by prior arts. Specifically, leakages in LAAs are not fully utilized and they can potentially lead to a more effective attack with higher recovery rates. When modelled as index matrices, the plaintext and encrypted databases can be linked through a series of elementary linear transformations, i.e., column and

row switching. As such, for an SSE with no padding, we show the existence of invariant characteristics (including but not limited to volume or query co-occurrence) that will always be preserved under the linear transformations. These invariant characteristics can be used to unambiguously link encrypted queries (identifiers) and plaintext keywords (identifiers). We find that they, especially relationship between encrypted and plaintext identifiers, remain quite useful for LAAs even with only partial prior knowledge about the dataset.

To validate the effectiveness of our algebraic modeling, we exploit the identified characteristics to devise an LAA that achieves much higher recovery rate than all existing counterparts, based on the abuse of access pattern leakage. Specifically, we first propose to recover document identifiers to foster query recovery, which has shown particularly effective in the partial knowledge scenario. Unlike prior attacks whose attack performance is mainly dominated by known knowledge rate [9], [17], our attack will not introduce error terms and significantly narrow down candidate keywords during query recovery and improves attack performance. In other words, with the same leakage, our revised LAA recover more queries than prior arts. The above improvement makes our attack outperform prior padding countermeasures [8], [12]. Even for the recent countermeasure which suggests using encrypted multi-maps to store documents directly [9], we show that if the individual document is not pre-padded with equal size, then the countermeasure can be weakened, and thus are also subject to our proposed attack.

Local Differential Privacy and New Countermeasure. In order to overcome the limitation of SSE, our key observation is to obscure the invariant characteristics of the database so as to disrupt the distinct linkages between the plaintext database and its encrypted image of SSE. In the literature, some careful efforts have been made. Initially, random padding [7], [8] is suggested to hide the query result length, i.e., the number of documents returned by a query. After that, volume-hiding multi-maps [13], [14] are proposed to protect the total volume (size) of the query response. In above schemes, returned query results always have the same volume, or volumes perturbed from it via differential privacy techniques. Most recently, Blackstone et al. [9] propose a countermeasure which focuses on hiding the query co-occurrence pattern. This scheme still overlooks the volume of individual document. As we state in the experiment parts, our refined leakage profiles cannot be effectively protected by prior countermeasures.

As known in the literature [8], [12], [14], the original definition of SSE may not fully capture real-world adversaries. While some work [12] has been aware of this problem and proposed to establish the security notion upon the adversarial prior background knowledge of the database, as acknowledged by the authors themselves, it appears very hard to achieve constructive designs with provable security against LAA adversaries. However, doing so will pose very big practical challenges to SSE. Therefore, how to provide a practical secure notion of SSE while still provide meaningful privacy is the problem tackled in the rest of this paper.

We first generalize a new security notion, by introducing an adjustable balance parameter ϵ between privacy and efficiency. Our security intuition is to introduce random perturbations on the plaintext index matrix, so as to effectively mitigate LAAs without much extra storage cost compared to prior arts. The mitigation effect on the resulting design would be effective against all LAA attempts with past, present, and future leakages. Then our problem becomes how to achieve such an efficient perturbation scheme, such that the leakage learned by the adversary is obfuscated and hard to be utilized to recover queries. A similar problem has been well studied in the area of local differential privacy (LDP) [18], [19], which focuses on perturbing data before publishing and thereby preventing the adversary from deanonymizing records with linkage attacks. Here linkage attack function just likes LAA, and it matches "anonymized" records (access pattern) with non-anonymized records (prior knowledge) in the plaintext dataset.

The above formalization naturally guides us to design an effective and efficient countermeasure to tackle the problem fundamentally. A straightforward approach is to exploit an LDP-based binary perturbation mechanism to probabilistically perturb the search index. With the index perturbed, the actual index information is hidden and would not be identified by the adversaries. Accordingly, after feeding this index into SSE, the leakage profile derived from SSE cannot be identified as well because such privacy is immune to post-processing [20]. In DP, the post-processing theorem ensures that no matter what additional processing executed on those perturbed indexes, the finally outputs (e.g., length, co-occurrence count) will not reveal their preimage (plaintext index). However, a significant limitation of this approach is that it would introduce false negatives in search results, unless introducing heavy storage cost approximating to naive padding.

Fortunately, in the context of SSE the relationship between the identifiers of documents at the server side (encrypted documents) and the plaintext ones is generally unknown. Such relationships can be formulated as an unknown permutation. This is a default treatment of SSE and assumed commonly in most of the existing LAAs [7], [8]. Otherwise, LAAs would be a trivial work of performing index matching. Grounded on this observation, we propose a non-lossy perturbation mechanism running the noise addition operation only. The key idea is to obfuscate the index of the keywords such that the output access patterns of queries are indistinguishable with each other assisted with the above inherent permutation. Note that, this approach does not introduce extra bogus documents, because the perturbation is performed by using the identifiers of the existing ones, thereby reducing the storage cost. Furthermore, we apply padding on existing documents to protect document volumes. Compared to prior works [9], [17], [21], our design makes an attempt to use significantly reduced extra cost to mitigate the LAA threats from exploiting the SSE access pattern leakages.

Overview of Evaluation Results. We conduct a series of experiments to confirm the performance of our attacks against various datasets. Under the assumption that the full plaintext dataset is disclosed to the adversary, for randomly selected queries in Enron dataset, our attack reaches a recovery rate of 85% which is much higher than that of 49% in count attack [8]

and 59% in subgraph attack [9]. While recovering queries with low-occurrence, our attack achieves a recovery rate of 48.3% and the rates are only 23.2% and 23.5% in other two attacks. This superior performance is still retained in the adversarial scenario of only knowing the partial dataset. The recovery rate of our attack still achieves 42.1% when 50% of documents are unknown, while the recovery rates of others are only 3% and 18.2%. As seen, our attack based on refined leakage is more robust, with the document volume information, our attack can still recover more than 53% queries.

We also compare the effectiveness and efficiency of our countermeasure with prior work. From the effectiveness perspective, the query recovery rate of prior arts and our attack drops to 1%. In terms of efficiency, to achieve above security strength, we only require to introduce $3.68\times$ storage overhead which is much smaller than $272\times$ in Blackstone et al.'s OPQ scheme [9].

II. Related Work

A. Searchable Symmetric Encryption

Searchable symmetric encryption (SSE) is proposed to address the problem of confidential queries over the encrypted database. Striking a good balance between security and efficiency compared to complicated cryptographic techniques, SSE has attracted surging interests from the research community. From the basic keyword query [1], [22], more advanced range query [23]–[27] and boolean query [3], [28], to versatile domain specific query types [15], [29]–[33], many variants have been proposed over the years. There are also a list of studies focusing on dynamic constructions [34], [35], performance improvement [6], [36] and secure defense (e.g., padding [12], volume-hiding multi-maps [13], and hardware-assisted approaches [37]–[39]). This paper considers the basic yet essential case of keyword search, through which we aim to reveal the fundamental security limits of SSE.

B. LAAs Against Searchable Encryption

Inference attacks also known as passive LAAs have gradually been studied to evaluate the security strength of searchable encryption schemes under practical adversarial assumptions. There are two lines of attacks categorized by query functions, i.e., keyword search and range search.

Our work is closely related to LAAs against SSE schemes for keyword search. It is first studied by Islam et al. [7] that the access pattern, given some prior knowledge, can be utilized to infer the queried keywords and infringe privacy. Later, Cash et al. [8] show an improved attack (i.e., count attack) by using the query co-occurrence count and query response length. Blackstone et al. [9] propose a graph-based attack which views the keyword and document as vertexes and their connection as edge. It exploits document sizes and correlations within the graph to recover the query.

Most recently, Oya et al. point out that, beyond access pattern, SSE also leaks search pattern leakage which can further be leveraged to realize query recovery [40]. However, the effectiveness of this attack relies on a strong assumption of fully knowing the query frequency in the real world. Besides, the accuracy of their attack based on maximum likelihood estimation is also probabilistic, and it depends on the query distribution significantly.

Some other LAAs [41]–[49] target on schemes for range queries [25], [32], [50], [51]. They exploit the information of result volumes, order relations of the underlying data values, and have been demonstrated effectively in the applications of searching numeric data. However, it is unclear whether these attack techniques from range search can be extended to keyword search in document retrieval.

In addition, active LAAs like file-injection attacks [9], [52], [53] exploit the leakage in dynamic operations to compromise the security of SSE. This paper focuses on the static case and studies a full characterization of the leakage, explaining why the characteristics fundamentally lead to privacy breach.

C. Countermeasures Against LAAs

A few countermeasures have been proposed to thwart the attacks. The first suggestion is database padding [8], [12], which aims to eliminate the unique query result size by inserting bogus entries to the database. However, existing padding strategies only focus on obfuscating the volume leakage, a single characteristic in the leakage, leaving large room for possible attacks that exploit other characteristics. The effectiveness of the protection on other leakage such as the query co-occurrence counts is not known. Although ORAM(-like) constructions [15], [21], [54]–[59] are applicable to reduce query leakage, high computation and communication overhead will be introduced. For example, even with the state-of-the-art Path-ORAM, it incurs $\mathcal{O}(\log n)$ to $\mathcal{O}(\log^2 n)$ cost when applied to SSE.

Kamara and Moataz recently propose a new data structure called volume-hiding encrypted multi-maps [13]. Their design ensures that the actual volume associated with a single keyword cannot be extracted from the leakage by any computationally-bounded adversaries. Meanwhile, the storage overhead is reduced compared to the worse-case padding. Other variants of volume-hiding multi-map schemes adapt differential privacy to design countermeasures [14]. Patel et al. introduce the notion of differential privacy volume-hiding for Multi-Maps and propose to hide the volume by perturbing it according to a certain distribution [14]. However, as they claimed, such mitigation is sufficient only for the multi-maps whose volumes do not differ significantly.

To some extent, the above efforts alleviate the risk of volume leakage. Nevertheless, as mentioned before, when applied to SSE, these countermeasures require treating the document as the value to remain the response-hiding property. Under this setting, multiplicities for documents will lead to significant storage overhead again. While if this countermeasure only focuses on safeguarding the index information and ignores that of document level, one can still build aforementioned leakage profiles from documents (e.g., co-occurrence pattern, individual document volume pattern), thus the response-hiding property may be broken and the scheme becomes volume-hiding only. Prior LAAs against SSE have proved that only hiding volume in SSE is not sufficient [12], because the adversary can infer query contents via co-occurrence pattern. The

TABLE I
NOTATIONS

$\mathrm{M_D}$	the inverted index matrix of plaintext database
$\mathrm{M_E}$	the inverted index matrix of encrypted database
w_i	the i-th keyword in the keyword space
q_j	the j-th query in the query space
rd_i	the identifier of the i-th plaintext document
id_j	the identifier of the j-th encrypted document
$\mathcal{W}$	the keyword space
$\mathcal{Q}$	the query space
$\lvert\mathcal{S}\rvert$	the cardinal number of the set $\mathcal{S}$
$\#\mathrm{D}_i$	the size of the document D_i
$\lVert\mathrm{M}[i,:]\rVert$	the row norm of the i-th row in matrix M
$\mathrm{M}[i,j]$	the i,j-entry of matrix M
$\mathrm{M}[i,:]$	the i-th row of the matrix M
$\mathrm{M}[:,j]$	the j-th column of the matrix M
$\langle x,y\rangle$	the inner-product of vector x and y

above observation is also confirmed in our attacks and evaluation later. Our later experiment evaluations in Section VII will also validate above argument by attacking a recent SSE construction built with response-hiding Multi-maps.

Another recent work proposed by Chen et al. [60] relies on the generalized version of differential privacy, named d-differential privacy access pattern. The design obfuscates the access pattern such that queries within at most certain distance can hardly be differentiated with the prior knowledge. However, they leverage a fixed probability to obfuscate the access pattern, and thus some leakage like result length can still be inferred. In addition, to mitigate the accuracy loss, prohibitively expensive storage cost is introduced, roughly $10\times \sim 15\times$ larger to the original database. There are also some work focusing on safeguarding the statistical query in encrypted databases [61], [62], their solutions also rely on differential private techniques.

To tackle the above security and efficiency issues, we propose a notion called ϵ-indistinguishablity to build our security framework. Compared to prior works, we find that pursuing both privacy and accuracy via this notion introduces less storage overhead and can be demonstrated to be effective against powerful LAAs.

III. PRELIMINARIES

A. SSE and Notations

Let $\mathcal{W} = \{w_1, \ldots, w_m\}$ be a set of m keywords in lexicographic order. A database $\mathrm{DB} = \{\mathrm{D}_1, \ldots, \mathrm{D}_n\}$ is a collection of n documents consisting keywords in $\mathcal{W}$. Let rd_i be the identifier of the document D_i and $\mathrm{W}_i = \{w_{i_1}, \ldots, w_{i_k}\}$ represents the set of all keywords that appear in D_i. We denote the set of identifiers that appear in DB as $\mathcal{D} = \{\mathrm{rd}_i, \ldots, \mathrm{rd}_n\}$, and denote the set of documents containing the keyword w as $\mathrm{DB}(w) = \{\mathrm{rd}_{\mathrm{i}_1}, \ldots, \mathrm{rd}_{\mathrm{i}_j}\}$. We use $|\mathrm{W}_i|$ to denote the cardinal number of W_i and $\#\mathrm{D}_i$ to denote the size of D_i. With these notations, we introduce a typical SSE scheme as follows.

Searchable Symmetric Encryption. An SSE scheme [1], [6] allows keyword search over the encrypted documents with sublinear time complexity. It consists of three protocols between the client and server. `Setup` is a probabilistic protocol that takes the database DB as input and outputs an encrypted database EDB with a secret key K. `TokGen` is a protocol that generates a search token t_w for a query q with the secret key K, where the object of the query is keyword w. `Search` is a deterministic protocol which performs search over EDB via t_w, and returns a set of matched documents as response. In the `Search` protocol, the server first gets the identifier (i.e., id) of the encrypted documents which contain w and then fetches the corresponding encrypted documents for the client.

The classic security model of SSE permits certain leakage from `Setup` and `Search` algorithms, which is formalized by leakage functions. Specifically, $\mathcal{L}_{\mathtt{setup}}$ outputs the total size $(\mathrm{DB}, \mathbf{w}) = \sum_{w\in\mathcal{W}} |\mathrm{DB}(w)|$ of DB. During search, $\mathcal{L}_{\mathtt{search}}$ outputs the access pattern $\mathcal{L}_{\mathrm{Search}}(\mathrm{EDB}, \mathrm{t}_w) = \{\mathrm{id}_i : \mathrm{id}_i \in \mathrm{DB}(w)\}$. Here id_i denotes the identifier of the i-th encrypted document in EDB. From the output of $\mathcal{L}_{\mathrm{Search}}$, the size of the result set $|\mathrm{DB}(w)|$ is also exposed. For ease of presentation, we use q to denote the search token t_w.

In this paper, our analysis focuses on the widely-used index-based SSE schemes for document databases [1], because it leaks the minimally necessary leakage of SSE as recognized in [8]. We would like to clarify that, in SSE, the server only knows the encrypted identifiers of the encrypted documents (i.e., $\mathrm{id}_1, \ldots, \mathrm{id}_n$) and does not know the mappings to the identifiers (i.e., $\mathrm{rd}_1, \ldots, \mathrm{rd}_n$) of plaintext documents. In the rest of this work, we call rd_i as plaintext identifier and id_i as encrypted identifier.

B. LAA Assumptions and Definition

Following the assumption of LAA [8], we define the capability and goal of the adversary below.

Adversary: we consider a passive and persistent adversary. He can be the server or someone who can continually monitor and capture the communication records between the client and server. He can also obtain the size of the encrypted database via leakage $\mathcal{L}_{\mathrm{Setup}}$ and the client-server interaction records through $\mathcal{L}_{\mathrm{Search}}$.

Knowledge: like prior attacks [7]–[9], we assume that the adversary is able to get the full or partial knowledge of the plaintext database DB. This setting is reasonable, because the the data owner may build the encrypted database, and allow authorized users to search through it. LAA can be launched there to track/monitor users' queries.

Target: the goal of the adversary is to recover the observed queries (search tokens), i.e., finding the underlying keyword corresponding to each token, and to further recover the documents. We assume that the document identifiers have been permuted before encryption, and the server returns the permuted id to the client [8].

Based on above descriptions, an LAA, particularly query recovery attack, can be formalized as follows:

Definition 1. *Let* DB *be a collection of* n *documents consisting of the keywords in* $\mathcal{W} = \{w_1, \ldots, w_n\}$*, Let* EDB *be* DB*'s encrypted copy and* $\mathcal{Q} = \{q_1, \ldots, q_n\}$ *are observed queries. A leakage-abuse attack takes the query, query result and the background knowledge on* DB *as input and outputs a set of corresponding pairs* $\{(q_i, w_i)\}$*, where* w_j *is the content of query* q_i*.*

DB	rd_1	rd_2	rd_3	rd_4	rd_5
w_1	0	1	0	0	1
w_2	1	0	1	0	1
w_3	1	0	1	0	0
w_4	0	1	1	1	1

Plaintext Database DB

EDB	id_1	id_2	id_3	id_4	id_5
q_1	0	1	0	1	0
q_2	1	1	1	0	1
q_3	0	0	1	0	1
q_4	0	1	1	1	0

Encrypted Database EDB

Fig. 1. Database and logic relation matrix

IV. Linear Algebraic Modelling of SSE

A. Algebraic Link Between Databases

Given a database DB defined before, we can derive an *index matrix* $\mathrm{M_D} \in \{0,1\}^{m\times n}$, where the (i,j)-th entry $\mathrm{M_D}[i,j]=1$ if the document rd_j contains the keyword w_i. We define the i-th row of $\mathrm{M_D}$, $\mathrm{M_D}[i]$, as the *index vector* of w_i. Feeding the DB to an SSE scheme for setup, we can obtain an *encrypted index matrix* $\mathrm{M_E} \in \{0,1\}^{m\times n}$, the entries of which indicate the occurrence of pseudo-random document identifiers in query responses. Accordingly, we have the notion of *encrypted index vector* $\mathrm{M_E}[i]$ for q_i, which records the query occurrences in encrypted documents. All rows/columns of $\mathrm{M_D}$ and $\mathrm{M_E}$ are associated with corresponding labels, i.e., keywords/queries or identifiers. Combining with Figure 1, following we give an example of above formalization.

Example. Let

$$\mathrm{M_D}=\begin{pmatrix}0&1&0&0&1\\1&0&1&0&1\\1&0&1&0&0\\0&1&1&1&1\end{pmatrix}, \mathrm{M_E}=\begin{pmatrix}0&1&0&1&0\\1&1&1&0&1\\0&0&1&0&1\\0&1&1&1&0\end{pmatrix}$$

be the index matrices for DB and EDB, where DB is built on the keywords $(w_1, w_2,\ w_3, w_4)$ and identifiers $(\mathrm{rd}_1, \mathrm{rd}_2, \mathrm{rd}_3, \mathrm{rd}_4, \mathrm{rd}_5)$, EDB is built on the query tokens (q_1, q_2, q_3, q_4) and encrypted identifiers $(\mathrm{id}_1, \mathrm{id}_2, \mathrm{id}_3, \mathrm{id}_4, \mathrm{id}_5)$. In this example, the underlying mappings of the keyword-token and identifier-encrypted identifier are (w_1, q_3), $(w_2,\ q_4)$, (w_3, q_1), (w_4, q_2) and $(\mathrm{rd}_1, \mathrm{id}_4)$, $(\mathrm{rd}_2, \mathrm{id}_5)$,$(\mathrm{rd}_3, \mathrm{id}_2)$,$(\mathrm{rd}_4, \mathrm{id}_1)$, $(\mathrm{rd}_5, \mathrm{id}_3)$. This mapping can be represented as the transition matrices

$$\mathrm{T}_\sigma=\begin{pmatrix}0&0&1&0\\0&0&0&1\\1&0&0&0\\0&1&0&0\end{pmatrix}, \mathrm{T}_\pi=\begin{pmatrix}0&0&0&1&0\\0&0&0&0&1\\0&1&0&0&0\\1&0&0&0&0\\0&0&1&0&0\end{pmatrix}$$

It is obvious that the equation $\mathrm{M_E}=\mathrm{T}_\sigma * \mathrm{M_D} * \mathrm{T}_\pi$ holds.

Because there is a one-to-one correspondence between keywords and queries, and between documents and encrypted documents, the two index matrices can be transformed to each other by permuting their columns and rows. Formally, there exist two elementary transition matrices T_π and T_σ that perform column and row switching operations, respectively. The two index matrices can be linked by the linear transformation:

$$\mathrm{M_E}=\mathrm{T}_\sigma \mathrm{M_D} \mathrm{T}_\pi. \tag{1}$$

Note that, it is easy to verify $\mathrm{T}_\sigma \mathrm{T}_\sigma^\top = \mathbf{I}$ ($\mathrm{T}_\sigma^\top$ is the transpose of T_σ and $\mathbf{I}$ is the identity matrix), and $\mathrm{T}_\sigma^\top = \mathrm{T}_\sigma^{-1}$ (the inverse of T_σ). The same properties also hold for T_π.

According to the leakage profiles of SSE, each query will leak one row of $\mathrm{M_E}$ to the adversary, and he will eventually obtain the entire $\mathrm{M_E}$ when all distinct keywords have been queried. This leads to a natural interpretation of the linear transformation defined above. That is, once a knowledgeable adversary manages to recover the transition matrices, he can break the security guarantee of SSE outright by learning which keyword each query maps to. Let us consider an adversary knowing the plaintext index matrix $\mathrm{M_D}$ (i.e., one with the full knowledge of the plaintext database [17]). With both index matrices in hand, we show how to compute the transition matrices as follows.

We first derive two co-occurrence matrices as $\mathrm{C}_w = \mathrm{M_D}\mathrm{M_D}^\top$ and $\mathrm{C}_q = \mathrm{M_E}\mathrm{M_E}^\top$. Substituting the terms and following the properties of elementary transition matrices, we get

$$\mathrm{C}_q = (\mathrm{T}_\sigma \mathrm{M_D} \mathrm{T}_\pi)(\mathrm{T}_\sigma \mathrm{M_D} \mathrm{T}_\pi)^\top = \mathrm{T}_\sigma \mathrm{C}_w \mathrm{T}_\sigma^{-1}. \tag{2}$$

The last equation tells us that C_w and C_q are similar and hence they have identical eigenvalues. Let V be the diagonal matrix composed of the eigenvalues [63], [64] in ascending order. Since the co-occurrence matrices are symmetric and diagonalizable, we can apply eigendecomposition to them and obtain $\mathrm{C}_w = \Lambda_w \mathrm{V} \Lambda_w^{-1}$ and $\mathrm{C}_q = \Lambda_q \mathrm{V} \Lambda_q^{-1}$, where Λ_w, Λ_q are respective eigenvector matrices. Substituting the terms in the above equation, we have

$$\mathrm{C}_q = \mathrm{T}_\sigma \mathrm{C}_w \mathrm{T}_\sigma^{-1} = \mathrm{T}_\sigma \Lambda_w \mathrm{V} \Lambda_w^{-1} \mathrm{T}_\sigma^{-1} = (\mathrm{T}_\sigma \Lambda_w)\mathrm{V}(\mathrm{T}_\sigma \Lambda_w)^{-1}.$$

Comparing the equation with the eigendecomposition of C_q, we have $\mathrm{T}_\sigma \Lambda_\mathrm{w} = \mathtt{sgn} \cdot \Lambda_\mathrm{q}$ and a valid solution $\mathrm{T}_\sigma = \mathtt{sgn} \cdot \Lambda_q \Lambda_w^{-1}$. Here $\mathtt{sgn}$ is a sign matrix, where the diagonal element of $\mathtt{sgn}$ is '1' or '-1' and others are 0. That is, the eigenvectors of the corresponding query and keyword are exactly the same except their sign. An analytical solution to T_π can be similarly derived. The adversary thus is able to obtain the exact mapping between keywords (resp. real document identifiers) and queries (resp. pseudo-random document identifiers). Although above approach provides us a feasible way to recover the query, its computation cost can be up to $\mathcal{O}(n^3)$.

Remark. Despite its theoretical interest, an attack based on the above analysis is of limited practicality for two reasons. The obvious one is that the assumption of having full knowledge of the plaintext index matrix may be strong in practice. The other subtle reason lies in the uniqueness of the solution. If there is one and only one solution, then the adversary can recover all queries with certainty. This happens when all eigenvalues are distinct. Otherwise, he will not be able to determine if any of the queries matches the correct keyword. For example, when multiple rows in the plaintext index matrix are identical, there will be multiple possible values of T_σ, then the adversary cannot distinguish the true mappings of the keyword-query pair. This drives us to consider how to make use of the algebraic connection from a more practical perspective.

B. Invariant Characteristics

Our key observation is that a diversity of row-wise (and column-wise) characteristics of $\mathrm{M_D}$ and $\mathrm{M_E}$ will be preserved under the linear transformation in Equation (1). The same

holds for C_w and C_q under the transformation in Equation (2). One simple example is the number of 1's in a row in the index matrices, and it is interpreted as the query result length [17], [65]. One example is the set of values of a row in the co-occurrence matrices, and the basic forms of which are also used in previous attacks [7], [17].

Such *invariant characteristics* can be used to link the rows (and columns) between the pairing matrices and as a result, between queries and keywords. From a high level point of view, we can unambiguously link a query to a keyword if they have the same value of certain invariant characteristic and that value is unique among all keywords/queries. Intuitively, we look for characteristics that capture not only the limited information about a keyword, e.g., query volume, but also other rich information, e.g., its correlation with other keywords in the index like co-occurrence patterns. As will be shown later, we find more refined leakage computed from the index matrix and volume. Combining multiple characteristics derived from different matrices can usually produce more efficient attacks, especially in the case of partial adversarial knowledge.

V. Refined Leakage and New LAA

A. Refined Leakage Profiles

In essence, our refined leakage profiles generally describe a set of invariant characteristics of a database before and after linear transformation of a standard SSE with no padding. Recall that the volume of the plaintext and encrypted document stays the same, and the plaintext and encrypted index matrices are proven to be congruent. Therefore, the invariant characteristics during the transformation are the key to understand the information that exactly being leaked by the access pattern and volume pattern.

1) Leakage Profiles Connected to the Query: The formal notion of leakage profiles about the query is given as follows:

Query result pattern reports the search results for the submitted queries. Let $\mathrm{M_E}[i,j]=1$ denote the event that the result set of query q_i contains the encrypted document id_j. For $q_1,\ldots,q_m$, the query result pattern is defined as

$$\mathcal{K}_{\mathtt{qrp}}(\mathrm{EDB},q_1,\ldots,q_m)=\mathrm{M}_q=\mathrm{M_E}\in\{0,1\}^{m\times n}$$

Query response volume pattern reports the size of the result for the submitted query. Let $v_i=\sum_{\mathrm{M_E}[i,j]=1}\#\mathrm{D}_j$ denote the result volume of q_i. For $q_1,\ldots,q_m$, the query volume pattern is defined as

$$\mathcal{K}_{\mathtt{qvp}}(\mathrm{M_E},q_1,\cdots,q_m)=\mathcal{V}_q=[v_1,\ldots,v_m]\in\mathbb{R}^m$$

Query co-occurrence pattern reports the number of documents returned by two queries (may be the same one). Let $\mathrm{C}[i,j]=\langle\mathrm{M_E}[i,:]),\mathrm{M_E}[j,:]\rangle$ denote the number of encrypted documents returned by q_i and q_j. For $q_1,\ldots,q_m$, the query co-occurrence pattern is defined as

$$\mathcal{K}_{\mathtt{qcp}}(\mathrm{EDB},q_1,\cdots,q_m)=\mathrm{C}_q=\mathrm{M_E}\cdot\mathrm{M_E^\top}\in\mathbb{Z}^{m\times m}$$

Query norm pattern reports the norm of the rows in the query co-occurrence matrix. Let $\mathrm{N}_q^i=\sqrt{\langle\mathrm{C}_q[i,:],\mathrm{C}_q[i,:]\rangle}$ denote the i-th row norm of the co-occurrence matrix C. For queries $\mathrm{q}_1,\ldots,q_m$, the query norm pattern is defined as

$$\mathcal{K}_{\mathtt{qnp}}(\mathrm{C}_q,q_1,\cdots,q_m)=\mathrm{N}_q=\{\mathrm{N}_q^1,\ldots,\mathrm{N}_q^m\}\in\mathbb{R}^m$$

2) Leakage Profiles Connected to the Document: Likewise, similar leakage profiles can be defined from the document perspective.

Document occurrence pattern reports the keyword associated with a document. Let $\mathrm{M}_e[i,j]=1$ indicate that the document id_i is returned by q_j. For documents with identifiers $\mathrm{id}_1,\ldots,\mathrm{id}_n$, the document occurrence pattern is defined as

$$\mathcal{K}_{\mathtt{dop}}(\mathrm{EDB},\mathrm{id}_1,\ldots,\mathrm{id}_n)=\mathrm{M}_e=\mathrm{M_E^\top}\in\{0,1\}^{n\times m}$$

Document volume pattern reports the size of a document. For the observed documents with identifiers $\mathrm{id}_1,\ldots,\mathrm{id}_n$, the document volume pattern is defined as

$$\mathcal{K}_{\mathtt{dvp}}(\mathrm{EDB},\mathrm{id}_1,\ldots,\mathrm{id}_n)=\mathcal{V}_e=\{\#\mathrm{D}_1,\ldots,\#\mathrm{D}_n\}\in\mathbb{R}^n$$

Document co-occurrence pattern reports the number of queries associated with two documents. Let $\mathrm{C}_e[i,j]$ be the number of documents appearing in both id_i and id_j. For id_1,…,id_n, the document co-occurrence pattern is defined as

$$\mathcal{K}_{\mathtt{dcp}}(\mathrm{EDB},\mathrm{id}_1,\cdots,\mathrm{id}_n)=\mathrm{C}_e=\mathrm{M_E^\top}\times\mathrm{M_E}\in\mathbb{Z}^{n\times n}$$

Document co-occurrence norm pattern reports the norm of rows in the document co-occurrence matrix. Let $\mathrm{N}_e^i=\|\mathrm{C}_e[i,:]\|$ denote the i-th row norm of C_e. For $\mathrm{id}_1,\ldots,\mathrm{id}_n$, the document co-occurrence norm pattern is defined as

$$\mathcal{K}_{\mathtt{dnp}}(\mathrm{C},\mathrm{id}_1,\cdots,\mathrm{id}_n)=\mathrm{N}_e=\{\mathrm{N}_e^1,\ldots,\mathrm{N}_e^n\}\in\mathbb{R}^n$$

Remark. Our leakage profiles capture more precise information extracted from the query compared to the existing coarse ones. They identify the distinctiveness of each query, which is the key to an efficient LAA. Because the query and keyword co-occurrence matrices are congruent, the leakages extracted from them are exactly the same except the position changes.

B. LAA with Refined Leakage

According to the leakage profiles defined above, following we present an LAA against SSE. From a high level point of view, the adversary aims to link the derived leakage to each query and keyword, respectively. Here we consider a practical scenario where only a subset of documents is disclosed to the adversary. Under this setting, the effectiveness of prior attacks [7], [8] always downgrades because of a gap between the prior knowledge and the observed one, specifically in query result pattern and query co-occurrence pattern. He has to randomly guess among the possible candidates. In our LAA, we find that partial documents knowledge is of limited impacts on the co-relationship between known documents. Therefore, we can utilize them to select out the encrypted identifiers corresponded to the disclosed document index first, and then use the leakage derived from these mapped plaintext and encrypted documents to recover the query.

Like prior work [9], we assume that the adversary $\mathcal{A}$ waits until search results for all queries in the database are revealed. This is reasonable as $\mathcal{A}$ can continuously monitor the transcripts in the protocol of SSE. Under this setting, $\mathcal{A}$ knows partial index matrix $\mathrm{M_D}\in\{0,1\}^{m\times s}$ (missing some columns in the index matrix) and encrypted index matrix $\mathrm{M_E}^{m\times n}$. If $s=n$, the attack is performed under the setting of full prior

Let $\mathrm{M_D} \in \{0,1\}^{m\times s}$ be the index matrix of known documents $\{\mathrm{rd}\}_{i=1}^{s}$ on keyword space $\mathcal{W} = \{w_i\}_{i=1}^{m}$ and $\mathrm{M_E} \in \{0,1\}^{m\times n}$ be the index derived from the query $\mathcal{Q} = \{q_i\}_{i=1}^{m}$ and encrypted documents $\{\mathrm{id}\}_{i=1}^{n}$. Let $\mathcal{V}_d = \{v_d^1, \ldots, v_d^s\}$ and $\mathcal{V}_e = \{v_e^1, \ldots, v_e^n\}$ be the volume of the plaintext and encrypted documents. The goal is to output the mapping from keyword w to query q.

`Identifier_Recovery` :

Initialize a map $\mathcal{I}$ for $(\mathrm{id}_i, \mathrm{rd}_j)$ having unique volume
Compute $\mathrm{C}_d \leftarrow \mathcal{K}_{\mathrm{dcp}}(\mathrm{M_D^\top}, \mathrm{rd}_1, \ldots, \mathrm{rd}_n)$
Compute $\mathrm{C}_e \leftarrow \mathcal{K}_{\mathrm{dcp}}(\mathrm{M_E^\top}, \mathrm{id}_1, \ldots, \mathrm{id}_n)$
while size of $\mathcal{I}$ is increasing **do**
  for each unknown identifier $\mathrm{id}_i \in \{\mathrm{id}\}_{i=1}^{n}/\mathcal{I}$ **do**
    Set candidate set $\mathcal{S}_i = \{\mathrm{rd}_j : \mathrm{C}_e[i,i] = \mathrm{C}_d[j,j]\}$
    for $\mathrm{rd}_j \in \mathcal{S}_i$ **do**
      for known identifiers $(\mathrm{id}_{i*}, \mathrm{rd}_{j*}) \in \mathcal{I}$ **do**
        if $\mathrm{C}_e[i,i^*] \neq \mathrm{C}_d[j,j^*]$ or $\{v_e^i \neq v_d^j\}$ **then**
          Remove rd_j from $\mathcal{S}_i$
        end if
      end for
    end for
    Add $(\mathrm{id}_i, \mathrm{rd}_j)$ to $\mathcal{I}$ if $|\mathcal{S}_i| = 1$
  end for
end while
for each unknown identifier $\mathrm{id}_i \notin \mathcal{I}$ **do**
  Set $\mathcal{T}_i = \{\mathrm{id}_j : \mathcal{S}_i = \mathcal{S}_j\}$
  Add $(\mathcal{T}_i, \mathcal{S}_i)$ to $\mathcal{J}$ if $|\mathcal{T}_i| = |\mathcal{S}_i|$
end for
return recovered identifier pairs $\mathcal{I}$ and possible pairs $\mathcal{J}$.

`Query_Recovery` :

Run $(\mathcal{I}, \mathcal{J}) \leftarrow$ `Identity_Recovery`$(\mathrm{M_D}, \mathrm{M_E})$
Build index matrices $\mathrm{M_E^+}$ and $\mathrm{M_D^+}$ using $\mathcal{I}$ and $\mathcal{J}$
Compute $\mathrm{C}_w^+ \leftarrow \mathcal{K}_{\mathrm{qcp}}(\mathrm{M_D^+}, w_1, \ldots, w_n)$
Compute $\mathrm{C}_q^+ \leftarrow \mathcal{K}_{\mathrm{qcp}}(\mathrm{M_E^+}, q_1, \ldots, q_n)$
Compute $\mathrm{N}_w^+ \leftarrow \mathcal{K}_{\mathrm{qnp}}(\mathrm{C}_w^+, w_1, \ldots, w_m)$
Compute $\mathrm{N}_q^+ \leftarrow \mathcal{K}_{\mathrm{qnp}}(\mathrm{C}_q^+, q_1, \ldots, q_m)$
Initialize a map $\mathcal{U}$ for (q_i, w_j) having unique query norm
while size of $\mathcal{U}$ is increasing **do**
  for each query $q_i \notin \mathcal{U}$ and $(\mathrm{id}_s, \mathrm{rd}_t) \in \mathcal{I}$ **do**
    $\mathcal{S}_i = \{w_j : \mathrm{N}_q[i] = \mathrm{N}_w[j] \wedge \mathrm{M_E^+}[i,s] = \mathrm{M_D^+}[j,t]\}$
    for $w_j \in \mathcal{S}_i$ **do**
      for $(q_{i^*}, w_{j^*}) \in \mathcal{U}$ **do**
        if $\mathrm{C}_q^+(i,i^*) \neq \mathrm{C}_w^+[j,j^*]$ **then**
          Remove w_j from $\mathcal{S}_i$
        end if
      end for
    end for
    Add (q_i, w_j) to $\mathcal{U}$ if $|\mathcal{S}_i| = 1$
  end for
end while
return $\mathcal{U}$

Fig. 2. Leakage Abuse Attack with Refined Leakage

knowledge. Here we assume that $\mathrm{M_D}$ and $\mathrm{M_E}$ have the same row dimension, because the keyword space and the query space are known. If a certain keyword does not appear in the known documents, we fill such a row with '0'. Also, the volume $\mathcal{V}_d = \{v_d^1, \ldots, v_d^s\}$ and $\mathcal{V}_e = \{v_e^1, \ldots, v_e^n\}$ of plaintext and encrypted documents are known to $\mathcal{A}$.

Attack Construction. As described in Fig. 2, our attack consists of two algorithms, `Identifier_Recovery` and `Query_Recovery`, where `Identifier_Recovery` attempts to recover the co-relation between the plaintext and encrypted documents/identifiers and `Query_Recovery` attempts to recover the query. The details are given as follows.

In `Identifier_Recovery`, for an identifier id_i, $\mathcal{A}$ first initializes a known identifier map $\mathcal{I}$ with plaintext and encrypted documents $(\mathrm{id}_i, \mathrm{rd}_j)$ having a unique volume, and builds the candidate set as $\mathcal{S}_i = \{\mathrm{rd}_j : v_e^i = v_d^j\}$ according to their volume information. After that, $\mathcal{A}$ computes the document co-occurrence matrices $(\mathrm{C}_d^*, \mathrm{C}_e^*)$ with $\mathrm{M_D^\top}$ and $\mathrm{M_E^\top}$ and uses them as the reference to tick out the unmatched items in the candidate set for other identifiers. Specifically, for a known mapping id_{i^*} and rd_{j^*}, $\mathcal{A}$ removes j from $\mathcal{S}_i$ if $\mathrm{C}_e^*[i,i^*] \neq \mathrm{C}_d^*[j,j^*]$. When there is only one element remaining in $\mathcal{S}_i$, id_i and rd_j are matched identifiers. $\mathcal{A}$ adds $(\mathrm{id}_i, \mathrm{rd}_j)$ into $\mathcal{I}$. When all the unique mappings are determined, for $\mathrm{id}_i \notin \mathcal{I}$, $\mathcal{A}$ sets $\mathcal{T}_i = \{\mathrm{id}_j : \mathcal{S}_i = \mathcal{S}_j\}$ and checks if $|\mathcal{T}_i| = |\mathcal{S}_i|$. If yes, he outputs such documents into a map $\mathcal{J}$. Doing this can find an additional plaintext document set and an encrypted document set that exactly match, even though the correlations of individual elements in the two sets are unknown. Note that, if $s = n$, we can directly obtain $\mathcal{J}$, which is the set of rest identifiers in the databases.

In `Query_Recovery`, we utilize the above recovered identifier maps $\mathcal{I}$ and $\mathcal{J}$ to recover encrypted queries. Let $\mathrm{M_E^+}$ and $\mathrm{M_D^+}$ be the index matrices built on the documents in $\mathcal{I}$ and $\mathcal{J}$. It is clear that the index matrix built over the document set consisting of $\mathcal{I}$ and $\mathcal{J}$ have the same dimension. Similar to `Identifier_Recovery`, $\mathcal{A}$ first initializes a map $\mathcal{U}$ with (q_i, w_j) having the unique query norm and determines the candidate set for uncovered queries according to the leakages (i.e., query norm pattern, volume pattern and co-occurrence pattern) extracted from the index matrices $\mathrm{M_E^+}$, $\mathrm{M_D^+}$. Afterwards, $\mathcal{A}$ gradually removes unmatched candidates from the candidate set by checking the leakage. Once there is only one candidate remained, it indicates that it is the content of the query. $\mathcal{A}$ repeats this operation until the recovery set $\mathcal{U}$ stops increasing.

Notably, in our attack construction, we leverage the individual document volume to link encrypted and plaintext identifiers and then uses their linkages to help recover the query. While for the volumetric attack in [9], it directly recovers the query by matching the query and keyword with the same unique response volume (i.e., the total size of returned documents). To highlight the importance of the recover encrypted identifiers, following we illustrate how to use identifiers to narrow down the candidates of queries, thus advancing attack performances.

Example: Our first example is to explain how to use identifier

to further narrow down the candidate set. For example: let q_1, q_2 be two queries which have the same candidate set $\{w_1, w_2\}$, let id_1 be the result of query q_1 and id_2 be the query of q_2, assume that w_1 only occurs in the document rd_1 and w_2 only occurs in the document rd_2. Now, if the adversary knows id_1 and id_2 are the encrypted copies of $\mathrm{rd}_1 = \{w_1, w_1, w_1\}$ and $w_2 = \{w_2, w_2\}$ (because the document volumes are different), respectively, then he can determine that the underlying keywords of q_1, q_2 are w_1, w_2, respectively.

The second example provides a straightforward way to use identifiers can recover the query. Assume that the adversary knows $\mathrm{rd}_1, \mathrm{rd}_2, \mathrm{rd}_3$ are corresponding plaintext identifiers for encrypted documents $\mathrm{id}_1, \mathrm{id}_2, \mathrm{id}_3$. By observing the query results, if he finds that $\{\mathrm{rd}_1\}$, $\{\mathrm{rd}_2\}$, $\{\mathrm{rd}_3\}$, $\{\mathrm{rd}_1, \mathrm{rd}_2\}$, $\{\mathrm{rd}_1, \mathrm{rd}_3\}$, $\ldots$, $\{\mathrm{rd}_1, \mathrm{rd}_2, \mathrm{rd}_3\}$ are all unique in the result space, then he can recover these 8 queries. Ideally, one can use $\log_2 n$ documents to recover n queries if all query have different query results. Similar results can also be found in the work of file-injection attack [52].

C. Advantages Compared to Prior LAAs

As seen in Figure 2, our attack leverages several refined leakages from access pattern which have not been mentioned in prior arts, e.g., document co-occurrence pattern, document volume pattern, etc. Following we specify the advantage of our attack when compared to prior work.

Our proposed LAA is more robust compared to prior arts. First, in the setting of partially known knowledge, prior works [9] redefine the candidate set by introducing an error term derived from the missing index information. For example, suppose $\mathcal{A}$ knows 70% of the original dataset and observes a query with response volume 10, then he will select the keyword with response volume around 7 (i.e., $[5, 9]$) as the candidates, which clearly leads to accuracy loss in query recovery. Different from these works, we exploit the new leakage profiles to find a set of encrypted documents that exactly match a target set of plaintext documents and then launch the attack over such documents to recover queries. During the whole procedure, our attack performs equality matching between leakage and prior knowledge without information loss.

Second, our LAA is still effective against those countermeasures that direclty apply volume-padding [17] or response-hiding encrypted multi-map [9] to the context of document search, without proper defence on individual documents. Here we take a typical padding countermeasure in [17] as an example, which adds randomly generated bogus documents into the database to obfuscate the query occurrence pattern and query result length. However, as mentioned before, our attack can match the plaintext and encrypted documents, and thus can exclude those bogus documents. Another important reason contributing to the effectiveness of our attack is that it exploits the volume leakage of the individual document, which is not protected in most countermeasures. More details are evidenced in our evaluations later.

VI. Our Proposed Countermeasure

A. Design Intuition

Our analysis and attack show that for a standard SSE scheme that returns exact results or without proper countermeasures on both index and document levels, an adversary with prior knowledge of the database may always break its confidentiality by LAAs. A general and effective countermeasure needs to achieve: 1) it is imposed on the database before encryption; and 2) it must obfuscate the unique invariant characteristics intrinsically preserved by linear transformations to break the links between the prior knowledge and the leakage.

In light of the above observation, we find that the problem of defending LAAs can be essentially reduced to protect keyword against query result publishing. The similar problem has been well studied in the area of LDP [20], which ensures a server is unable to learn much about individual data when answering queries over a dataset aggregated from multiple users. This is achieved by requiring each user to perturb her data before submission to server. In our case, each row of the index matrix is analogous to a user's data. To this end, we give a new security notion by introducing an adjustable balance parameter ϵ between privacy and efficiency.

With the notion, the next task is how to implement it. By properly perturbing the index matrix, we may achieve a level of privacy guarantee similar to LDP. However, note that, directly applying an LDP-perturbation algorithm may lead to accuracy loss. That is, some elements in the index matrix may be flipped from '1' to '0' due to perturbation. Fortunately, the link between the plaintext and encrypted identifiers is usually hidden after encryption, which helps us achieve the desired security without lossy queries. Specifically, our countermeasure only utilizes existing documents and identifiers to perform perturbations, without introducing extra bogus documents to the database. This is different from many prior arts that suggest injection of bogus documents with extra storage overhead [8], [12]. Meanwhile, to hide the document volume, we still perform padding on documents for a comprehensive countermeasure.

B. Our Construction

Now we give the formal security notion and show how to design a countermeasure that can achieve the above goals. As mentioned, the proposed countermeasure is a separate algorithm to be applied on the database before the encryption of SSE. We term this algorithm as `Encode`. As the rest of functions are standard in SSE and will not be modified.

To capture this adversary's capability, we introduce a complementary security notion for SSE, called ϵ-indistinguishable leakage, which is proposed for ensuring pattern-hiding. Here we use the term pattern-hiding in order to differentiate it with the prior volume-hiding and multi-map based response-hiding notions. In formally, pattern-hiding focuses on protect leakages (e.g., index and document level leakage) that can be exploited to launch LAAs on SSE schemes that supports document search. Afterwards, we will conduct theoretical analysis to

prove that our countermeasures is ϵ-indistinguishable secure against a passive adversary with database prior knowledge.

Following the same way in Section V, we use a general term $\mathcal{K}_f$ to denote the leakage associated with leakage profile f (includes but no limited our definition in Section V-A), and the definition ϵ-indistinguishability is formalized as follows:

Definition 2 (ϵ-Indistinguishable Secure)**.** *Let* Π *be an SSE scheme over database* DB *and keyword space* $\mathcal{W}$*, let* EDB *be the encrypted database and* $\mathcal{Q}$ *be the query space, we say that an SSE scheme is* ϵ*-indistinguishable secure if for any* $w_i, w_j \in \mathcal{W}, q \in \mathcal{Q}$*,*

$$e^{-\epsilon} \leq \frac{\Pr[\mathcal{K}_f(\mathrm{DB}, w_i) \simeq \mathcal{K}_f(\mathrm{EDB}, q)]}{\Pr[\mathcal{K}_f(\mathrm{DB}, w_j) \simeq \mathcal{K}_f(\mathrm{EDB}, q)]} \leq e^{\epsilon}$$

holds, where $\mathcal{K}_f(\mathrm{DB}, w) \simeq \mathcal{K}_f(\mathrm{EBD}, q)$ *means the derived characteristics of* w *and* q *from function* $\mathcal{K}_f$ *are equal.*

Definition 2 guarantees that for any two observed queries and the related responses, the leakage extracted from the observation is sufficiently indistinguishable. In other words, inferring the content of the original DB from the observed leakage profiles becomes difficult. The above definition borrows the principle of LDP, which is adapted to the context of SSE. We term it ϵ-indistinguishable secure since it provides the constraint atop of SSE that the leakage are indistinguishable and can no longer be available in query inference.

1) Customized Perturbation on Search Index: To realize a countermeasure with non-lossy, the operation flipping "1" to "0' should be fully denied in the binary perturbation over index matrix $\mathrm{M_D}$. Note that the published data is the encrypted index, and the specific relationship between the plaintext identifier and its encrypted copy is unknown without further exploration. In other words, besides perturbation, a permutation (although the same) has been run over each index vector, and it is the key to design a non-lossy countermeasure. Additionally, since the result of query recovery task against textual data is either black or white, a specific index vector can only be recovered if the adversary exactly knows the i-th bit in the index vector comes from perturbation or permutation. Our empirical result shows that recovering such permutation is not trivial when noises are inserted to obfuscate the plaintext index.

From the above observation, we propose a perturbation by only adding "1" in the index matrix $\mathrm{M_D}$. Note that the adversary cannot perform equality matching between the access pattern and prior knowledge, because of the inherent identifier permutation in SSE. Hence, the adversary should mine the knowledge for attack from the co-occurrence matrix, as analyzed in our attacks. It tells us that our perturbations algorithm should obfuscate this co-occurrence information as much as possible. Accordingly, we give a construction of the `Encode` algorithm. Before that, we review Definition 3 which will be used later.

Definition 3. *Let* x_1, x_2 *be two vectors in the set* $\mathcal{X} \subset \{0,1\}^n$*, the distance of* x_1 *and* x_2 *is defined as* $\mathrm{dist}(x_1, x_2) = |x_1 \oplus x_2|$*, where* $\oplus$ *denotes the XOR operation on binary vectors. Accordingly, the distance between the element and the set is defined as* $\mathrm{dist}(x, \mathcal{X}) = \max_{x' \in \mathcal{X}} \mathrm{dist}|x \oplus x'|$*. A set* $\mathcal{X}$ *is said to be* d*-controllable if for any* $x_1, x_2 \in \mathcal{X}$*,* $\mathrm{dist}(x_1, x_2) \leq d$ *holds. We denote such a set as* $\mathcal{X}_d$*.*

Input: $x \in \mathcal{X}, \mathcal{X} \subset \{0,1\}^n$
Output: $y \in \{0,1\}^n$
1: $d \leftarrow \max\{\mathrm{dist}(x, \mathcal{X})\}$ // distance parameter
2: $\ell \leftarrow \max_{x \in \mathcal{X}} |x|$ // $|\cdot|$ denotes the hamming weight
3: $\mathrm{len} \leftarrow |x|$ // hamming weight of vector x
4: $(x^{(1)}, \ldots, x^{(n)}) \leftarrow x$
5: $\rho \leftarrow (\ell - \mathrm{len})/(n - \mathrm{len})$ // perturbation probability
6: **for** $1 \leq i \leq n$ **do**
7: set $y^{(i)} \leftarrow 1$ with probability ρ if $x^{(i)} \neq 1$
8: **end for**
9: **if** $\mathrm{dist}(y, \mathcal{X}) \leq d$ or $|y| \geq \ell$ **then**
10: $y \leftarrow (y^{(1)}, \ldots, y^{(n)})$
11: **else**
12: repeat line 4 to line 9
13: **end if**
14: **Return** y

Fig. 3. Our Binary Perturbation Algorithm

Next, we describe our perturbation algorithm for database encoding. As depicted in Fig. 3, it takes an n-bit vector x and a set $\mathcal{X}$ as inputs and outputs an obfuscated vector y. Specifically, $d = \max\{\mathrm{dist}(x, \mathcal{X})\}$ is firstly set as the vector distance controllable parameter, and $\ell = \max_{x \in \mathcal{X}} |x|$ is set as the expected hamming weight of the obfuscated vector. Then for the i-th bit in x, $y^{(i)} = 1$ is executed with probability ρ if $x^{(i)} \neq 1$ and a vector $y = (y^{(1)}, \ldots, y^{(n)})$ is obtained, where $\rho = (\ell - |x|)/(n - |x|)$ and $|x|$ denotes the hamming weight of x. Here we have $x < n$, otherwise, there could be a document containing all keywords in the keyword space. After that, whether $\mathrm{dist}(y, \mathcal{X}) \leq d$ or $|y| \geq \ell$ holds is checked. If yes, y is returned as the obfuscated vector for x. Otherwise, the above operations are repeated until a qualified y is obtained.

As seen, when applying the perturbation algorithm to SSE, each returned vector y is required to meet the condition $\mathrm{dist}(y, \mathcal{X}) \leq d$. By adopting this constraint, therein distance between the outputs will be closer, which helps to perturb the similarity of index vectors (aka co-occurrence information between different queries). Therefore, the risk of learning the query content from analyzing the co-occurrence information can be degraded. After that, we rebuild the encoded database DB' according to the obfuscated index vectors and encrypt it. Note that, during this procedure (only perturbing the index, not adding keywords to the documents), the individual document volume will not be changed.

2) Document Volume Padding: SSE also leaks the volume information of individual documents. To protect such information, a straightforward approach is to pad them into the same volume. However, the cost of such a padding strategy is prohibitive. Inspired by Bost et al.'s cluster-based padding approach [12], following we give a similar document padding approach to balance efficiency and security. Specifically, the client first groups the documents into several clusters such that each cluster contains at least a certain number of documents. According to the number of each cluster, the client inserts keywords in the documents and makes the documents in each cluster have the same volume. The padding strength

relies on a selected padding parameter, which determines the number of clusters in the group. For example, for a given document set $\mathrm{DB} = \{D_1, \cdots, D_n\}$, the client first divides all documents into t clusters $\mathcal{C}_1, \cdots, \mathcal{C}_t$. Then for each cluster, it first computes $v_i = \max_{D_j \in \mathcal{C}_i} \#D_j$ as the final padding size. Then for each document $D_j \in \mathcal{C}_i$, it randomly selects $v_i - \#D_j$ keywords in the set D_j and appends them after the file. After that, all documents in the same cluster keep the same volume size, and the adversary cannot exploit the volume information to identify each document. Note that, selecting keywords from the document itself is to maintain the original index information.

Advantages Compared to Prior Countermeasures. Tiered random padding [8] was proposed by Cash et al. as an intuitive defense against LAAs. Specifically, this method inserts a set of bogus documents into the database such that the number of documents returned by each query is padded to the nearest multiple of an integer n. However, hiding query result length by random padding may not be effective because such a method cannot effectively hide the co-occurrence leakage between documents. As shown later, our attack can identify the relationship between the encrypted and plaintext documents and remove the bogus documents from the database.

To further hide the co-occurrence leakage of the documents, Blackstone et al. [9] introduced a response hiding multi-map encryption scheme to directly store the documents rather than document identifiers. They requires that the same document associated with different keywords is encrypted into different copies. In particular, the number of copies is equal to the number of keywords in the document. As a result, the co-occurrence count of every two queries will always be "0", and the adversary cannot directly exploit the co-occurrence pattern to recover the query. Likewise, this countermeasure does not work in our attack, because the volume of some documents may be unique. Based on this leakage, some of the encrypted documents can be identified and be further used in the attack.

Compare to the above work, our defense perturbs over the searchable index in the existing document space, which essentially obfuscates the query access pattern. Moreover, our defense does not introduce much storage overhead to the database, as no bogus document is added. Meanwhile, the potential privacy risk from document volume is also taken into account, thus document padding strategy is also provided.

C. Security and Overhead Analysis

The following theorem is given to show that our proposed countermeasures can achieve above ϵ-indistinguishable security in an average case, where the permutation among index vector is assumed to be independent. For ease of presentation, we term our countermeasure as pattern-hiding SSE (PH-SSE) in this rest of this paper.

Theorem 1. *Let* Π *be our PH-SSE scheme over database* DB *and keyword space* $\mathcal{W}$*, and* d *be the maximum hamming distance of index vectors in database* DB*. For each* $w_i \in \mathcal{W}$*, assume that the hamming weight of its index vector locates in the range* $[\hbar, \ell]$*, where* $\hbar, \ell \ll n$*, and* n *is the number of documents. Then* Π *is an* ϵ*-indistinguishable pattern-hiding scheme, where* $\epsilon = d \ln(\ell/\hbar)$.

Proof. Let x denote the index vector of plaintext database DB and y denote the index vector of encoded database DB$'$. Let z be the encrypted index vector derived from query result over encrypted database. In this work we have that perturbation algorithm takes x as input and returns encode index vector y as a output. And SSE takes y as input and outputs the encrypted index vector z. Therefore, we have

$$\Pr[z|x] = \Sigma_{y \in \{0,1\}^n} \Pr[z|y] \Pr[y|x] \tag{3}$$

where $\Pr[y|x]$ denotes the probability of outputting y conditioned on x and $\Pr[z|y]$ denotes the probability of outputting z conditioned on y. Let $x^{(i)}, y^{(i)}, z^{(i)}$ be the i-th bit of x, y, z, respectively. According to Eq (3), we can get

$$\Pr[z^{(i)}|x^{(i)}] = \Sigma_{y^{(i)} \in \{0,1\}} \Pr[z^{(i)}|y^{(i)}] \Pr[y^{(i)}|x^{(i)}] \tag{4}$$

Now, we define $\Pr[y^{(i)}|x]$ as the probability of the i-th output of y is $y^{(i)}$ conditioned on input x. Since $y^{(i)}$ only relies on the i-th bit of input x and is independent with other bits. Thus we have $\Pr[y^{(i)}|x] = \Pr[y^{(i)}|x^{(i)}]$ and thereby $\Pr[z^{(i)}|x] = \Pr[z^{(i)}|x^{(i)}]$. Clearly,

$$\Pr[\mathrm{Z} = z|\mathrm{X} = x] = \prod_{i=1}^{n} \Pr[z^{(i)}|x] = \prod_{i=1}^{n} \Pr[z^{(i)}|x^{(i)}]. \tag{5}$$

Because encryption of identifiers in SSE can be regarded as a permutation, we assume that the unchanged permutation probability for bit in index vector is $1/n$ without loss generality. Combining with the algorithm in Fig. 3, we have that

$$\Pr(z^{(i)}|x^{(i)}) = \begin{cases} \frac{|x|}{n}, & \text{if } z^{(i)} = 1 \text{ and } x^{(i)} = 1 \\ \frac{n-|x|}{n}, & \text{if } z^{(i)} = 0 \text{ and } x^{(i)} = 1 \\ \frac{|x|}{n}, & \text{if } z^{(i)} = 1 \text{ and } x^{(i)} = 0 \\ \frac{n-|x|}{n}, & \text{if } z^{(i)} = 0 \text{ and } x^{(i)} = 0 \end{cases} \tag{6}$$

Applying above result to Eq (5), it is clear that

$$\begin{aligned} \frac{\Pr[z|x_1]}{\Pr[z|x_2]} &= \frac{\Pr[z^{(1)}|x_1^{(1)}] \cdots \Pr[z^{(n)}|x_n^{(i)}]}{\Pr[z^{(1)}|x_2^{(1)}] \cdots \Pr[z^{(n)}|x_2^{(n)}]} \\ &\leq \max\left\{ \frac{\ell}{\hbar}, \frac{(n-\hbar)}{(n-\ell)}, \frac{\hbar}{\ell}, \frac{(n-\ell)}{(n-\hbar)} \right\}^d \leq (\ell/\hbar)^d \end{aligned}$$

As seen, we obtain $\epsilon = d \ln(\ell/\hbar)$. Based on Proposition 2.2 of post processing in [20], when we view leakage as the output of leakage profiles derived from query result of Fig. 3, this leakage also satisfies ϵ-indistinguishability. Thus, our PH-SSE scheme is a $d \ln(\ell/\hbar)$-indistinguishable PH-SSE scheme. This completes the proof. □

At a high level, the access pattern observed by an adversary is produced through a two-step process including a perturbation on the original keyword-document index and an SSE encryption scheme. The security of the PH-SSE scheme relies on both the perturbation algorithm presented above and the adopted SSE scheme, where the perturbation modifies the occurrence of the keyword in the document and the SSE permutes the keywords and identifiers (aka, row and column in index matrix). Under the assumption that the identifier

permutations here are nearly randomly, then the combination of these two approaches makes it possible and easy to achieve that the returned identifier list of a certain query may be from any truly one of the keywords in the databases.

VII. Experiment Evaluation

Our goals are to evaluate the performance and advantages of the proposed attack and countermeasure. Through the experiments, we will demonstrate that: (1) our LAA is more powerful than prior attacks and can even work well against existing countermeasures; (2) the proposed countermeasure provides comprehensive and effective protection of SSE against LAAs and incurs less storage cost than prior countermeasures.

A. Experiment Setup

To demonstrate the effectiveness of our attack based on refined leakage, we first report the statistics of the unique leakage values computed from the defined leakage profiles. After that, we compare our attack with two prior arts-count attack [8] (the most known one) and SubGraph [9] (the most recent one) in terms of attack efficiency and effectiveness. To demonstrate the efficiency and effectiveness of our countermeasure, we compare the security strength, i.e., query recovery ratio against different LAAs, and storage overhead (including index and documents) between the existing padding countermeasure and our countermeasure. We perform evaluations of our proposed attack and countermeasure over two real-world datasets, i.e., Enron email dataset [66] and IMDB movie comments dataset [67]. The Enron dataset consists of emails on different topics and IMDB dataset displays the movie comments. We clarify this to show that this inherent nature determines that query in IMDB is more likely to have similar co-occurrence counts. We implement the experiments via Python and conduct them on a laptop with an Intel Core i7, 3.2GHz processor and 16GB memory running MacOS.

Statistics of Leakages. In our attacks, an adversary with prior knowledge of a database can obtain the mapping of search tokens and keywords from the unique leakage characteristics. To demonstrate the exploitability of the refined leakage, we measure the sensitivity (frequency) of various leakages in two datasets, which include query result length, document volume, co-occurrence count, row norm, and eigenvector. High sensitivity means the leakage profile can generate more unique characteristics. For both datasets, we randomly choose 20,000 keywords from 50,000 documents.

The two plots on the left in Fig. 4 show the ratio of the unique leakage values at two databases. As we can see, 33% documents in the Enron dataset and 29% documents in the IMDB dataset have unique volumes, which validates the existence of unique document volumes in real-world datasets. Another observation is that document volume, row norm, and eigenvectors have more unique values compared to the result length and co-occurrence count. In the Enron email database, the unique value with the highest ratio is eigenvector which is nearly close to 100%. The ratio of the unique query norm is 99.9%. The ratio of the unique result length and co-occurrence count is relatively low, i.e., 22% and 29%, respectively. The two plots on the right in Fig. 4 show the maximum frequency of different types of leakage values. If there is no unique value in certain leakage, the adversary can only randomly guess the keyword from a candidate set. These two plots tell us document volume, query norm, and eigenvectors lead to lower probability for the adversary to correctly guess the keywords than result length and co-occurrence count.

B. Evaluation on LAAs

We evaluate our attack on different types of datasets processed from two databases to demonstrate it effectiveness (w.r.t. query recovery rate) and robustness (w.r.t. datasets with various features). Moreover, we compare our attack with prior arts to confirm its superior performance.

Attack Setup. We first summarize the methodologies and settings of the attacks we evaluated in this work. Meanwhile, we specify the difference of our attack and other LAAs implementation.

(1) Count attack [8], [17] exploits the unique response length and query co-occurrence pattern to map the query to the keyword. Specifically, it first maps the keyword and query that have the same yet unique response length, and then utilizes these known keyword/query pairs to determine the unknown pairs by checking their co-occurrence counts;

(2) SubGraph attack [9] views the database as a graph. The vertexes represent documents or keywords, and the edge can be created if the keyword appears in that document. For the given graphs of plaintext and encrypted databases, SubGraph attack leverages the observed knowledge (degree, namely response length) to match the query and corresponding keyword. Beyond count attack, they assume that the adversary also learns the leakage related to document size, which can help the adversary to identify the encrypted identifiers.

(3) Our attack uses the proposed leakages derived from original index and performs equality test to output the corresponding query and keyword pairs. More specifically, it first uses the newly defined leakage from the document (e.g., document co-occurrence pattern and individual volume patten) to recover the relationship of plaintext and encrypted identifiers and then leverages these identifiers to further check the correctness of the candidates by observing the candidates' occurrence. Notably, recovering encrypted identifiers and narrowing down the query candidates by observing the keyword occurrence also play an important role in advancing the attack performance. Compared to the Subgraph attack, using newly defined document level leakage to recover the identifier is one of the core contributions in our work.

Evaluation of LAA on Fully Known Dataset. We first run our attack over two datasets, where the keyword space consists of the randomly selected keywords in the database. Figure 5(a) and 5(b) report the query recovery rate for the dataset composed with various sizes of keyword spaces. From the figures, we learn that the average recovery rate of our attack on Enron dataset and IMDB dataset are about from 80% and 98%, respectively. The recovery rate of the attack has a slight drop-off as keyword space increases, because the portion of queries with lower frequencies and weak co-relationship enlarges, particularly for those appear only once in

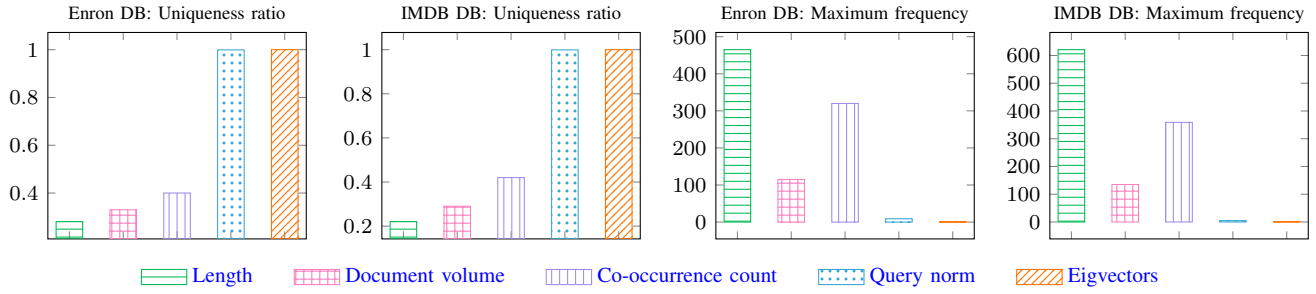
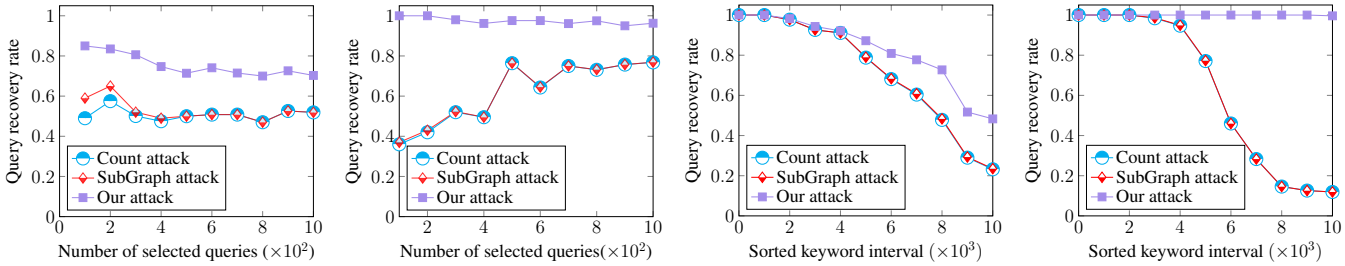
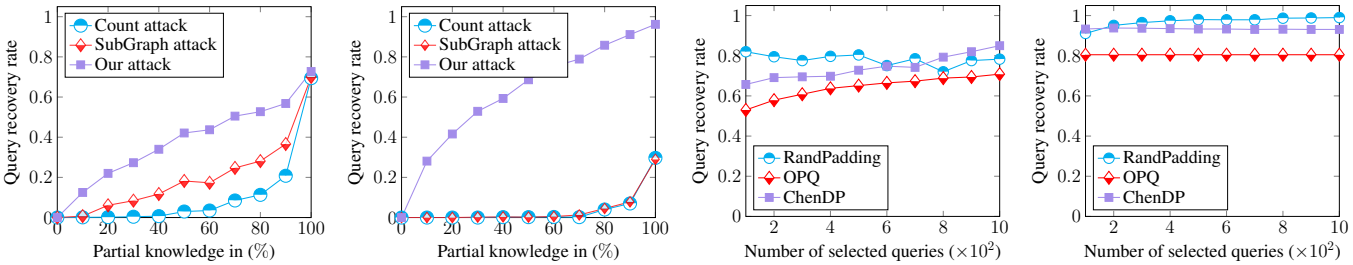


Fig. 4. Statistics of refined leakages (invariant characteristics) derived from the leakage profiles



(a) Attack v.s random queries in Enron (b) Attack v.s random queries in IMDB (c) Attack v.s selected queries in Enron (d) Attack v.s selected queries in IMDB

Fig. 5. The query recovery rate for known prior knowledge via prior attacks and our attack



(a) Attack v.s random queries in Enron (b) Attack v.s random queries in IMDB (c) Our attack v.s defense on Enron (d) Our attack v.s defenses on IMDB

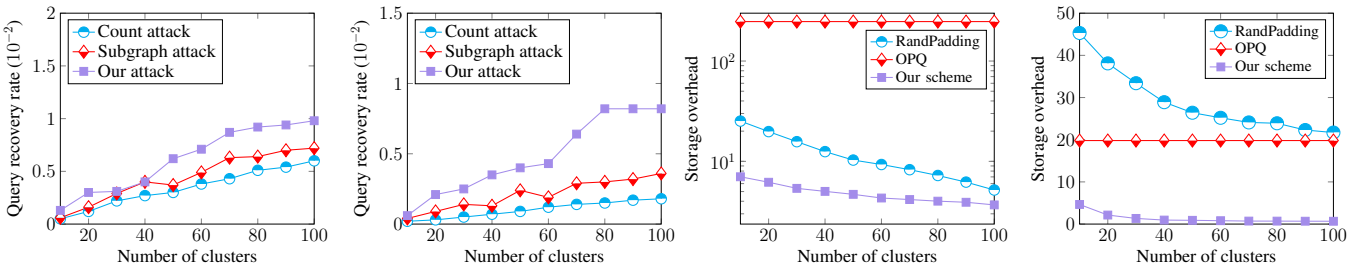
Fig. 6. The query recovery rate of different attacks over partially known datasets and the recovery rate of our attack over different SSE countermeasures. Here RandPadding denotes the scheme proposed by Cash et al. [17], OPQ denotes the SSE scheme proposed by Blackstone et al. [9] which deploys response-hiding multi-map encryption scheme to store documents, and ChenDP denotes the differential private access pattern scheme proposed by Chen et al. [60]

the database. In this case, the adversary may hardly distinguish the candidates of the query by poor leakage. Based on prior empirical results that the count attack cannot work with low-occurrence queries [9], we suspect that the occurrences of the keywords in database may affect the attack performance.

To validate our observation and further study the above impact on the robustness and effectiveness of the attacks, we conduct specialized experiments under more systematic settings. Specifically, we sort all keywords in the database according to their number of occurrences (frequencies) in descending order, and then select the keywords in different intervals from the keyword space, i.e., top 1000, top 1000 - top 2000, etc. After that, we run our attack and record the results in Fig. 5(c) and 5(d), respectively. The attack results clearly show that the recovery rate reduces smoothly as the query occurrences decline. In the IMDB dataset, the recovery rates of our attack stay high, achieving 98%, even the keyword occurrence ranks after 10,000. Through analyzing these unrecovered queries, we find that for each of these uncovered queries, there is at least one query that has the same access pattern with it, so it cannot be recovered at all.

In the above or next below experiments, we find that SubGraph attack and count attack often have the similar query recover rate. Because SubGraph attack relies heavily on the distribution of documents. Namely, it only suits for the dataset that has sufficient number of documents with distinctive sizes, but our test dataset don't fit this, in particular IMDB.

Evaluation of LAA on Partially Known Dataset. Our attack performs well even in the face of the scenario that only a partial dataset is disclosed to the adversaries. As seen in Fig. 6(a) and 6(b), the recovery rate line falls smoothly when the known-data rate decreases. Our attack still achieves 70.4% recovery rate when 50% datasets loses in IMDB dataset. Even in the Enron dataset with less-occurrence keywords, the recovery rate reaches up to 41.5%. Such positive attack results benefit from the adoption of the identifier recovery procedure, it first finds the exact matching plaintext and encrypted datasets from the existing ones and then launch the attack in the same way as the fully known datasets setting. Another observation in our work is that as the adversary can recover the encrypted identifiers with a high probability, current mitigations by adding random bogus indexes may be



(a) Attacks on our hardening Enron (b) Attacks on our hardening IMDB (c) Storage overhead of Enron (d) Storage overhead of IMDB

Fig. 7. The evaluation of the performance on our our countermeasure from effectiveness and efficiency

invalid (shown next in evaluations on countermeasures).

Comparison with Prior Attacks. We here compare our attack with count attack and SubGraph attack. We start by comparing their recovery rates of fully knowledge known attacks. From Fig 5(a), our attack can recover $18\% - 30\%$ more queries than count attack and SubGraph attack in Enron dataset,and this difference enlarges to 60% when we select IMDB as the target database (as seen in Fig 5(b)). The first reason is that our attack learns finer leakage from the database, which can better distinguish different candidates. The other reason is the using of identifier recovery function, which compensates the information loss in co-occurrence counts.

Moreover, the combination of the two strategies contributes to the robust execution of our attack against prior countermeasures. Such improvement is more visible in Fig. 6(c) and Fig. 6(d). Here we choose Cash et al's random padding [17], Chen et al.'s differentially-private access pattern [60] and Blackstone et al's OPQ scheme [9] as the targeted countermeasures. From the figure, we can see that all these three countermeasures work little against our attack, the recovery rate is still up to 53% and more. These results demonstrate that when capturing much finer leakage, padding only focuses on hiding some certain type of leakage like result length or access pattern is far from enough. This guides us to design comprehensive countermeasures.

Regarding the partially known dataset scenario, the benefits of employing identifier recovery also stretch. As seen in Fig 6(b), when 50% datasets loses in IMDB dataset, our attack preserves a high recovery rate at 70.4%, while the rate has dropped to 0 in other attacks. Here the reason SubGraph attack performs better than count attack in the Enron dataset is that it exploits the leakage of document volume/size, which can help recover identifiers and queries, and mitigates the downturn of recovery rate. For attacks on dataset without unique result length, the advancement of our attack is also obvious.

C. Evaluation on Countermeasure

In this section, we evaluate the performance of the countermeasures on effectiveness and efficiency through a sequence of experiments.

Countermeasure Setup. In this experiment, we make comparisons among Cash et al. [17], Blackstone et al. [9] and our countermeasure, which are designed for SSE that is adopted in document storage system. In Cash et al.'s scheme, they consider a scheme (denoted as RandPadding) in which the number of entries in each index row is padded up to the nearest multiple of an integer n. In Blackstone et al.'s scheme (termed OPQ), they encrypt the same documents into different copies for different keyword-identifier pairs, so as to hiding the access pattern. And for our countermeasures, we directly applying Bost et al.'s clustering technique in [12] to effectively reduce the storage and query communication cost [21], we regard it as the baseline for comparison to demonstrate the security and efficiency of our design. In addition, the hamming distance d of two rows is set to be half of the maximum hamming weight ℓ of rows in the cluster.

All evaluations are conducted on the Enron and IMDB dataset, and padding is implemented by adding random identifiers into the database to protect the result lengths of queries. Before we analyze our results, we review Bost et al's padding approaches. In this padding, all keywords are sorted by frequencies, and grouped together into clusters. Then the size of each query result in a cluster is padded to the maximum size of query result in this cluster. Regarding the parameter setting, we set the number of clusters from 10 to 100. Our countermeasure follows the same setting of Bost et al's padding, where keywords are grouped first and then the encoding algorithm is run on each group. In particular, the hamming weight of our binary perturbation algorithm (i.e., the expected result length after bit flipping) in Fig. 3 will be set as the maximum size of query result in this cluster.

Evaluation on Countermeasures. Now we follow the above framework to evaluate the performance of our countermeasure. We first measure the query recovery rate to quantify the security strength from the practical layer. We equip the selected databases with different countermeasures and then run our attack algorithm. The attack results are illustrated in Fig. 7(a) and Fig. 7(b), they demonstrate that our design defeats the proposed attack well. For example, when number of keyword space is 10,000, the recovery ratio of different attacks against our countermeasure on Enron dataset falls below than 1%, respectively. The recovery rate of our attack is also higher than other two selected attacks, which confirms the contribution of refined leakage to our attack performance again.

Regarding efficiency, we evaluate the storage overhead brought by our countermeasures. Figure 7(c) and 7(d) depict the storage overhead of our countermeasures. As shown in the figure, when the size of keyword space is 1.0×10^4 and the number of cluster is 10, our countermeasure in Enron dataset introduces $7.02\times$ storage overhead. Besides, when the number of clusters is increased, the total storage overhead decreases

accordingly. For example, the storage overhead of our design is reduced to $3.68\times$ when the number of size increases to 100. **Comparison with Prior Countermeasures.** Here we show performance comparison between our countermeasure and prior works. Without considering the weak effectiveness of prior countermeasures, from Fig. 7(c) and Fig. 7(d), we can find that our padding countermeasure appears to be more efficiency than priors. Specifically, in Enron email dataset, the Blackstone et al's OPQ scheme introduces almost $247.3\times$ storage overhead to achieve the security notion, this is because keywords in Enron dataset always have a large number of co-occurrence counts. Cash et al.'s countermeasure requires $25\times$ storage overhead which realizes on the distribution of document volume. Compared to work, our countermeasure is cost-effective in effectiveness and efficiency.

VIII. Conclusion

In this paper, we conduct systematic investigations on leakage exploitation of SSE from the algebraic perspective, and provide corresponding mitigation. Specifically, we model SSE as linear transformation and point out that the invariant characteristics during the transformation are the key contributing to privacy risk of SSE. Based on these findings, we design an attack that generalizes existing attacks and improve robustness and efficiency of them. Accordingly, we propose a new security notion to refine the prior ones and develop a two-phase countermeasure to protect the access pattern and volume information of the database, so as to mitigate existing LAAs. Our evaluation results show that our countermeasure further reduces the query recovery ratio compared to prior arts. In addition, our work also leaves a two open questions. The first is whether our attack and defense methodologies can be extended to the dynamic constructions, which support addition and deletion operations. The second question is to investigate the countermeasures for constructions with more advanced queries. We believe that our investigations will establish theoretical foundations towards exploring the fundamental security limit of searchable encryption and designing effective hardening techniques to push forward the scientific frontier of this research area.

Acknowledgment

This work was supported in part by the Natural Science Foundation of Jiangsu Province under Grant BK20210330, by the National Natural Science Foundation of China under Grant 62072240, by the National Key Research and Development Program of China under Grant 2020YFB1804604, by the Research Grants Council of Hong Kong under Grant CityU 11217819, Grant CityU 11217620, and Grant R6021-20F, by the ARC Discovery Projects under Grant DP200103308, and by Laboratory for AI-Powered Financial Technologies.

References

[1] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in *Proc. of ACM CCS*, 2006.

[2] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "Cryptdb: protecting confidentiality with encrypted query processing," in *Proc. of SOSP*, 2011.

[3] D. Cash, S. Jarecki, C. S. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries," in *Proc. of CRYPTO*, 2013.

[4] S. Tu, M. F. Kaashoek, S. Madden, and N. Zeldovich, "Processing analytical queries over encrypted data," in *Proc. VLDB Endow.*, 2013.

[5] S. Kamara and T. Moataz, "SQL on structurally-encrypted databases," in *Proc. of ASIACRYPT*, 2018.

[6] D. Cash, J. Jaeger, S. Jarecki, C. S. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, "Dynamic searchable encryption in very-large databases: Data structures and implementation," in *Proc. of NDSS*, 2014.

[7] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: Ramification, attack and mitigation," in *Proc. of NDSS*, 2012.

[8] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, "Leakage-abuse attacks against searchable encryption," in *Proc. of ACM CCS*, 2015.

[9] L. Blackstone, S. Kamara, and T. Moataz, "Revisiting leakage abuse attacks," in *Proc. of NDSS*, 2020.

[10] M. Naveed, S. Kamara, and C. V. Wright, "Inference attacks on property-preserving encrypted databases," in *Proc. of ACM CCS*, 2015.

[11] D. Pouliot and C. V. Wright, "The shadow nemesis: Inference attacks on efficiently deployable, efficiently searchable encryption," in *Proc. of ACM CCS*, 2016.

[12] R. Bost and P.-A. Fouque, "Thwarting leakage abuse attacks against searchable encryption – a formal approach and applications to database padding," Cryptology ePrint Archive, 2017.

[13] S. Kamara and T. Moataz, "Computationally volume-hiding structured encryption," in *Proc. of EUROCRYPT*, 2019.

[14] S. Patel, G. Persiano, K. Yeo, and M. Yung, "Mitigating leakage in secure cloud-hosted data structures: Volume-hiding for multi-maps via hashing," in *Proc. of ACM CCS*, 2019.

[15] S. Kamara, T. Moataz, and O. Ohrimenko, "Structured encryption and leakage suppression," in *Proc. of CRYPTO*, 2018.

[16] Z. Gui, K. G. Paterson, S. Patranabis, and B. Warinschi, "Swissse: System-wide security for searchable symmetric encryption," *IACR Cryptol. ePrint Arch.*, 2020.

[17] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, "Leakage-abuse attacks against searchable encryption," *IACR Cryptology ePrint Archive*, vol. 2016, p. 718, 2016.

[18] Z. Qin, T. Yu, Y. Yang, I. Khalil, X. Xiao, and K. Ren, "Generating synthetic decentralized social graphs with local differential privacy," in *Proc. of the 2017 ACM CCS*, 2017.

[19] M. Mohammady, L. Wang, Y. Hong, H. Louafi, M. Pourzandi, and M. Debbabi, "Preserving both privacy and utility in network trace anonymization," in *Proc. of ACM CCS*, 2018.

[20] N. Li, M. Lyu, D. Su, and W. Yang, *Differential Privacy: From Theory to Practice*, ser. Synthesis Lectures on Information Security, Privacy, & Trust. Morgan & Claypool Publishers, 2016.

[21] I. Demertzis, D. Papadopoulos, C. Papamanthou, and S. Shintre, "SEAL: attack mitigation for encrypted databases via adjustable leakage," in *Proc. of Usenix Security*, 2020.

[22] D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. of IEEE S&P*, 2000.

[23] D. Boneh, K. Lewi, M. Raykova, A. Sahai, M. Zhandry, and J. Zimmerman, "Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation," in *Proc. of EUROCRYPT*, 2015.

[24] S. Faber, S. Jarecki, H. Krawczyk, Q. Nguyen, M. Rosu, and M. Steiner, "Rich queries on encrypted data: Beyond exact matches," in *Proc. of ESORICS*, 2015.

[25] I. Demertzis, S. Papadopoulos, O. Papapetrou, A. Deligiannakis, and M. N. Garofalakis, "Practical private range search revisited," in *Proc. of ACM SIGMOD*, 2016.

[26] D. Cash, F. Liu, A. O'Neill, M. Zhandry, and C. Zhang, "Parameter-hiding order revealing encryption," in *Proc. of ASIACRYPT*, 2018.

[27] I. Demertzis, S. Papadopoulos, O. Papapetrou, A. Deligiannakis, M. N. Garofalakis, and C. Papamanthou, "Practical private range search in depth," *ACM Trans. Database Syst.*, vol. 43, no. 1, pp. 2:1–2:52, 2018.

[28] S. Kamara and T. Moataz, "Boolean searchable symmetric encryption with worst-case sub-linear complexity," in *Proc. of EUROCRYPT*, 2017.

[29] M. Chase and S. Kamara, "Structured encryption and controlled disclosure," in *Proc. of ASIACRYPT*, 2010.

[30] M. Bellare, M. Fischlin, A. O'Neill, and T. Ristenpart, "Deterministic encryption: Definitional equivalences and constructions without random oracles," in *Proc. of CRYPTO*, 2008.

[31] S. Jarecki, C. S. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, "Outsourced symmetric private information retrieval," in *Proc. of ACM CCS*, 2013.
[32] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proc. of TCC*, 2007.
[33] S. Lai, S. Patranabis, A. Sakzad, J. K. Liu, D. Mukhopadhyay, R. Steinfeld, S. Sun, D. Liu, and C. Zuo, "Result pattern hiding searchable encryption for conjunctive queries," in *Proc. of the ACM CCS*, 2018.
[34] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proc. of ACM CCS*, 2012.
[35] S. Kamara and C. Papamanthou, "Parallel and dynamic searchable symmetric encryption," in *Proc. of FC*, 2013.
[36] I. Miers and P. Mohassel, "IO-DSSE: scaling dynamic searchable encryption to millions of indexes by improving locality," in *Proc. of NDSS*, 2017.
[37] P. Mishra, R. Poddar, J. Chen, A. Chiesa, and R. A. Popa, "Oblix: An efficient oblivious search index," in *Proc. of IEEE S&P*, 2018.
[38] A. Ahmad, K. Kim, M. I. Sarfaraz, and B. Lee, "OBLIVIATE: A data oblivious filesystem for intel SGX," in *Proc. of NDSS*, 2018.
[39] C. Priebe, K. Vaswani, and M. Costa, "Enclavedb: A secure database using SGX," in *Proc. of IEEE S&P*, 2018.
[40] S. Oya and F. Kerschbaum, "Hiding the access pattern is not enough: Exploiting search pattern leakage in searchable encryption," in *Proc. of USENIX Security*, 2021.
[41] G. Kellaris, G. Kollios, K. Nissim, and A. O'Neill, "Generic attacks on secure outsourced databases," in *Proc. of ACM CCS*, 2016.
[42] M. Lacharité, B. Minaud, and K. G. Paterson, "Improved reconstruction attacks on encrypted data using range query leakage," in *Proc. IEEE S&P*, 2018.
[43] P. Grubbs, M. Lacharité, B. Minaud, and K. G. Paterson, "Pump up the volume: Practical database reconstruction from volume leakage on range queries," in *Proc. of ACM CCS*, 2018.
[44] P. Grubbs, K. Sekniqi, V. Bindschaedler, M. Naveed, and T. Ristenpart, "Leakage-abuse attacks against order-revealing encryption," in *Proc. of IEEE S&P*, 2017.
[45] P. Grubbs, M. Lacharité, B. Minaud, and K. G. Paterson, "Learning to reconstruct: Statistical learning theory and encrypted database attacks," in *Proc. of IEEE S&P*, 2019.
[46] Z. Gui, O. Johnson, and B. Warinschi, "Encrypted databases: New volume attacks against range queries," in *Proc. of ACM CCS*, 2019.
[47] E. M. Kornaropoulos, C. Papamanthou, and R. Tamassia, "The state of the uniform: Attacks on encrypted databases beyond the uniform query distribution," in *Proc. of S&P*, 2020.
[48] F. Falzon, E. A. Markatou, Akshima, D. Cash, A. Rivkin, J. Stern, and R. Tamassia, "Full database reconstruction in two dimensions," in *Proc. of ACM CCS*, J. Ligatti, X. Ou, J. Katz, and G. Vigna, Eds., 2020.
[49] E. M. Kornaropoulos, C. Papamanthou, and R. Tamassia, "Response-hiding encrypted ranges: Revisiting security via parametrized leakage-abuse attacks," in *Proc. of IEEE S&P*, 2021.
[50] K. Lewi and D. J. Wu, "Order-revealing encryption: New constructions, applications, and lower bounds," in *Proc. of ACM CCS*, 2016.
[51] E. M. Kornaropoulos, C. Papamanthou, and R. Tamassia, "Data recovery on encrypted databases with k-nearest neighbor query leakage," in *Proc. of S&P*, 2019.
[52] Y. Zhang, J. Katz, and C. Papamanthou, "All your queries are belong to us: The power of file-injection attacks on searchable encryption," in *Proc. of USENIX Security*, 2015.
[53] S. Wang, R. Poddar, J. Lu, and R. A. Popa, "Practical volume-based attacks on encrypted databases," in *Proc. of EURO S&P*, 2019.
[54] T. H. Chan, K. Chung, B. M. Maggs, and E. Shi, "Foundations of differentially oblivious algorithms," in *Proc. of SODA*, 2019.
[55] E. Stefanov, M. van Dijk, E. Shi, T. H. Chan, C. W. Fletcher, L. Ren, X. Yu, and S. Devadas, "Path ORAM: an extremely simple oblivious RAM protocol," *Journal of ACM*, vol. 65, no. 4, pp. 18:1–18:26, 2018.
[56] E. Stefanov, C. Papamanthou, and E. Shi, "Practical dynamic searchable encryption with small leakage," in *Proc. of NDSS*, 2014.
[57] M. Maffei, G. Malavolta, M. Reinert, and D. Schröder, "Privacy and access control for outsourced personal records," in *Proc. IEEE S&P*, 2015.
[58] V. Bindschaedler, M. Naveed, X. Pan, X. Wang, and Y. Huang, "Practicing oblivious access on cloud storage: the gap, the fallacy, and the new way forward," in *Proc. of ACM CCS*, 2015.
[59] E. Stefanov, M. van Dijk, E. Shi, C. W. Fletcher, L. Ren, X. Yu, and S. Devadas, "Path ORAM: an extremely simple oblivious RAM protocol," in *Proc. of ACM CCS*, 2013.
[60] G. Chen, T. Lai, M. K. Reiter, and Y. Zhang, "Differentially private access patterns for searchable symmetric encryption," in *Proc. of INFOCOM*, 2018.
[61] A. Agarwal, M. Herlihy, S. Kamara, and T. Moataz, "Encrypted databases for differential privacy," *PoPETs*, vol. 2019, no. 3, pp. 170–190, 2019.
[62] C. Wang, J. Bater, K. Nayak, and A. Machanavajjhala, "Dp-sync: Hiding update patterns in secure outsourced databases with differential privacy," in *Proc. of SIGMOD*, 2021.
[63] G. A. Korn and T. M. Korn, *Mathematical handbook for scientists and engineers: definitions, theorems, and formulas for reference and review*. Courier Corporation, 2000.
[64] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge University Press, 2008.
[65] L. Xu, X. Yuan, C. Wang, Q. Wang, and C. Xu, "Hardening database padding for searchable encryption," in *Proc. of IEEE INFOCOM*, 2019.
[66] Enron Email Dataset, Online at: https://www.cs.cmu.edu/~./enron, 2015.
[67] IMDB Movie Comments Dataset, Online at: https://datasets.imdbws.com, 2020.