Rethinking Searchable Symmetric Encryption

Zichen Gui
University of Bristol

Kenneth G. Paterson
ETH Zürich

Sikhar Patranabis
ETH Zürich

December 10, 2021

Abstract

Symmetric Searchable Encryption (SSE) schemes enable keyword searches over encrypted documents. To obtain efficiency, SSE schemes incur a certain amount of leakage. The vast majority of the literature on SSE considers only leakage from one component of the overall SSE system, the *encrypted search index*. This component is used to identify which documents to return in response to a keyword query. The actual fetching of the documents is left to another component, usually left unspecified in the literature, but generally envisioned as a simple storage system matching document identifiers to encrypted documents.

This raises the question: do SSE schemes actually protect the security of data and queries when considered from a system-wide viewpoint? We answer this question in the negative. We do this by introducing a new inference attack that achieves practically efficient, highly scalable, accurate query reconstruction against end-to-end SSE systems. In particular, our attack works even when the SSE schemes are built in the natural way using the state-of-the-art techniques (namely, volume-hiding encrypted multi-maps) designed to suppress leakage and protect against previous generations of attack.

A second question is whether the state-of-the-art leakage suppression techniques can instead be applied on a system-wide basis, to protect both the encrypted search index and the encrypted document store, to produce *efficient* SSE systems. We also answer this question in the negative. To do so, we implement SSE systems using those state-of-the-art leakage suppression methods, and evaluate their performance. We show that storage overheads range from $100\times$ to $800\times$ while bandwidth overheads range from $20\times$ to $100\times$, as compared to a naïve baseline system.

Our results motivate the design of new SSE systems that are designed with system-wide security in mind from the outset. In this regard, we show that one such SSE system due to Chen *et al.* (IEEE INFOCOM 2018), with provable security guarantees based on differential privacy, is also vulnerable to our new attack.

In totality, our results force a re-evaluation of how to build end-to-end SSE systems that offer both security and efficiency.

1 Introduction

Searchable Symmetric Encryption. Database encryption is a key enabler for secure storage-as-a-service, wherein clients can securely outsource the storage and processing of large databases to (potentially untrusted) third party servers. Searchable symmetric encryption (SSE) [SWP00, CGKO06, CK10, CJJ$^+$13] is a special sub-class of database encryption that aims to efficiently support search queries over symmetrically encrypted databases. The core functionality enabled by SSE is the following: given an encrypted document collection in which each document is tagged with keywords, find the set of all documents tagged with a given keyword w. In this paper, we focus primarily on SSE for *static* document collections. This has historically received the most attention.

Leakage. The term "leakage" is popularly used in the SSE literature to denote any information that the server learns about either the database itself or the queries made by the client. For any SSE scheme, leakage can be of two kinds: *setup leakage* – information learnt by the server from the encrypted database it receives at setup (i.e., prior to any query execution), and *query leakage* – information learnt by the server from the query token and the interaction between the query token and the encrypted database. Informally, an SSE scheme is "more" secure if it incurs "less" leakage. Ideally, an SSE scheme should be leakage-free, but this comes at the cost of huge performance overheads. The vast majority of SSE schemes incur some leakage as the price for acceptable efficiency (e.g. [CGKO06, CK10, KPR12, CJJ$^+$13, CJJ$^+$14, SPS14, NPG14, FJK$^+$15, Bos16, BMO17, KM17, KM18, CPPJ18]).

Leakage Cryptanalysis. A natural question to ask is: *how should we assess the impact of leakage on the real-world security of SSE?* The practice commonly adopted in the SSE literature is to perform *leakage cryptanalysis.* This involves developing concrete cryptanalytic attacks that exploit the leakage to subvert some security guarantee (such as data/query privacy) of the SSE scheme. Starting with the seminal work of Islam et al. [IKK12], leakage cryptanalysis has been studied extensively in the context of SSE for document collections [CGPR15, PW16, GPPW20, BKM20, OK21a, OK21c, NHP$^+$21, DHP21]. The commonly studied leakage profiles for SSE are:

- **Response Length.** For a given query on a keyword w, the response length (or volume) leakage reveals the size of the query response set, i.e., number of documents containing w.
- **Access Pattern.** For a given query on a keyword w, the access pattern leakage reveals the set of (potentially randomized) identifiers for documents containing w.
- **Co-occurrence Pattern.** For a pair of queries over keywords w_i and w_j, the co-occurrence leakage reveals the number of documents containing *both* w_i and w_j.
- **Search Pattern.** For a pair of queries over keywords w_i and w_j, the search pattern leakage reveals whether w_i and w_j are identical.

Structure-Only SSE. The *structure-only* approach to designing SSE schemes was introduced and formalized by Chase and Kamara in [CK10]. In this approach, a search query

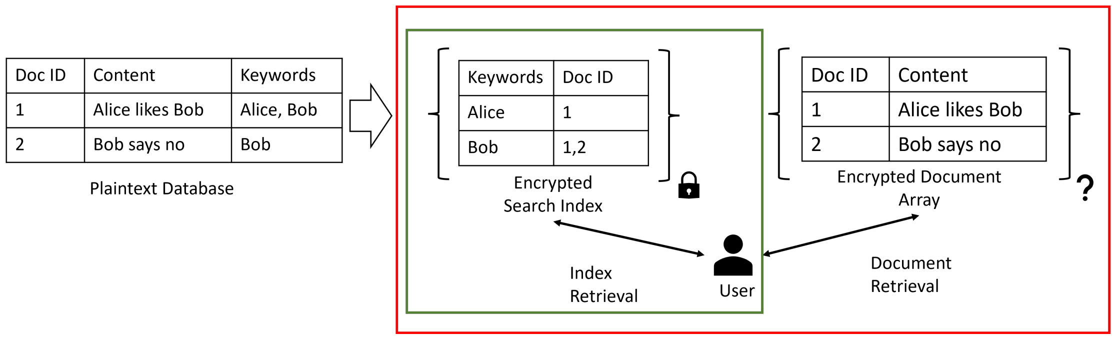


Figure 1: A "system-wide" view of structure-only SSE schemes (e.g., [CGKO06, CK10, KPR12, CJJ+13, CJJ+14, SPS14, NPG14, FJK+15, Bos16, BMO17, KM17, KM18, CPPJ18, KM19, PPYY19]). While structure-only SSE design focuses only on leakage from the encrypted search index (depicted by the inner green box), end-to-end SSE design focuses on system-wide leakage (depicted by the outer red box).

is broken down into two phases. The first phase, called the index retrieval phase, uses a specially designed *encrypted search index* to efficiently recover the set of (encrypted) document identifiers corresponding to documents matching the query. The second phase is the document retrieval phase, in which the client *actually* fetches the encrypted documents matching the query.

We illustrate the structure-only approach to SSE in Figure 1. The inner green box in Figure 1 depicts an structure-only SSE sub-system, while the outer red box depicts the SSE system-as-a-whole. In the rest of the paper, we use the term *structure-only SSE* to refer to the sub-system depicted by the inner green box, and *end-to-end SSE* to refer to the system-as-a-whole depicted by the outer red box.

The vast majority of SSE schemes follow the structure-only approach approach. Examples include [CGKO06, CK10, KPR12, CJJ+13, CJJ+14, SPS14, NPG14, FJK+15, Bos16, BMO17, KM17, KM18, CPPJ18]. These all focus only on designing a secure encrypted search index (i.e. the inner green box in Figure 1). None of them concretely specify how the document retrieval phase is to be executed. And, in each case, their security analysis considers only leakage from the index retrieval phase, ignoring any leakage from the document retrieval phase.

As the above list of papers shows, the structure-only approach has the become de facto standard one in the SSE literature. There exist only a handful of alternative design approaches that target end-to-end SSE with in-built leakage suppression for both index and document retrieval. These include SSE with differentially private access-patterns [CLRZ18] and SWiSSSE [GPPW20]. However, structure-only SSE schemes are not useful on their own unless they are used in conjunction with secure, efficient mechanisms for document retrieval. The research community's focus on the structure-only approach has yielded little progress on the question of how to build secure, efficient, *end-to-end* SSE systems.

Leakage Suppression in SSE. Motivated by the need to counter leakage cryptanalysis, recently proposed SSE schemes have started to use dedicated techniques to suppress setup and search leakage. The state-of-the-art is represented by volume-hiding encrypted multi-maps (EMMs) [KM19, PPYY19]. These follow the same structure-only approach described above. In particular, while it is clear how to apply volume-hiding EMMs to search indices to obtain low-leakage, structure-only SSE schemes, the authors of [KM19, PPYY19] do not specify how to design end-to-end SSE systems based on volume-hiding EMMs.

System-Wide Leakage in SSE. In this paper, we revisit the current dominant approach to leakage analysis for SSE by taking an alternative "system-wide" perspective. Consider two kinds of adversaries – one that observes the leakage from the encrypted search index (e.g., by passively corrupting the server that stores this index), and one that observes the leakage from the final query processing step on the encrypted documents themselves (e.g., by passively corrupting the server that stores the encrypted documents, potentially different from one storing the index). In terms of real-world security, any realization of end-to-end SSE should be secure against cryptanalysis of system-wide leakage.[1]

Since the vast majority of SSE schemes, including state-of-the-art schemes based on volume-hiding EMMs, are structure-only schemes, they only focus on security against the first kind of adversary. Consequently, it is unclear what kind of system-wide leakage is incurred when these schemes are used to build end-to-end SSE systems, and whether such leakage potentially leads to attacks. It is also unclear whether volume-hiding EMMs can be used to mitigate this leakage without compromising on efficiency.

1.1 Our Contributions

System-Wide Security of Structure-Only SSE. In this paper, we analyze the system-wide leakage that arises when state-of-the-art structure-only SSE schemes are used in the natural way to build end-to-end SSE systems. In particular, we focus on the simple construction where only the leakage from the index is mitigated using techniques such as volume-hiding EMMs, while the document retrieval is implemented using a straightforward look-up table of encrypted documents. In the absence of any statements to the contrary in the extensive literature, this seems to be the generally assumed mechanism. In this context, we ask the following:

Do structure-only SSE schemes result in secure end-to-end SSE systems when system-wide leakage is taken into account?

We answer this question in the negative. We show that all the structure-only SSE schemes [CGKO06, CK10, KPR12, CJJ+13, CJJ+14, SPS14, NPG14, FJK+15, Bos16, BMO17, KM17, KM18, CPPJ18], including those built from volume-hiding EMMs [KM19, PPYY19], incur damaging system-wide leakage when used to construct end-to-end SSE systems in the natural way. Concretely, we show that for all of the above-cited schemes, the leakage from document retrieval is actually different from and significantly more damaging than the

[1]A widely used assumption in the SSE literature is that these two adversaries are non-colluding. Our analysis also extends to this setting since the attack we develop targets leakage from document retrieval only (in fact, for all of our target schemes, the leakage from document retrieval subsumes the leakage from index retrieval).

leakage from index retrieval, and that this leakage can be exploited to launch a strong query-recovery attack.[2]

Our attack establishes that using structure-only SSE schemes in the natural way leads to end-to-end SSE systems that are insecure in practice. Fundamentally, this is because the careful leakage mitigation for the encrypted search index, obtained using techniques such as volume-hiding EMMs, is undermined by the system-wide leakage that arises from the document retrieval phase. This clearly motivates the need for new security definitions that carefully model system-wide leakage in end-to-end SSE systems, as well efficient techniques to mitigate such leakage.

System-Wide Leakage Mitigation. A natural approach to mitigating system-wide leakage in end-to-end-SSE systems would be to apply existing leakage suppression techniques such as volume-hiding EMMs [KM19, PPYY19] to *both* the encrypted index *and* the final document retrieval step. Intuitively, this enhances resistance to system-wide attackers, but potentially degrades efficiency. In this context, we now ask the following question:

Can existing leakage suppression techniques efficiently *mitigate system-wide leakage?*

We also answer this question in the negative. We demonstrate that it is practically infeasible to use volume-hiding EMMs [KM19, PPYY19] to hide system-wide leakage in structure-only SSE schemes. We validate this observation by experimentally evaluating the storage and query processing overheads incurred when applying EMMs to encrypt the whole database, and not just the search index. We concretely establish that, with currently available techniques, we can have either efficient and scalable structure-only SSE schemes that suffer from damaging system-wide leakage (as illustrated by our attack), or we can have end-to-end leakage-protected SSE systems that are inefficient in practice and do not scale to large databases.

Differentially Private Access Patterns. As a final contribution, we show that the end-to-end SSE system based on differentially private access patterns in [CLRZ18], which we henceforth call **DPAP-SE**, is also vulnerable to our new query recovery attack. Our attack breaks **DPAP-SE** for the *same* parameter set that the authors of [CLRZ18] advocated using to counter leakage-abuse attacks. While it is possible to degrade our attack's efficiency by altering the parameter set, this also greatly reduces the practical efficiency of the resulting scheme. Our attack does not invalidate the original security properties proven by the authors of [CLRZ18], but instead indicates that these security properties (and the corresponding usage of differential privacy techniques) are, in fact, insufficient in practice. In fact, the scheme in [CLRZ18] *does* resist naïve adaptations of existing cryptanalytic attacks [IKK12, CGPR15, CGPR16, BKM20] that work only for *unperturbed* leakage such as co-occurrence patterns and access patterns, as was claimed in [CLRZ18]. However, we demonstrate the possibility of a stronger attack that is *not* restricted to unperturbed leakage and, hence, bypasses the limitations of existing attacks.

[2]Note that even though the document retrieval phase is implemented straightforwardly, the leakage from this phase is not necessarily the trivial (unperturbed) access pattern leakage. In particular, since document retrieval is performed based on the encrypted document identifiers from index retrieval, any leakage mitigation techniques used in index retrieval may perturb the leakage arising in document retrieval. Exploiting this leakage for query-recovery is non-trivial, and is a novel aspect of our attack.

Putting everything together, we see that, while significant progress has been made over the last decade in designing secure structure-only SSE schemes, none of these schemes actually yield practically efficient end-to-end SSE systems with resistance to system-wide leakage attacks. At the same time, we show that one alternative design approach for end-to-end SSE design also suffers from damaging system-wide leakage. Indeed, the only candidate end-to-end SSE scheme that we do not break is SWiSSSE [GPPW20]. This is perhaps not surprising since, aside from [CLRZ18], it is (as far as we are aware) the only scheme that was designed with system-wide leakage in mind.

We expand further on our main contributions below.

1.2 Attack on System-Wide Leakage

We show that the vast majority of structure-only SSE schemes, including state-of-the-art schemes built from volume-hiding EMMs [KM19, PPYY19], incur damaging system-wide leakage when used in the natural way to build end-to-end SSE systems. We show that such system-wide leakage can be exploited to launch a query recovery attack, and in certain cases, document recovery attacks against these schemes. We experimentally validate the practical efficiency of our attacks over the Enron email corpus,[3] which is widely regarded as the standard choice of experimental database in the SSE literature [IKK12,BKM20,OK21b].

Modeling and Attacking System-Wide Leakage. In a prior work, Gui *et al.* [GPPW20] discussed system-wide leakage in SSE schemes, and its impact on the security guarantees of SSE schemes. However, they only presented a system-wide analysis of their own construction, SWiSSSE and did not address the impact of system-wide leakage for SSE schemes more generally. In this paper, we refine and extend the analysis of [GPPW20] to develop a full-fledged query reconstruction attack. We apply the attack against end-to-end SSE schemes built in the natural way from different variants of volume-hiding EMMs, including **PRT-EMM** [KM19], as well as **FP-EMM** and **DP-EMM** [PPYY19]. Our attack works for volume-hiding EMM implementations using the same parameters and the same design/implementation choices advocated in [KM19] and [PPYY19].

We show how to statistically model the noisy co-occurrence leakage pattern arising in SSE schemes using the different volume-hiding EMMs as a function of the keyword queries, the system-wide leakage, and auxiliary data in the form of an approximate version of the original database. We then use the resulting models to develop a new inference-style leakage-abuse attack that targets query reconstruction. The core idea of our attack is to solve an optimization problem, where the objective function is the statistical likelihood of observing a given assignment of keywords to queries, given the observed leakage and auxiliary data as prior information. We then maximise the objective function using simulated annealing; this corresponds to maximising the likelihood of the solution. Thus the simulated annealing, if it works, will produce "good" solutions in which many keywords in the solution are correctly assigned to queries. This approach requires careful mathematical analysis to derive the likelihood functions for each targeted scheme and to efficiently implement their evaluation on large sets of queries and leakage. Of course, one could also use any performant optimization

[3]`https://www.cs.cmu.edu/~./enron/`

Attack	Attack Assumption	Leakage Exploited	Additional Requirements	Perturbed Leakage?
IKK [IKK12]	Known-data	Co-occurrence pattern	Known queries	No
Count* [CGPR15]	Known-data	Co-occurrence pattern	Known queries	No
BKM20 [BKM20]	Known-data	Co-occurrence/access pattern	–	No
LEAP [NHP$^+$21]	Known-data	Co-occurrence pattern	–	No
Graph Matching Attacks** [PW16]	Inference	Co-occurrence pattern	–	Yes
SAP [OK21a]	Inference	Search and volume pattern	Auxiliary info on query frequency pattern	Yes
GPPW20*** [GPPW20]	Inference	Co-occurrence pattern	–	Yes
IHOP [OK21c]	Inference	Search/volume/co-occurrence pattern	Auxiliary info on the target leakage(s)	Yes
DHP21 [DHP21]	Inference	Co-occurrence pattern	Known queries	No
This work	Inference	Co-occurrence pattern	–	Yes

Table 1: Comparison of existing leakage-abuse attacks achieving query reconstruction based on co-occurrence and/or access pattern leakage. *Count attack does not need known queries if the entire database is known by the attacker; known queries are only helpful when only part of the database is known by the attacker. **The attack targets in [PW16] only have weakly perturbed leakage, while our attack remains robust in the presence of significantly larger perturbation of the leakage. ***Gui *et al.* [GPPW20] also presented a system-wide leakage based attack; however, their attack techniques are specifically designed for cryptanalyzing their own construction called SWiSSSE (unlike our attacks, their attack cannot be generically used to break the vast majority of SSE schemes in the literature), and require significantly stronger assumptions on the auxiliary leakage available to the adversary compared to our attacks.

technique in place of simulated annealing.

We note that most well-known leakage-abuse attacks [IKK12, CGPR15, CGPR16, BKM20] rely on stronger models of auxiliary data: they are essentially *known-data* attacks, where the adversary knows a subset of the entries in the original database. On the other hand, our attack is an *inference* attack, where the auxiliary data about the database that is available to the adversary is *independent* but statistically "close" to the target database. Table 1 compares our attack to existing ones.

Experimental Evaluation. In Section 5, we present extensive experimental evaluations to validate the practicality of our proposed attack. Rather surprisingly, our apparently simple approach yields a very powerful system-wide leakage-abuse attack. Our experiments show that our attacks achieve high success rate with reasonable practical efficiency even if: (a) the target SSE scheme uses aggressive security parameters (beyond those advocated for the employed EMMs); (b) the keyword universe of the auxiliary information is significantly larger than the set of queried keywords (in contrast to existing attacks); and (c) the auxiliary information available to the adversary is very "noisy". Our attacks achieve over 60% query reconstruction rate in most of the settings we have tested. These include settings where the target SSE schemes use aggressive security parameters even beyond those originally proposed, as well as settings where the auxiliary information available to the adversary is significantly perturbed. With accurate auxiliary information, our attacks can achieve over 80% query reconstruction rate. Beyond query reconstruction, our attacks are also capable of (approximate) database reconstruction in certain cases (see Section 4.3 for more details).

1.3 On Using EMMs to Hide System-Wide Leakage

We also address the question of whether existing leakage suppression techniques can be efficiently applied to *both* the encrypted index *and* the final document retrieval step in order to mitigate system-wide leakage in end-to-end SSE systems. We demonstrate that it is practically infeasible to use volume-hiding EMMs, including the **PRT-EMM** scheme of [KM19], as well as the **FP-EMM** and **DP-EMM** schemes of [PPYY19], to hide system-wide leakage. We validate this observation by experimentally evaluating the concrete storage overheads and query processing overheads incurred when applying these state-of-the-art EMM techniques to encrypt the whole database, and not just the search index. We also study the overheads when using state-of-the-art ORAM and PIR schemes to do this.

Our experiments show that applying the volume-hiding EMM schemes from [KM19,PPYY19] to encrypt the whole database incurs prohibitively expensive storage overheads, ranging from 100 to 800 times the size of a naïve encrypted document store. Our experiments also establish that the volume hiding EMM schemes in [KM19,PPYY19] incur additional computational and communication overheads during query execution due to their usage of padding techniques. While this cost is manageable when querying the search index alone, it blows up to impractical proportions if applied directly to encrypted document retrieval, ranging from 20 to 100 times the communication costs for the naïve solution. Our volume-hiding EMM implementations use the same parameters and the same design/implementation choices advocated in [KM19] and [PPYY19].

2 Preliminaries

2.1 Syntax of Searchable Symmetric Encryption (SSE)

Let $\mathcal{T}$ be an abstract data type[4] supporting query operation **Query**. Then, an SSE scheme Σ for $\mathcal{T}$ is a tuple $\Sigma = (\mathbf{Setup}, \mathbf{Query}_e)$ where:

- **Setup** is the setup algorithm (locally executed by the client) which takes as input a plaintext database $\mathbf{D}$ of structure $\mathcal{T}$, and outputs a secret key sk and an encrypted database **ED**.
- $\mathbf{Query}_e$ is the query execution protocol between the client and server. The client takes as input a secret key sk and a query q (this could be a single keyword or a Boolean formula over multiple keywords), and the server takes as input the encrypted database **ED**; after the interaction between the client and server, the client obtains a final result rsp, which is the set of (encrypted) documents matching the query q.

Correctness. We say that Σ is correct if for any database $\mathbf{D}$ and any query q, an execution of the query protocol **Query** on the encrypted data $\mathbf{ED} \leftarrow \mathbf{Setup}(sk, \mathbf{D})$ yields the same response as a direct execution of the **Query** operation on the plaintext database $\mathbf{D}$ for query q.

Security. The security of an SSE scheme is defined formally with respect to a *leakage profile*, which is an upper bound on the information about the plaintext data and queries that an attacker can learn from the encrypted database **ED** and subsequent executions of $\mathbf{Query}_e$. We refer the reader to [CGKO06, CJJ+13] for the formal security definition.

[4]An abstract data type is a collection of data objects and a set of operations defined on those objects.

2.2 Notations

For the rest of our paper, we model a database in a searchable encryption scheme as target abstract data type. A database **DB** consists of a set of documents d_i, each associated to a set of keywords $\mathrm{kw}(\mathsf{d}_i)$, so $\mathbf{DB} = \{(\mathsf{d}_i, \mathrm{kw}(\mathsf{d}_i))\}$. It supports keyword search queries. For a keyword search query q on the keyword $\mathrm{kw}(\mathsf{q})$, the set of documents containing this keyword, denoted as $\mathbf{DB}(\mathrm{kw}(\mathsf{q})) = \{\mathsf{d}_i \mid \mathrm{kw}(\mathsf{q}) \in \mathrm{kw}(\mathsf{d}_i)\}$, is returned. To emphasize that we are only considering keyword search queries, we denote the query protocol as **Srch**.

3 Formal Description of Query Reconstruction Attacks using Co-Occurrence Leakage

This section formally establishes the leakage profile for SSE systems that we are targeting in the paper. We also formally describe our attack setting.

Access-pattern Leakage. Access-pattern leakage refers to the information leakage associated to document retrieval. If a naïve searchable encryption scheme Σ leaks the "exact" access pattern and nothing else, we can write the leakage of a query q on document collection **DB** as:

$$\mathcal{L}_{\mathbf{Srch}}(\mathsf{q}, \mathbf{DB}) = \{i \mid \mathrm{kw}(\mathsf{q}) \in \mathrm{kw}(\mathsf{d}_i), (\mathsf{d}_i, \mathrm{kw}(\mathsf{d}_i)) \in \mathbf{DB}\}\,,$$

where $\mathrm{kw}(\mathsf{q})$ denotes the keyword associated to query q.

Co-Occurrence Leakage. Access-pattern leakage from different queries can be represented equivalently as a matrix, known as co-occurrence pattern. Consider a small document collection **DB** where

$$\mathbf{DB} = \{(\mathsf{d}_1, \{\mathrm{kw}_1, \mathrm{kw}_2, \mathrm{kw}_3\}), (\mathsf{d}_2, \{\mathrm{kw}_1, \mathrm{kw}_2\}), (\mathsf{d}_3, \{\mathrm{kw}_3\})\}.$$

Let q_i be a query on keyword kw_i. If the original access pattern is leaked, we know that

$$\mathcal{L}_{\mathbf{Srch}}(\mathsf{q}_\ell, \mathbf{DB}) = \{1, 2\} \text{ for } \ell \in \{1, 2\}, \mathcal{L}_{\mathbf{Srch}}(\mathsf{q}_3, \mathbf{DB}) = \{1, 3\}\,.$$

This allows us to take intersections between the leakages as:

$$\begin{aligned}\mathcal{L}_{\mathbf{Srch}}(\mathsf{q}_1, \mathbf{DB}) \cap \mathcal{L}_{\mathbf{Srch}}(\mathsf{q}_2, \mathbf{DB}) &= \{1, 2\}\,,\\ \mathcal{L}_{\mathbf{Srch}}(\mathsf{q}_\ell, \mathbf{DB}) \cap \mathcal{L}_{\mathbf{Srch}}(\mathsf{q}_3, \mathbf{DB}) &= \{1\} \text{ for } \ell \in \{1, 2\}.\end{aligned}$$

The cardinality of the intersections can be very useful in an attack. For example, the co-occurrence pattern of the document collection above can be represented as a *co-occurrence matrix* $\bar{M}$ defined as follows:

$$\bar{M}(\mathsf{q}_1, \mathsf{q}_2, \mathsf{q}_3; \mathbf{DB}) = \begin{bmatrix} 2 & 2 & 1 \\ 2 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix},$$

where the (i, j)-th entry of the matrix is

$$\left|\mathcal{L}_{\mathbf{Srch}}(\mathsf{q}_i, \mathbf{DB}) \cap \mathcal{L}_{\mathbf{Srch}}(\mathsf{q}_j, \mathbf{DB})\right| .$$

If we know the underlying document collection perfectly, we can re-identify q_3 as a query on kw_3 as it is the only keyword that only shares one document with other keywords. This qualifies as a query reconstruction attack.

We note that a co-occurrence matrix contains strictly less information than the original access-pattern leakage as the information on intersections of more than two queries are removed. However, the co-occurrence matrix is often sufficient in attacks so it is used instead of the full leakage. We refer to the co-occurrence matrix as the co-occurrence leakage.

There are three complications to the representation of co-occurrence leakage in practice. Firstly, the schemes we consider in practice usually leak query equality pattern too. That is, if $\mathrm{kw}(\mathsf{q}_i) = \mathrm{kw}(\mathsf{q}_j)$, the attacker knows that the two queries are for the same keyword. In terms of co-occurrence leakage, we use only one of the queries in the representation to simplify the problem. Secondly, the queries are unordered in practice. That means there is no standard representation of the leakage in terms of the known keywords. We use the convention that the i-th row and column of the co-occurrence matrix corresponds to the i-th non-repeating query in our representation. Finally, not all schemes leak the original access pattern and some schemes may even be randomised. In those cases, we need to use a suitable representation of the co-occurrence information, which may differ from what we have described above.

Auxiliary Information. Similar to a co-occurrence matrix, the auxiliary information the attacker receives can be represented as a co-occurrence matrix M. The co-occurrence matrix is indexed by the known keywords and typically contains full information on all keywords. In stronger attacks, M is assumed to be noisy in the sense that it is not generated directly from the target document collection. Instead, an auxiliary dataset is used for the purpose.

Let $\mathbf{DB_{aux}} = \left\{(\mathsf{d}'_i, \mathrm{kw}(\mathsf{d}'_i))\right\}$ be an auxiliary document collection with keywords $\left\{\mathrm{kw}'_1, \ldots, \mathrm{kw}'_n\right\}$. In our attack, the (i, j)-th entry of M represents the empirical probability (derived from the auxiliary data $\mathbf{DB_{aux}}$) of seeing kw'_i and kw'_j together in a document, computed as:

$$M_{i,j} = \left|\left\{\mathsf{d}'_i \mid \mathrm{kw}'_i \in \mathrm{kw}(\mathsf{d}'_i) \wedge \mathrm{kw}'_j \in \mathrm{kw}(\mathsf{d}'_i)\right\}\right| / |\mathbf{DB_{aux}}|,$$

where $|\mathbf{DB_{aux}}|$ is the total number of auxiliary documents.

Attack Setting. Let Σ be a structured encryption scheme. Our attack exploits its co-occurrence leakage $\bar{M}(\cdot; \mathbf{DB})$ from retrieval of actual documents. If the scheme is an index-only one, we assume that the scheme used for document retrievals leaks the access pattern induced by the index-only scheme. We abuse the notation $\bar{M}(\cdot; \mathbf{DB})$ to mean co-occurrence leakage from document retrieval, and the said leakage is used in our query reconstruction attacks.

We can describe a query reconstruction attack formally as follows. Let queries $\mathsf{q}_1, \ldots, \mathsf{q}_l$ be a sequence of queries on the document collection, so the attacker observes co-occurrence leakage $\bar{M}(\mathsf{q}_1, \ldots, \mathsf{q}_l; \mathbf{DB})$. Suppose that the attacker has access to some auxiliary information M. *The goal of the attacker is to recover* $kw(\mathsf{q}_i)$ *after observing the co-occurrence leakage* $\bar{M}$ *and knowing auxiliary information* M.

4 New Query Reconstruction Attacks using Co-Occurrence Leakage

In this section, we show that all the structure-only SSE schemes [CGKO06, CK10, KPR12, CJJ+13, CJJ+14, SPS14, NPG14, FJK+15, Bos16, BMO17, KM17, KM18, CPPJ18], including those built from volume-hiding EMMs [KM19, PPYY19], incur damaging system-wide leakage when used to construct end-to-end SSE systems in the natural way. In particular, we propose new inference-style query recovery attacks that completely break the query privacy guarantees of these systems. We additionally show a new query recovery attack on the end-to-end SSE system **DPAP-SE** (based on differentially private access patterns) proposed in [CLRZ18].

Attack Overview. We achieve inference-style leakage-abuse attacks by using a combination of statistical modeling and simulated annealing, while assuming that the attacker has access to noisy co-occurrence leakage from observed queries and auxiliary data that is independent of but statistically "close" to the target database. We model the attack as an optimization problem, where the objective function is the statistical likelihood of observing a given assignment of keywords to queries, given the observed leakage and auxiliary data as prior information. Then maximizing the objective function corresponds to maximising the statistical likelihood of the solution. In this way, we obtain optimal assignments of keywords to queries, given the available leakage and auxiliary data. We use simulated annealing for the maximization step because it is relatively easy to implement and performs well in practice, although one could use any other suitable optimization technique..

In the rest of this section, we detail how we identify and mathematically model the (potentially "noisy") leakage information from each targeted scheme as a function of the auxiliary data and a given keyword assignment, and how we transform the resulting model into an appropriately structured input to the simulated annealing algorithm. We begin by describing the targets for our attacks more precisely.

4.1 Attack Targets

As mentioned earlier, we target system-wide leakage from end-to-end SSE systems constructed in a natural way from structure-only SSE schemes, which are in turn built from volume-hiding EMMs [KM19, PPYY19]. We begin by recalling the natural way to design end-to-end SSE from EMMs.

End-to-End SSE from EMMs. At a high level, a multi-map is a set of key-value pairs, i.e. $\{(\mathsf{key}_i, \overrightarrow{\mathsf{v}_i})\}$, where key_i's are the keys which are assumed to be non-repeating and $\overrightarrow{\mathsf{v}_i}$'s are the tuples of values associated to their keys. The natural way to design a structure-only

SSE scheme from an EMM is to realize an encrypted search index as follows: let each key of the EMM be a deterministic function of a keyword in the database, and let the corresponding value be the set of encrypted document identifiers pertaining to the documents matching this keyword. This is precisely the approach taken by all of the structure-only SSE schemes [CGKO06, CK10, KPR12, CJJ$^+$13, CJJ$^+$14, SPS14, NPG14, FJK$^+$15, Bos16, BMO17, KM17, KM18, CPPJ18] (while the exact specification/implementation of the EMM might vary from scheme-to-scheme depending on the nature of queries supported by the scheme, the overall approach is the same). Finally, the natural and widely accepted approach to transition from structure-only SSE to end-to-end SSE is to additionally realize document retrieval using a simple storage system matching document identifiers to encrypted documents. We refer the reader to [CGKO06, CK10, CJJ$^+$13] for a more formal exposition of the overall approach.

Our Targets. We target system-wide leakage from end-to-end SSE built in the aforementioned natural way from state-of-the-art volume-hiding EMMs [KM19, PPYY19], which are specially designed EMMs equipped with additional cryptographic mechanisms for leakage suppression. Our attacks rely on the leakage from the document retrieval phase, which subsumes the leakage from the index-retrieval phase. It is important to note that even though the document retrieval phase is implemented as a simple storage system, the leakage from this phase is *not* the trivial (unperturbed) access pattern leakage. In particular, since document retrieval is performed based on the encrypted document identifiers from index retrieval, the leakage mitigation from the volume-hiding EMM also induces noise/perturbation in the leakage from document retrieval. Hence, we specifically design our attacks to be robust against such noisy leakage. Since the exact nature of the leakage depends on the specific realization of volume-hiding EMM used, we recall the schemes from [KM19, PPYY19].

PRT-EMM. Our first target is an end-to-end SSE scheme realized naturally from the first construction of volume-hiding EMM via pseudorandom transform (abbreviated as **PRT-EMM** throughout) due to Kamara and Moataz [KM19]. At a high level, their idea is to pad or truncate the query response lengths of queries on any EMM with a pseudorandom function (PRF) as follows. Let key be a key for the multi-map and $F_{sk}(\cdot)$ be a PRF with key sk. The client computes: $n'_{\mathsf{key}} = \lambda + F_{sk}(\mathsf{key}||n_{\mathsf{key}})$, as the new query response length, where λ is a free parameter which the client can choose and n_{key} is the original query response length. These new query response lengths are used to build a multi-map on as follows:

- If $n_{\mathsf{key}} \leq n'_{\mathsf{key}}$, add $\perp$ symbols in the multi-map on key key before encryption.
- If $n_{\mathsf{key}} > n'_{\mathsf{key}}$, truncate the multi-map on keyword key to the first n'_{key} entries.

We note here that the original construction in [KM19] pads query responses with $\perp$ symbols if the real query response length is shorter. If the $\perp$ symbols are ignored in actual document retrieval, the attacker will be able to learn the true query response lengths when there is padding. In our attack, we assume the $\perp$ symbols are replaced by randomly picked indices, so the true query response lengths are not leaked.

Our attack targets "noisy" co-occurrence leakage during document retrieval in an end-to-end SSE scheme realized naturally from **PRT-EMM**. Due to lack of space, the detailed formal derivation of the precise leakage distribution for this SSE scheme is deferred to Appendix

A.

FP-EMM and DP-EMM. We next target end-to-end SSE scheme realized naturally from two constructions of volume-hiding EMM based on hash-tables presented in [PPYY19]. The first scheme (abbreviated as **FP-EMM** throughout) uses full padding, meaning that all query response lengths are padded to the maximum query response length. Concretely, this is done by querying additional addresses in the hash-table deterministically (generated by a PRF) for each key.

The second scheme (abbreviated as **DP-EMM** throughout) uses differentially-private volume hiding as opposed to full padding. Let $2n_{\mathsf{key}}$ be the true query response length of a query on key key (where the factor of 2 arises from the usage of two hash tables in Cuckoo hashing). Then the scheme pads the query response length to $2n_{\mathsf{key}} + n^* + \mathbf{Lap}_{sk}(2/\epsilon)$, where n^* is a parameter set by the client to offset the query response length in case the latter random variable is negative, and $\mathbf{Lap}_{sk}(\cdot)$ is a Laplace distribution with secret key sk as the seed.

As in the case of **PRT-EMM**, we design attacks targeting "noisy" co-occurrence leakage during document retrieval in end-to-end SSE schemes realized naturally from **FP-EMM** and **DP-EMM**, respectively. The formal derivation of the precise leakage distribution for each of these SSE schemes is again deferred to Appendix A.

DPAP-SE. Finally, we also target the end-to-end SSE system **DPAP-SE** (based on differentially private access patterns) proposed in [CLRZ18]. At a high level, this scheme uses a differential privacy mechanism is to "obfuscate" the plaintext database prior to encryption, such that a slight change in the real access pattern does not affect the obfuscated access pattern significantly. There are two key ingredients in their construction. First, an erasure code [DGWR07] is used to split every document into m shards, each with size $\frac{1}{k}$ of the original document. The erasure code has the property that any k shards of a document can be used to reconstruct the original document. The client then picks two probabilities p and q, and does the following to each shard:

1. For any keyword that is originally in the shard, remove the keyword with probability $1 - p$.
2. For any keyword that is originally not in the shard, add it to the shard with probability q.

For our attack, we rely on the observation that the resulting leakage from this scheme can be interpreted as follows: since the scheme effectively transforms query response into "noisy" keyword response lengths on the shards, the observed co-occurrence counts are actually "noisy" co-occurrence counts on the shards, rather than the actual documents. We then design an attack procedure that takes this noisy leakage on the shards into account. As in our earlier attacks, we again rely on precisely modeling this noisy co-occurrence leakage, albeit on the shards. Details can be found in Appendix A.

4.2 Attack Technique

We now detail our attack technique. For all of our attacks, we assume that the precise leakage modeling steps outlined above (and described formally in Appendix A) yield an observed co-occurrence matrix $\bar{M}$, which is the leakage from the observed queries on the actual target database. To simulate the auxiliary information available to the attacker, we build an auxiliary co-occurrence matrix M from an auxiliary database. This auxiliary database consists of a subset of uniformly randomly sampled documents from a second database (independent of the target database). To simulate different noise levels, we vary the number of such sampled documents – fewer documents means more noise. The level of noise determines how "close" the auxiliary co-occurrence matrix M is to the observed matrix $\bar{M}$.

The goal of our attack is to find the assignment P between the queries and the keywords with the maximum statistical likelihood, given the observed leakage $\bar{M}$ and auxiliary co-occurrence M as prior information. In other words, we aim to maximize the likelihood function $\mathbf{L}\left[P \mid \bar{M}, M\right]$. As we will show, the function $\mathbf{L}\left[P \mid \bar{M}, M\right]$ can be derived from the leakage distributions obtained above. We use simulated annealing [AeOJIP$^+$12] for the maximization step.

Simulated Annealing. At a high level, simulated annealing is a probabilistic technique for searching for the global optimum of a given function. It is very similar to a greedy search algorithm – randomize the input of the function, in our case, that is the assignment P, recompute the score, and if the score is larger than before, the assignment is kept as the new solution, and it is discarded otherwise – except that a worse solution is accepted in simulated annealing if it is not *too bad*. This is to prevent the algorithm from getting stuck in a local optima. More concretely, simulated annealing uses a *temperature* T which decreases per iteration and the differences between the current score of the target function and the previous best score maintained by the algorithm to compute an acceptance probability p, and with probability p the new solution is accepted. This probability is 1 if the new score is higher than the previous best, and less than 1 otherwise. For the same difference in the scores, a lower temperature T leads to a lower acceptance probability, which means simulated annealing behaves as a greedy search algorithm progressively.

Formally, simulated annealing consists of five subroutines, namely a function `InitPerm` to generate an initial assignment, a cooling scheme `Cooling`, a neighbourhood generation algorithm `Neighbour`, a function `Score` to compute the score and a function `AccptProb` to compute the acceptance probability. The syntax of the subroutines are defined below:

- `InitPerm`: takes as input an observed co-occurrence matrix $\bar{M}$ and an auxiliary co-occurrence matrix M, and outputs an assignment P.
- `Cooling`: takes as input a temperature T and the current iteration number i and outputs a new temperature T'.
- `Neighbour`: takes as input an assignment P, an observed co-occurrence matrix $\bar{M}$ and an auxiliary co-occurrence matrix M, and outputs a new assignment P'.
- `Score`: takes as input an observed co-occurrence matrix $\bar{M}$, a auxiliary co-occurrence

Algorithm 1 Simulated Annealing

```
1: procedure $\textsc{Attack}(\bar{M}, M, T_0, i_{\max})$
2:     $P \leftarrow \texttt{InitPerm}(\bar{M}, M)$
3:     $T \leftarrow T_0$
4:     $s \leftarrow \texttt{Score}(\bar{M}, M, P)$
5:     for $i \leftarrow 1, \ldots, i_{\max}$ do
6:         $T \leftarrow \texttt{Cooling}(T, i)$
7:         $P' \leftarrow \texttt{Neighbour}(P, \bar{M}, M)$
8:         $s' \leftarrow \texttt{Score}(\bar{M}, M, P')$
9:         if $\texttt{AccptProb}(T, s, s') > \texttt{rand}(0, 1)$ then
10:            $P \leftarrow P'$
11:            $s \leftarrow s'$
12:     return $P$
```

matrix M and an assignment P, and outputs a score.

- `AccptProb`: takes as input a temperature T, a previous best score s and the new score s', and outputs a probability.

The algorithm begins with an initial temperature T_0 and a random assignment P. An initial score s is computed on this assignment P. Then, the algorithm computes a new temperature $T \leftarrow \texttt{Cooling}(T_0, 1)$, find a new assignment P' using the neighbourhood function $\texttt{Neighbour}(\cdot)$, and compute a new score s' with the score function $\texttt{Score}(\cdot)$. An acceptance probability is computed as $p \leftarrow \texttt{AccptProb}(T, s, s')$. A random number between 0 and 1 is generated and if the random number is less or equal to p, the new solution s' is accepted by the algorithm and kept as the new optimum solution. This process is repeated until the maximum number of iteration is reached. See Algorithm 1 for the formal description.

Our Query Reconstruction Attacks. Our proposed query reconstruction attacks on all of the attack targets described earlier use the aforementioned simulated annealing procedure for maximum likelihood estimation. In all our attacks, we use $T' \leftarrow 0.995T$ as our cooling scheme $\texttt{Cooling}(\cdot)$ and $p \leftarrow \exp(-\frac{s-s'}{T})$ as our function $\texttt{AccptProb}(\cdot)$ to compute the acceptance probability. The precise score function $\texttt{Score}(\cdot)$ that we aim to maximise is the likelihood function $\mathbf{L}\left[P \mid \bar{M}, M\right]$. We defer the formal description of how this function is derived for each target scheme to Appendix B.

We next describe our choices of the initial assignment finding subroutine $\texttt{InitPerm}(\cdot)$ and neighbourhood generation subroutine $\texttt{Neighbour}(\cdot)$ which have a significant impact on the performance and effectiveness of our attacks. The choices we make are tuned for maximizing the attack recovery rate.

$\texttt{InitPerm}(\cdot)$. An initial assignment finding subroutine $\texttt{InitPerm}(\cdot)$ is an efficient algorithm for guessing keywords/keys of the queries, so as to provide a starting point for the more expensive iterative steps later. For our attacks, only the query response lengths are used to avoid expensive computations. We observe that although the observed query response lengths are different from the true query response lengths for all of the schemes we target, these two are related. In particular, for **DP-EMM** [PPYY19] and **DPAP-SE** [CLRZ18], we can compute the expected observed query response lengths from the query response lengths in the auxiliary co-occurrence matrix, and match the queries to the keywords in the auxiliary co-occurrence matrix as well as we can. For **PRT-EMM** [KM19] and **FP-EMM** [PPYY19], the observed keyword frequencies are independent from the true keyword frequencies, .

`Neighbour`$(\cdot)$**.** A neighbourhood generation subroutine generates new assignments for the attacks. Although a uniformly randomly picked assignment works all the time, it may not be the most efficient choice. In particular, for **DP-EMM** [PPYY19] and **DPAP-EMM** [CLRZ18], we know that if an observed query response length is too far from the expectation, the assignment is very unlikely, and can be safely discarded. This means the neighbourhood generation subroutines for the attacks on these two schemes can make use of this, and output a new assignment only if it is sound.

We note that these neighbourhood generation subroutines may prevent some correct assignments in the output of the attacks if their observed query response lengths are too far from the expected query response lengths. By relaxing the bounds, we can make the chance of that happening arbitrarily small. However, the algorithm would then be less efficient as more iterations are required for a convergence. Hence, we see our choice of bounds as a trade-off between query recovery rate and attack efficiency.

For **PRT-EMM** [KM19] and **FP-EMM** [PPYY19], we have to use uniformly randomly picked assignments, since the observed query response lengths are independent from the true query response lengths.

Detailed pseudocodes for the neighbourhood generation subroutines can be found in Appendix C.

4.3 From Query Reconstruction to Database Reconstruction

It turns out that for some of the targeted schemes, our query reconstruction attacks can be extended to *database reconstruction* attacks (by which we mean recovering the keywords associated with the encrypted documents).

DP-EMM. Our query reconstruction attack on the end-to-end SSE system built naturally from **DP-EMM** [PPYY19] implies a database reconstruction attack for practical security parameters; we rely on the noisy access pattern leakage during document retrieval to recovers the *actual* keywords occurring in the encrypted documents retrieved across various queries.

Recall that in **DP-EMM**, the response length to a query on some key is padded to $(2n_{\mathsf{key}} + n^* + \mathbf{Lap}_{sk}(2/\epsilon))$, where n_{key} is the real response length, and n^* is a fixed constant that depends solely on the choice of ϵ. For example, the authors of [PPYY19] suggest using $\epsilon = 0.2$, which yields $n^* = 567$. Our attack stems from the observation that the leakage perturbation is rather small for more frequent keywords, since the corresponding response length n_{key} is much larger than n^*. Hence, for the choice of parameters suggested in [PPYY19], our query reconstruction attack also allows us to progressively recover the set of keywords associated with a given document as more and more queries are observed, leading to a database reconstruction attack. The reconstruction rate is somewhat close to 50% due to the padding-factor of $2\times$ in **DP-EMM**, which causes half of the recovered keywords to be "fake". While this attack can be prevented by altering the parameter ϵ for increased leakage perturbation, such alteration also significantly degrades the practical performance of the scheme. In other words, all practically efficient realizations of the natural end-to-end SSE from **DP-EMM** are broken by our database reconstruction attack.

Our attack does not immediately extend to the end-to-end SSE systems built naturally from **PRT-EMM** [KM19] and **FP-EMM** [PPYY19] due to the inherently different and somewhat larger perturbation to the access pattern leakage in these schemes, and we leave it as an open question to investigate database reconstruction attacks on such schemes.

DPAP-SE. For implementations of **DPAP-SE** [CLRZ18] with reasonably practical parameters (e.g. $p = 0.89999$ and $q = 6.997 \times 10^{-6}$, where p and q are the parameters described earlier), our query reconstruction attack allows guessing the real keywords in the shards (obtained as part of the document retrieval process) with high probability (more concretely, probability p; recall that p needs to be large for the scheme to satisfy correctness in practice). As in the case of **DP-EMM**-based end-to-end SSE, we rely on the noisy access pattern leakage for this database reconstruction attack. Again, while this attack can be prevented by altering the parameters p and q, such alteration also significantly degrades the practical performance of the scheme. Hence, all practically efficient realizations of **DPAP-SE** are broken by our database reconstruction attack.

5 Experimental Evaluation

5.1 Overview of Experimental Setup

Experimental Data and Auxiliary Information. We use the Enron email corpus [WWC] as the target dataset for all of our attacks. A description of the dataset and our preprocessing step can be found in Appendix D. A major challenge for inference-style leakage-abuse attacks is deciding an appropriate model for evaluating their effectiveness in practice. Such a model should take into account both the distribution of queries as well as the distribution of the auxiliary information available to the adversary. Unfortunately, there do not exist concrete guidelines in the literature for how to construct such models; given this lack of precedence, we make certain assumptions that we believe are reasonable in practice.

Query Distribution. We use uniformly distributed keyword queries to evaluate our attacks. This is exactly as in previous attacks [IKK12, CGPR15, BKM20]. We note here that our attacks do not explicitly depend on the distribution of queries; hence a uniform distribution appears to be a reasonable choice.

Auxiliary Data Distribution. For the IKK attack, Islam *et al.* [IKK12] proposed a method to model auxiliary information in an inference-style attack setting; their suggestion was to use an auxiliary co-occurrence pattern leakage obtained by adding Gaussian noise to the original co-occurrence pattern. However, this implicitly assumes a homogeneous distribution of keywords amongst the documents, which may not always be the case in practice. Instead, we opt to split the overall dataset into two halves: out of the 480,000 documents in the dataset, half of the documents are used as the attack target and a fraction of the other half of the documents are used to generate auxiliary information about the dataset. Measurements of 'noisiness' of our auxiliary data can be found in Appendix E. In total, we generate 10 different splits of the documents. For each split, we run 10 independent attacks with freshly generated observed co-occurrence matrices. We measure the fraction of correctly guessed keywords and report the average over the 100 runs as the query recovery rate.

Keyword Extraction. We extract keywords using the Natural language toolkit [Pro] in Python. Keywords with frequency higher than 5% are removed. We do not apply stemming to the keywords as that does not affect query recovery rate in our experiments.

Keyword and Query Selection. We use the 1,000, 2,000, 3,000 and 4,000 most frequent keywords to build auxiliary co-occurrence matrices, and sample uniformly randomly without replacement from these most frequent keywords subsets of 250, 500, 750 keywords as queried keywords. These queried keywords are used to build observed co-occurrence matrices. These observed and auxiliary co-occurrence matrices are then used as the inputs to our attacks.

Security Parameter Selection for the Target Constructions. We use the security parameters suggested in the original papers to run our attacks. We also investigate how changes in the security parameters affect query reconstruction rates.

PRT-EMM Recall that **PRT-EMM** from [KM19] allows the client to pick a public parameter λ which controls the padded query response lengths as: $n'_{\mathsf{key}} = \lambda + F_{sk}(\mathsf{key}||n_{\mathsf{key}})$. The authors suggested to set λ between 0 and $0.25n_{max}$. We used $\lambda = 0$ and $0.25n_{max}$ in our experiments. In addition, we used $\lambda = 0.5n_{max}$ to see the effect of additional padding on query reconstruction rate.

FP-EMM and DP-EMM **FP-EMM** from [PPYY19] does not have any tunable parameters and so we run our attacks on the **FP-EMM** as it is. **DP-EMM** from [PPYY19] uses parameter ϵ to set query response volumes to $(2n_{\mathsf{key}} + n^* + \mathbf{Lap}_{sk}(2/\epsilon))$. The authors suggested $\epsilon = 0.2$ and we use the same value. We also run experiments where ϵ is significantly smaller, ranging from 0.1 to 0.01.

DPAP-SE For **DPAP-SE** [CLRZ18], the authors suggested $m = 6$ (the number of shards per document), $k = 2$ (a parameter of the erasure code which does not affect our attack), $p = 0.88703$ (the probability for which a keyword is kept in a shard) and $q = 0.04416$ as the parameters to use for the Enron [WWC] dataset. We used similar parameters ($m = 6, k = 2, p = 0.89$ and $q = 0.045$) in our experiments. A smaller p or a bigger q significantly reduces the efficiency of the construction so we opt to not run additional experiments with those parameters. Instead, we investigate how a smaller q affects query reconstruction rate. We use $q = 0.0045, 0.00045$ and 0.000045 as additional choices of parameters.

Implementation. We implemented our attacks in C using GNU Scientific Library [Gou09] for randomness generation and probability calculations. We used our own custom code for simulated annealing for best performance. We parallelized our implementation using OpenMP [Ope18]. Our implementation is highly scalable. It takes less than one minute per run on the differentially-private schemes (**DPAP-SE** and **DP-EMM**) and no more than 6 minutes per run on the other schemes (**PRT-EMM** and **FP-EMM**) for all of our experimental settings, on a machine with an 8-core (16-thread) Sandy Bridge CPU clocked at 2.6 GHz.

Experiments. We present three sets of experimental results on the target constructions discussed above. In Section 5.2, we present the experimental results in basic settings, where the auxiliary co-occurrence matrix is built from 48,000 documents (20% of the available auxiliary data) using the 1,000 most frequent keywords. We set the number of queried keywords to $250, 500$ or 750, and the security parameters are allowed to vary. In Section 5.3, we set the number of queried keywords to $250, 500$ or 750, and the security parameters to those suggested in the original papers, and vary the number of keywords used to build auxiliary information between 1,000 and 4,000. Just as before, 48,000 documents are used in building auxiliary information. Finally, in Section 5.4, we use anywhere from 12,000 to 96,000 documents to build the auxiliary co-occurrence matrix, as a means to simulate auxiliary information with different levels of noise. The number of keywords used is set to 1,000, and the number of queries is allowed to vary from 250 to 750. The security parameters are set to the ones recommended in the original papers.

5.2 Varying the Security Parameters of the Constructions

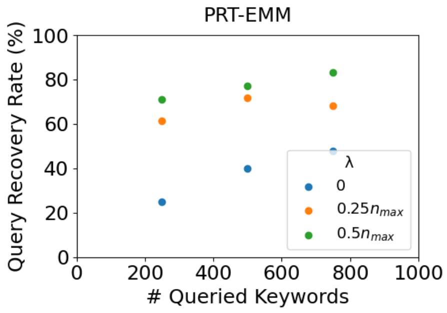
PRT-EMM. The experimental results on **PRT-EMM** are shown in Figure 2a. We observe an increasing query recovery rate with more queried keywords and larger λ. The attack performs significantly worse when $\lambda = 0$.

The worse performance of our attack on small λ is caused by truncations of query response volumes (see Section 4.1) which lead to loss of co-occurrence information. It should be noted that although a smaller λ leads to better query privacy, it results in more truncations and less complete query responses.

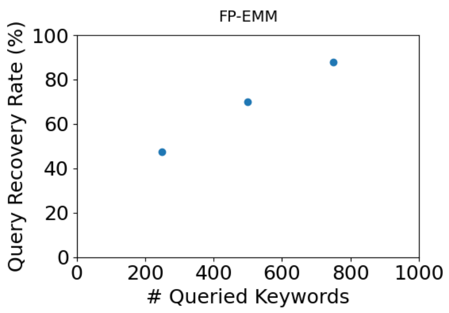
FP-EMM. The experimental results of our attack on **FP-EMM** are shown in Figure 2b. As expected, the attack performs better with more queried keywords. The attack is able to recover over 70% of the queried keywords if over 500 keywords have been queried, suggesting that full padding is ineffective at adding noise to the co-occurrence leakage.

DP-EMM. The experimental results on **DP-EMM** are shown in 2c. The attack performs slightly better with a smaller ϵ, suggesting that our attack is over-fitting the auxiliary data.

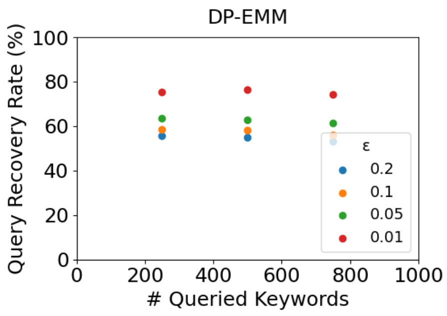
DPAP-SE. The results of our attacks are shown in Figure 2d. The attack is able recover between 50% and 80% of the queries in all cases we have considered.



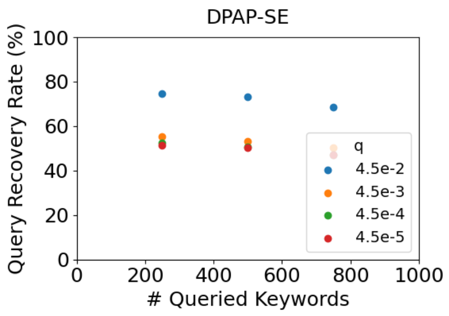
(a) **PRT-EMM** [KM19]. n_{max} is the maximum query response length.



(b) **FP-EMM** [PPYY19].



(c) **DP-EMM** [PPYY19].



(d) **DPAP-SE** [CLRZ18].

Figure 2: Experimental results with varying security parameters. The 1,000 most frequent keywords are used in the auxiliary information.

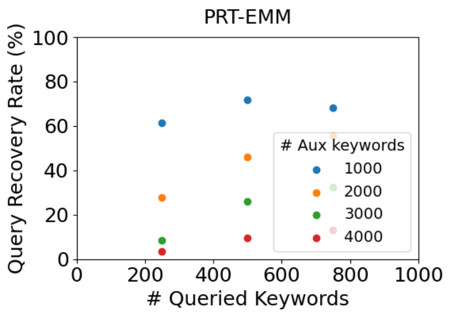
As opposed to what one might expect, the attack performs better with a larger q. It is certainly true that a larger q masks the true co-occurrences better as there are more fake keywords introduced to each document, but it also reduces the effectiveness of perturbation with respect to keyword frequencies.

For simplicity, consider a database with N documents, of which n documents contain keyword kw. With parameters m, p and q, the frequency of keyword kw in the resultant database processed by **DPAP-SE** is $k(np + (N - n)q)$. A larger q increases the separation of frequencies for different keywords, and we believe that is the main reason why our attack performs better with larger q.

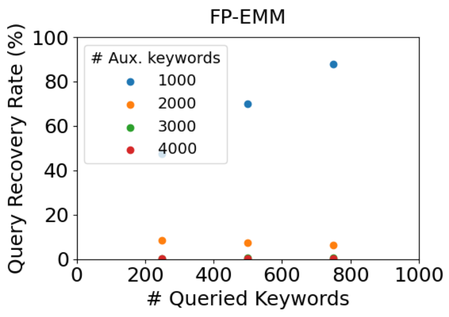
5.3 Varying the Number of Keywords in Auxiliary Information

Our experimental results on varying the number of keywords in the auxiliary information are shown in Figure 3. The security parameters we used can be found in the captions.

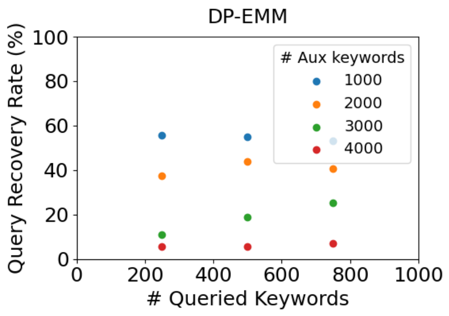
As the keywords are uniformly randomly picked, the attacks with more auxiliary keywords are necessarily less successful. There are two main reasons for this. Firstly, since we have fixed the number of queried keywords, the search space for the attacks with more auxiliary



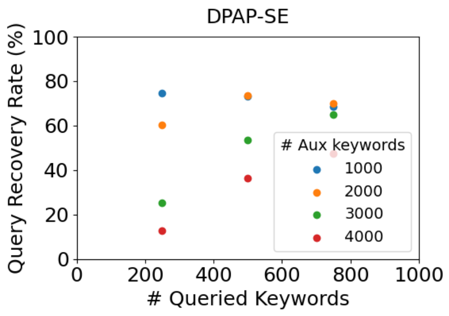
(a) **PRT-EMM** [KM19]. λ is set to $0.25n_{max}$.



(b) **FP-EMM** [PPYY19].



(c) **DP-EMM** [PPYY19]. $\epsilon = 0.2$.



(d) **DPAP-SE** [CLRZ18]. $(m, k, p, q) = (6, 2, 0.89, 0.045)$.

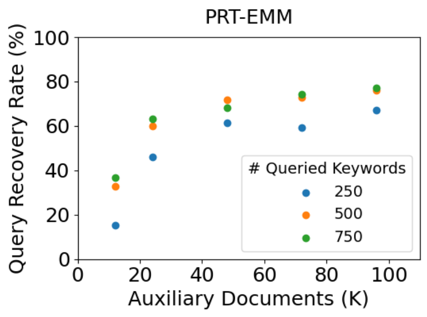
Figure 3: Experimental results with varying number of keywords in auxiliary information.

keywords are larger, and there is naturally more uncertainty associated to those attacks. Secondly, as we have picked the queried keywords uniformly, it is more likely for the attacks with more auxiliary keywords to hit low frequency keywords, which naturally contain less co-occurrence information to begin with. Nevertheless, the constructions behave very differently with respect to the number of keywords in the auxiliary information.

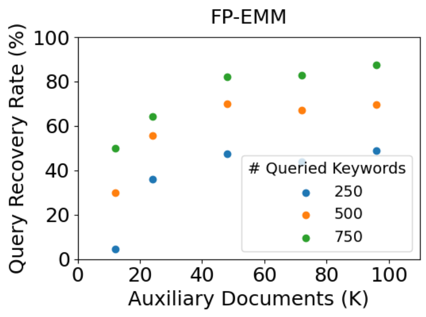
Our attacks on **PRT-EMM** and **DP-EMM** work reasonably well for up to 2,000 auxiliary keywords. More queries are required for the attack to succeed with 3,000 and 4,000 auxiliary keywords.

Our attack is less successful on **FP-EMM** when the number of auxiliary keywords is large. This shows that full padding is effective at masking co-occurrence information if there is enough uncertainty within the queries.

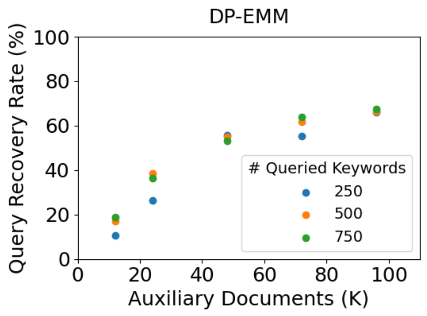
For **DPAP-SE**, the attacks with more auxiliary keywords do not perform well when only 250 keywords are queried. However, the query recovery rate increases significantly as more keywords are queried. The attack is able to recover over 50% of the queries even if only 750 out of 4,000 keywords have been queried.



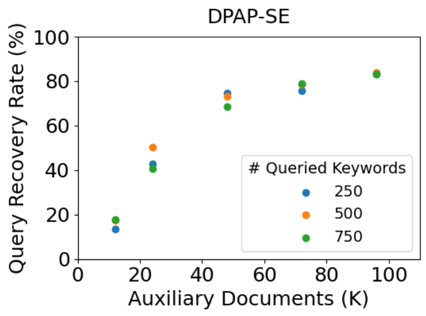
(a) **PRT-EMM** [KM19]. λ is set to $0.25n_{max}$.



(b) **FP-EMM** [PPYY19].



(c) **DP-EMM** [PPYY19]. ϵ is set to 0.2.



(d) **DPAP-SE** [CLRZ18]. $(m, k, p, q) = (6, 2, 0.89, 0.045)$.

Figure 4: Experimental results with varying auxiliary information.

5.4 Varying the Level of Noise in Auxiliary Information

Given that there is no widely accepted way of modelling noise in auxiliary information, we opt to use different numbers of documents in auxiliary information as a way to simulate different levels of noise – fewer documents means more noise. We use absolute distance and modified probability score to measure the level of noise introduced in each set of experiments we run (see Appendix E for details).

Our experimental results on varying auxiliary information are shown in Figure 4. The security parameters we used can be found in the captions. The attacks do not perform well when only 12,000 documents are used to construct the auxiliary information. However, the query reconstruction rate increases as the number of auxiliary documents increases. Interestingly, the query reconstruction rates using 48,000 and 96,000 auxiliary documents are comparable, suggesting that using 48,000 documents (10% of the total in the Enron dataset) is sufficient as auxiliary information and that our attacks are robust in a noisy setting. This aligns with the observation that the level of noise measured in Figure 7 of Appendix E stabilises at 48,000 documents.

6 Document Retrieval with EMMs and Other Primitives

In this section, we explore the alternative possibility of building end-to-end SSE systems where leakage-suppression techniques (such as volume-hiding EMMs) are applied to the whole system (i.e. to *both* the encrypted index *and* the encrypted document collection). In such a system, the document retrieval step is only allowed to leak 'trivial' information such as the number of documents and the number of keyword-document pairs, but not potentially sensitive information such as the access pattern. We will show that, with currently available techniques, no end-to-end SSE system using volume-hiding EMMs (or similar leakage-suppression techniques) can achieve this without incurring significant storage, computation and/or bandwidth overheads.

6.1 Primitives for Index and Document Retrieval

In the rest of this section, we assume the SSE system follows the two-phase approach illustrated in Figure 1. We also assume that the index retrieval phase (where the client recovers the set of (encrypted) document identifiers corresponding to documents matching the query) is implemented using one of the state-of-the-art schemes **PRT-EMM** [KM19] or **FP-EMM** [PPYY19], so the leakage from this phase consists of at most: (a) the number of documents $|\mathbf{DB}|$, (b) search pattern (query equality), and (c) the maximum response length $\max_{\mathrm{kw}} |\mathbf{DB}(\mathrm{kw})|$. This leakage profile is a consequence of the properties of the aforementioned volume-hiding EMMs. We demand that the leakage from the second document retrieval phase should be no more than that in the first phase, in order to achieve effective system-wide leakage-suppression. This naturally leads us to consider a variety of different cryptographic primitives for realizing the document retrieval step, namely EMMs (again), Private Information Retrieval, and Oblivious RAM.

Using EMMs for Document Retrieval. The state-of-the-art EMMs [KM19, PPYY19] can be easily transformed into document retrieval schemes by replacing the values stored in the EMMs with the encrypted documents themselves. We then obtain the same leakage from document retrieval as from the underlying EMM. We note that the resulting scheme does not actually need an index retrieval phase as the entire process can be carried out just using the document retrieval phase.

Using Private Information Retrieval. A Private Information Retrieval (PIR) scheme can be used to retrieve the documents. Here, we focus on computational PIR schemes [ABFK16, ACLS18] for which there is provably no leakage (per retrieval). As we are only permitted to leak the maximum response length in the document retrieval phase (in order to match the leakage profile of the first phase), we need to call the underlying PIR scheme $\max_{\mathrm{kw}} |\mathbf{DB}(\mathrm{kw})|$ times per query (padding with dummy retrievals if necessary) in order to hide the true response length. We use the single message version of SealPIR [ACLS18] in our evaluation below. This is a state-of-the-art PIR scheme using Fully Homomorphic Encryption (FHE) as a subcomponent.

Scheme	Storage (Server)	Query (Client)		Query (Server)	
		Computation	Communication	Computation	Communication
Naïve*	470 MB	f prf f dec	32 B	f acc	f KB
Duplication	17 GB (36x)	24K prf 24K dec	750 KB (46x)	24K acc	23 MB (46x)
PRT-EMM [KM19]	390 GB (860x)	12K prf 12K dec	370 KB (23x)	12K acc	12 MB (23x)
FP-EMM [PPYY19]	43 GB (94x)	48K prf 48K dec	1.5 MB (92x)	48K acc	47 MB (92x)
SealPIR [ACLS18]	7.3 GB (16x)	1 henc 1 hdec	370 MB (24,000x)	23B hmul 11B hsub 11B hadd	370 MB (740x)
Non-recursive Path ORAM [SvS+13]	1.8 GB (4x)	3.4M acc 1.7M dec 1.7M enc	1.65 GB (110,000x)	3.4M acc	1.7 GB (3,300x)

Table 2: Evaluation of different document retrieval primitives with minimal leakage. The numbers in the brackets indicate overheads beyond the baseline provided by the **Naïve** scheme. We assume 522 documents (mean keyword frequency) are retrieved by the **Naïve** scheme in the computations of the overheads. *: f is the real query response volume (since there is no padding).

Using Oblivious RAM. Oblivious RAM (ORAM) [Gol87, DMN11, SvS+13, CNS18] is another primitive that achieves zero-leakage per access. We use it to protect the entire document collection. Similarly to PIR, the number of data accesses for ORAM has to be padded to the maximum response length in order to hide true response lengths. We use the non-recursive version of Path ORAM [SvS+13] in our evaluation below. This specific choice is amongst the most efficient ORAM schemes available.

6.2 Performance Evaluation

Additional Schemes. On top of the four primitives mentioned above, we also add the following two schemes in our comparison as a baseline:

- **Naïve**: The scheme simply encrypts the documents and stores them in an array. To retrieve documents, the user sends the array locations (used as document identifiers) to the server and the server returns all documents in those locations. This scheme is insecure in a system-wide attack setting, e.g. it is vulnerable to our inference attack.
- **Duplication**: The scheme is identical to **Naïve** except that the encrypted documents are duplicated for each keyword in the same way as the encrypted document identifiers are duplicated in the search index of a traditional structure-only SSE scheme, e.g. [CGKO06, CK10, CJJ+13]. This represents a baseline method for using an EMM to build an end-to-end SSE scheme.

Experimental Data. A concrete dataset is necessary for the evaluation as the duplication techniques used in **Duplication**, **PRT-EMM** and **FP-EMM** are data-dependent. We again pick the Enron email corpus [WWC], details in Appendix D. We used 480K documents in the evaluation below. To simplify the evaluation, we assume that all documents have size 1 KB even if some of them are larger than that in reality. If we were to use document splitting into fixed-size chunks instead, this would result in a 10× larger storage overhead because of increased duplication of the chunks compared to the original files.

Parameters. The following parameters are used in our evaluation. We use PRFs with 256-bit output. We use the most space-efficient parameters for **PRT-EMM** proposed in the original papers [KM19], namely $\alpha = 0.5$. For SealPIR, we assume that the FHE ciphertexts are 16 KB in size each, as per the original paper [ACLS18]. For Path ORAM [SvS$^+$13], we assume each block has size 1 KB and there are 4 blocks per bucket.

Evaluation. We report communication volume, storage cost, and the number of core cryptographic operations needed for each option described above in Table 2. We split computation and communication costs into client and server costs, and report only server storage costs (client storage costs are low).

Storage and communication costs are measured in total volumes. Additional overheads arising from how the data is structured and packaged for communication are ignored.

Computation costs are measured by the number of core cryptographic operations. The operations that we consider include: prf for PRF computation; enc and dec for encryption and decryption with a symmetric primitive; acc for disk/RAM access (read or write); henc and hdec for encryption and decryption with FHE; hmul, hsub, hadd for multiplication, substitution and addition for FHE ciphertexts. Reporting operation counts in place of running times makes our comparison independent of implementation details.

We opt to not include latency as this depends on several factors such as data access speed and network delay, and these are hard to compare concretely and fairly.

Discussion. It is clear that all of the options suffer significant storage overheads. For **Duplication**, **PRT-EMM** and **FP-EMM**, this is caused by duplication. The expansion factor grows linearly with the number of keywords per document. For PIR, the expansion factor comes from the use of homomorphic encryption. It is not clear how the ciphertexts can be compressed to reduce the overhead. It is conceivable that alternative PIR schemes might avoid such expansion. For ORAM, the overhead comes from the use of multiple blocks per bucket. This is necessary to prevent overflowing buckets, meaning the storage overhead cannot be reduced significantly.

With regard to queries, **Duplication**, **PRT-EMM** and **FP-EMM** have reasonable computational costs, but the communication costs from the server to the client are high in each case. The server needs to send 2.5% to 10% of the entire document collection to the client per query, which is a lot more than the average keyword frequency might suggest (0.109% for the Enron corpus). The PIR and ORAM options naturally suffer from high computation and/or communication overheads since they are not designed for large-scale document retrieval.

We note that the primitives we have chosen in making our comparison of options are not necessarily optimal. Future work on these primitives should result in more efficient options. Nevertheless, the concrete numbers we provide are indicative of what is possible by employing state-of-the-art approaches.

7 Discussion

On System-Wide leakage. Our attacks on end-to-end SSE schemes built from volume-hiding EMMs in the natural way highlight the threats posed by system-wide leakage and the need to revisit existing security definitions that ignore such leakage. Note that many SSE constructions were designed prior to the proposal of volume-hiding EMMs; in their original form, these constructions effectively used EMMs that themselves leak the exact access-pattern. In this case, the leakage from the encrypted search index subsumes the leakage from encrypted document retrieval and so the latter leakage can be ignored in security analysis (see [CGKO06, CJJ+13] for relevant discussions). However, this approach to analysis is no longer valid when such access pattern-revealing EMMs are replaced by volume-hiding EMMs. In this case, the leakage arising from encrypted document retrieval is no longer covered by security definitions that focus purely on the encrypted search index.

Of course, one could try to use state-of-the art techniques like volume-hiding EMMs to protect both the encrypted search index and the encrypted document store, thereby reducing the system-wide leakage to an acceptable level. We have shown that the obvious way of doing this incurs enormous overheads.

On Differentially Private Access Patterns. Our attack applied to **DPAP-SE** [CLRZ18] does not negate the core idea of [CLRZ18], which is to design end-to-end SSE schemes that take into account leakage from the document retrieval phase. Rather, our attack serves as a warning about the potential pitfalls of applying differential privacy to SSE without appropriately modeling and analyzing the resulting leakage. As pointed out in [CLRZ18], differentially private access patterns provide provable guarantees of the form: an adversary cannot distinguish between queries over keywords such that their access pattern leakage is within a *small* statistical distance of each other. As demonstrated by our attack and our experiments, the provable guarantees provided by differential privacy do not necessarily translate into security guarantees against leakage-abuse attacks *in general.*

On the Practicality of Our Attack. Our attack is an inference attack in that it assumes the adversary has access to auxiliary data that is independent of but statistically "close" to the target database. We believe that this a weaker (and more realistic) attack setting compared to existing known-data attacks [IKK12, CGPR15, BKM20]. Our attack achieves high keyword recovery rates even when the target schemes use aggressive security parameters, or when the auxiliary data available to the adversary is relatively noisy (which is the case when we sample the auxiliary data from a small portion of the database). These observations further reinforce the practicality of our attack.

A drawback of our attack is that it assumes auxiliary information involving high-frequency keywords. This is a relatively strong assumption in practice (although one also made by all previous leakage-abuse attacks). One can of course filter out leakage from low frequency keywords based on response volume before running our attack. We leave it as an open problem to extend the attack to such keywords.

Future Research on SSE. We argue that, if the goal of the research community is to develop SSE towards practice, then a fundamental shift in approach is needed. In particular, researchers need to take a system-wide view of SSE and its security. This requires considering all components of an end-to-end SSE system when doing security analysis, and investigating techniques that reduce leakage and maintain efficiency of the SSE system as a whole. In this context, our view is that the approaches taken in [CLRZ18] and [GPPW20], where end-to-end SSE schemes are designed from scratch and dedicated leakage suppression for both index and document retrieval is included, represent the right way forward.

References

[ABFK16] Carlos Aguilar Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. XPIR: Private information retrieval for everyone. *Proceedings on Privacy Enhancing Technologies*, 2016(2):155–174, April 2016.

[ACLS18] Sebastian Angel, Hao Chen, Kim Laine, and Srinath T. V. Setty. PIR with compressed queries and amortized query processing. In *2018 IEEE Symposium on Security and Privacy*, pages 962–979, San Francisco, CA, USA, May 21–23, 2018. IEEE Computer Society Press.

[AeOJIP$^+$12] Hime Aguiar e Oliveira Junior, Lester Ingber, Antonio Petraglia, Mariane Rembold Petraglia, and Maria Augusta Soares Machado. *Adaptive Simulated Annealing.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[BKM20] Laura Blackstone, Seny Kamara, and Tarik Moataz. Revisiting leakage abuse attacks. In *ISOC Network and Distributed System Security Symposium – NDSS 2020*, San Diego, CA, USA, February 23-26, 2020. The Internet Society.

[BMO17] Raphaël Bost, Brice Minaud, and Olga Ohrimenko. Forward and backward private searchable encryption from constrained cryptographic primitives. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 1465–1482, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press.

[Bos16] Raphael Bost. $\Sigma o\phi o\varsigma$: Forward secure searchable encryption. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 1143–1154, Vienna, Austria, October 24–28, 2016. ACM Press.

[CGKO06] Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006: 13th Conference on Computer and Communications*

Security, pages 79–88, Alexandria, Virginia, USA, October 30 – November 3, 2006. ACM Press.

[CGPR15] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. Leakage-abuse attacks against searchable encryption. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015: 22nd Conference on Computer and Communications Security*, pages 668–679, Denver, CO, USA, October 12–16, 2015. ACM Press.

[CGPR16] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. Leakage-abuse attacks against searchable encryption. Cryptology ePrint Archive, Report 2016/718, 2016. `http://eprint.iacr.org/2016/718`.

[CJJ$^+$13] David Cash, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. Highly-scalable searchable symmetric encryption with support for Boolean queries. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 353–373, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[CJJ$^+$14] David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. Dynamic searchable encryption in very-large databases: Data structures and implementation. In *ISOC Network and Distributed System Security Symposium – NDSS 2014*, San Diego, CA, USA, February 23–26, 2014. The Internet Society.

[CK10] Melissa Chase and Seny Kamara. Structured encryption and controlled disclosure. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 577–594, Singapore, December 5–9, 2010. Springer, Heidelberg, Germany.

[CLRZ18] G. Chen, T. Lai, M. K. Reiter, and Y. Zhang. Differentially private access patterns for searchable symmetric encryption. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 810–818, 2018.

[CNS18] T.-H. Hubert Chan, Kartik Nayak, and Elaine Shi. Perfectly secure oblivious parallel RAM. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part II*, volume 11240 of *Lecture Notes in Computer Science*, pages 636–668, Panaji, India, November 11–14, 2018. Springer, Heidelberg, Germany.

[CPPJ18] Javad Ghareh Chamani, Dimitrios Papadopoulos, Charalampos Papamanthou, and Rasool Jalili. New constructions for forward and backward private symmetric searchable encryption. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 1038–1055, Toronto, ON, Canada, October 15–19, 2018. ACM Press.

[DGWR07] Alexandros G. Dimakis, Brighten Godfrey, Martin J. Wainwright, and Kannan Ramchandran. Network coding for distributed storage systems. *CoRR*, abs/cs/0702015, 2007.

[DHP21] Marc Damie, Florian Hahn, and Andreas Peter. A highly accurate query-recovery attack against searchable encryption using non-indexed documents. In Michael Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 143–160. USENIX Association, 2021.

[DMN11] Ivan Damgård, Sigurd Meldgaard, and Jesper Buus Nielsen. Perfectly secure oblivious RAM without random oracles. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 144–163, Providence, RI, USA, March 28–30, 2011. Springer, Heidelberg, Germany.

[FJK$^+$15] Sky Faber, Stanislaw Jarecki, Hugo Krawczyk, Quan Nguyen, Marcel-Catalin Rosu, and Michael Steiner. Rich queries on encrypted data: Beyond exact matches. In Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl, editors, *ESORICS 2015: 20th European Symposium on Research in Computer Security, Part II*, volume 9327 of *Lecture Notes in Computer Science*, pages 123–145, Vienna, Austria, September 21–25, 2015. Springer, Heidelberg, Germany.

[Gol87] Oded Goldreich. Towards a theory of software protection and simulation by oblivious RAMs. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 182–194, New York City, NY, USA, May 25–27, 1987. ACM Press.

[Gou09] Brian Gough. *GNU Scientific Library Reference Manual - Third Edition.* Network Theory Ltd., 3rd edition, 2009.

[GPPW20] Zichen Gui, Kenneth G. Paterson, Sikhar Patranabis, and Bogdan Warinschi. SWiSSSE: System-wide security for searchable symmetric encryption. *IACR Cryptol. ePrint Arch.*, 2020:1328, 2020.

[IKK12] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *ISOC Network and Distributed System Security Symposium – NDSS 2012*, San Diego, CA, USA, February 5–8, 2012. The Internet Society.

[KM17] Seny Kamara and Tarik Moataz. Boolean searchable symmetric encryption with worst-case sub-linear complexity. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part III*, volume 10212 of *Lecture Notes in Computer Science*, pages 94–124, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany.

[KM18] Seny Kamara and Tarik Moataz. SQL on structurally-encrypted databases. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part I*, volume 11272 of *Lecture Notes in Computer Science*, pages 149–180, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Heidelberg, Germany.

[KM19] Seny Kamara and Tarik Moataz. Computationally volume-hiding structured encryption. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part II*, volume 11477 of *Lecture Notes in*

Computer Science, pages 183–213, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany.

[KPR12] Seny Kamara, Charalampos Papamanthou, and Tom Roeder. Dynamic searchable symmetric encryption. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 2012: 19th Conference on Computer and Communications Security*, pages 965–976, Raleigh, NC, USA, October 16–18, 2012. ACM Press.

[NHP$^+$21] Jianting Ning, Xinyi Huang, Geong Sen Poh, Jiaming Yuan, Yingjiu Li, Jian Weng, and Robert H. Deng. Leap: Leakage-abuse attack on efficiently deployable, efficiently searchable encryption with partially known dataset. CCS '21, page 2307–2320, New York, NY, USA, 2021. Association for Computing Machinery.

[NPG14] Muhammad Naveed, Manoj Prabhakaran, and Carl A. Gunter. Dynamic searchable encryption via blind storage. In *2014 IEEE Symposium on Security and Privacy*, pages 639–654, Berkeley, CA, USA, May 18–21, 2014. IEEE Computer Society Press.

[OK21a] Simon Oya and Florian Kerschbaum. Hiding the access pattern is not enough: Exploiting search pattern leakage in searchable encryption, 2021.

[OK21b] Simon Oya and Florian Kerschbaum. Hiding the access pattern is not enough: Exploiting search pattern leakage in searchable encryption. In *USENIX Security 2021*, pages 127–142. USENIX Association, 2021.

[OK21c] Simon Oya and Florian Kerschbaum. Ihop: Improved statistical query recovery against searchable symmetric encryption through quadratic optimization, 2021.

[Ope18] OpenMP Architecture Review Board. OpenMP application program interface version 5.0, 2018.

[PPYY19] Sarvar Patel, Giuseppe Persiano, Kevin Yeo, and Moti Yung. Mitigating leakage in secure cloud-hosted data structures: Volume-hiding for multi-maps via hashing. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 79–93. ACM Press, November 11–15, 2019.

[Pro] NLTK Project. *Natural Language Toolkit.* `https://www.nltk.org/`.

[PW16] David Pouliot and Charles V. Wright. The shadow nemesis: Inference attacks on efficiently deployable, efficiently searchable encryption. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 1341–1352, Vienna, Austria, October 24–28, 2016. ACM Press.

[Smi48] N. Smirnov. Table for Estimating the Goodness of Fit of Empirical Distributions. *The Annals of Mathematical Statistics*, 19(2):279 – 281, 1948.

[SPS14] Emil Stefanov, Charalampos Papamanthou, and Elaine Shi. Practical dynamic searchable encryption with small leakage. In *ISOC Network and Distributed System Security Symposium – NDSS 2014*, San Diego, CA, USA, February 23–26, 2014. The Internet Society.

[SvS+13] Emil Stefanov, Marten van Dijk, Elaine Shi, Christopher W. Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path ORAM: an extremely simple oblivious RAM protocol. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013: 20th Conference on Computer and Communications Security*, pages 299–310, Berlin, Germany, November 4–8, 2013. ACM Press.

[SWP00] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *2000 IEEE Symposium on Security and Privacy*, pages 44–55, Oakland, CA, USA, May 2000. IEEE Computer Society Press.

[WWC] CMU William W. Cohen, MLD. Enron email dataset.

A Mathematical Derivations of the Co-occurrence Matrices

Derivation for PRT-EMM [KM19]. Recall that in **PRT-EMM**, the query response lengths are padded or truncated as:

$$n'_{\mathsf{key}} = \lambda + F_{sk}(\mathsf{key}||n_{\mathsf{key}}).$$

Let $\mathbf{DB}$ be a multi-map and $\mathsf{q}_1, \ldots, \mathsf{q}_l$ be non-repeating search queries with associated keys $\mathsf{key}_1, \ldots, \mathsf{key}_l$ on $\mathbf{DB}$ encrypted with **PRT-EMM**. We abuse the notation $\mathsf{key}(\mathsf{q}_i)$ to mean the key associated to q_i. By denoting the maximum value of the PRF F as $|F|$, the diagonal entries of the co-occurrence matrix can be expressed as:

$$\bar{M}(\mathsf{q}_1, \ldots, \mathsf{q}_l; \mathbf{DB})_{i,i} \sim \lambda + \mathbf{Uniform}(0, |F|),$$

where $\mathbf{Uniform}(\cdot)$ is a uniform distribution.

There are three cases to be considered for the off-diagonal entries of the co-occurrence matrix. Without loss of generality, let the keys in concern be keys key_i and key_j. In the first case, both of the query response lengths associated to the keys are larger than the true query response lengths. This corresponds to $n'_{\mathsf{key}_i} - |\mathbf{DB}(\mathsf{key}_i)|$ random document retrievals for queries on key key_i and $n'_{\mathsf{key}_j} - \left|\mathbf{DB}(\mathsf{key}_j)\right|$ random document retrievals for queries on key key_j. These random document retrievals can create additional co-occurrence counts among themselves or with the real document retrievals. The co-occurrence counts in this case can be approximated by:

$$\begin{aligned}\bar{M}(\mathsf{q}_1,\ldots,\mathsf{q}_l;\mathbf{DB})_{i,j} \sim& \left|\mathbf{DB}(\mathsf{key}_i,\mathsf{key}_j)\right| \\ &+\mathbf{HGeom}\left(n'_{\mathsf{key}_i} - \left|\mathbf{DB}(\mathsf{key}_i)\right|, \left|\mathbf{DB}\right|, n'_{\mathsf{key}_j}\right) \\ &+\mathbf{HGeom}\left(n'_{\mathsf{key}_j} - \left|\mathbf{DB}(\mathsf{key}_j)\right|, \left|\mathbf{DB}\right|, n'_{\mathsf{key}_i}\right),\end{aligned}$$

where $\mathbf{HGeom}(n, N, K)$ denotes a hypergeometric distribution which makes n draws without replacement, from a population of size N that contains exactly K objects with the desired feature.

In the second case, one of the query response lengths is truncated and the other one is padded. Without loss of generality, let key key_i be the truncated key and key key_j be the padded key. Then, the co-occurrence count associated to keys key_i and key_j can be modelled as a process where the co-occurrence count is first reduced by the truncation and then increased by the padding. Its distribution is given below:

$$x \sim \mathbf{HGeom}\Big(n'_{\mathsf{key}_i}, \left|\mathbf{DB}(\mathsf{key}_i)\right|, \left|\mathbf{DB}(\mathsf{key}_i,\mathsf{key}_j)\right|\Big),$$

$$\bar{M}(\mathsf{q}_1,\ldots,\mathsf{q}_l;\mathbf{DB})_{i,j} \sim x + \mathbf{HGeom}\Big(n'_{\mathsf{key}_j} - \left|\mathbf{DB}(\mathsf{key}_j)\right|, \left|\mathbf{DB}\right|, n'_{\mathsf{key}_i} - x\Big).$$

Finally, in the last case, both of the query response lengths are truncated. Similar to above, the distribution of the co-occurrence count associated to keys key_i and key_j can be expressed as:

$$x \sim \mathbf{HGeom}\Big(n'_{\mathsf{key}_i}, \left|\mathbf{DB}(\mathsf{key}_i)\right|, \left|\mathbf{DB}(\mathsf{key}_i,\mathsf{key}_j)\right|\Big),$$

$$\bar{M}(\mathsf{q}_1,\ldots,\mathsf{q}_l;\mathbf{DB})_{i,j} \sim \mathbf{HGeom}\Big(n'_{\mathsf{key}_j}, \left|\mathbf{DB}(\mathsf{key}_j)\right|, x\Big).$$

Derivation for new volume-hiding multi-maps in [PPYY19]. The volume-hiding multi-maps in [PPYY19] are special cases of **PRT-EMM** [KM19], where the query response lengths are either padded to the maximum query response length or ones that are larger than the true query response lengths. Specifically, for the full padding version (**PRT-EMM**),

$$\bar{M}(\mathsf{q}_1,\ldots,\mathsf{q}_l;\mathbf{DB})_{i,i} \sim 2\max_{\mathsf{key}} \left|\mathbf{DB}(\mathsf{key})\right|.$$

And for the differentially-private version (**DP-EMM**),

$$\bar{M}(\mathsf{q}_1,\ldots,\mathsf{q}_l;\mathbf{DB})_{i,i} \sim 2\left|\mathbf{DB}(\mathsf{key})\right| + n^* + \mathbf{Lap}(2/\epsilon),$$

where n^* is a fixed constant to offset the query response length in case the latter random variable is negative.

For the co-occurrence counts, we get:

$$\begin{aligned}\bar{M}(\mathsf{q}_1,\ldots,\mathsf{q}_l;\mathbf{DB})_{i,j} \sim & \left|\mathbf{DB}(\mathsf{key}_i,\mathsf{key}_j)\right| \\ & +\mathbf{HGeom}\left(n'_{\mathsf{key}_i} - \left|\mathbf{DB}(\mathsf{key}_i)\right|, \left|\mathbf{DB}\right|, n'_{\mathsf{key}_j}\right) \\ & +\mathbf{HGeom}\left(n'_{\mathsf{key}_j} - \left|\mathbf{DB}(\mathsf{key}_j)\right|, \left|\mathbf{DB}\right|, n'_{\mathsf{key}_i}\right),\end{aligned}$$

where n'_{key_i} and n'_{key_j} are the padded query response lengths for keyword kw_i and kw_j respectively.

Derivation for DPAP-SE [CLRZ18]. Let $\mathbf{DB}$ be a database and $\mathsf{q}_1,\ldots,\mathsf{q}_l$ be non-repeating search queries with associated keywords $\mathrm{kw}_1,\ldots,\mathrm{kw}_l$ on $\mathbf{DB}$ encrypted with the searchable encryption scheme above [CLRZ18]. The diagonal entries of the co-occurrence matrix $\bar{M}(\mathsf{q}_1,\ldots,\mathsf{q}_l;\mathbf{DB})$, i.e. the query response volumes, represent the numbers of shards retrieved by the client. For a particular query q_i, the number of shards retrieved is determined by:

- The number of shards which contain keyword kw_i before the pre-processing step, and the keyword is not removed from them.
- The number of shards which do not contain keyword kw_i before the pre-processing step, but the keyword is added to them.

Formally, the diagonal entries of the co-occurrence matrix can be expressed in terms of the true query response lengths as:

$$\bar{M}(\mathsf{q}_1,\ldots,\mathsf{q}_l;\mathbf{DB})_{i,i} \sim \mathbf{Bin}(m\cdot\left|\mathbf{DB}(\mathrm{kw}_i)\right|, p) + \mathbf{Bin}(m\cdot\left|\mathbf{DB}\right| - m\cdot\left|\mathbf{DB}(\mathrm{kw}_i)\right|, q),$$

where m comes from splitting the documents into shards, $\mathbf{DB}(\mathrm{kw}_i)$ denotes the set of documents containing keyword kw_i associated to query q_i, $|\mathbf{DB}|$ denotes the total number of documents, and $\mathbf{Bin}(\cdot)$ denotes a binomial distribution.

For the off-diagonal entries of the co-occurrence matrix, assume without loss of generality that the keywords in concern are kw_i and kw_j. The co-occurrence count for keywords kw_i and kw_j can increase if:

- A shard contains one of the keywords, say kw_i, and the keyword is not removed by the scheme. At the same time, the other keyword, kw_j in this case, is added to the shard.
- A shard contains none of the keywords, and both of the keywords are added to the shard.

On the other hand, the co-occurrence count for keywords kw_i and kw_j can decrease if a shard contains both of the keywords and at least one of the keywords is removed.

The actual distribution of the off-diagonal entries of the co-occurrence matrix is complicated due to dependencies. However, if we ignoring the fact that we already know the query response lengths for keywords kw_i and kw_j, the off-diagonal entries of the co-occurrence matrix can be approximated as:

$$\begin{aligned}\bar{M}(\mathsf{q}_1,\ldots,\mathsf{q}_l;\mathbf{DB})_{i,j} \sim &\mathbf{Bin}(m\cdot\left|\mathbf{DB}(\mathrm{kw}_i,\mathrm{kw}_j)\right|,p^2)\\ &+\mathbf{Bin}\left(m\cdot(|\mathbf{DB}|-|\mathbf{DB}(\mathrm{kw}_i)|-|\mathbf{DB}(\mathrm{kw}_j)|+|\mathbf{DB}(\mathrm{kw}_i,\mathrm{kw}_j)|),q^2\right)\\ &+\mathbf{Bin}(m\cdot|\mathbf{DB}(\mathrm{kw}_i)|-\left|\mathbf{DB}\mathrm{kw}_i,\mathrm{kw}_j\right|,pq)\\ &+\mathbf{Bin}(m\cdot|\mathbf{DB}(\mathrm{kw}_j)|-\left|\mathbf{DB}\mathrm{kw}_i,\mathrm{kw}_j\right|,pq).\end{aligned}$$

B Mathematical Derivations of the Likelihood Functions

Likelihood Function and its Decomposition. The likelihood function $\mathbf{L}\left[P \mid \bar{M}, M\right]$ can be written as follows:

$$\begin{aligned}&\mathbf{L}\left[P \mid \bar{M}, M\right]\\ =&\mathbf{Pr}\left[\bar{M}, M \mid P\right]\\ =&\sum_{M'\in\mathcal{N}^{N\times N}}\mathbf{Pr}\left[\bar{M}, M, M' \mid P\right]\\ =&\sum_{M'\in\mathcal{N}^{N\times N}}\mathbf{Pr}\left[\bar{M} \mid M, M', P\right]\mathbf{Pr}\left[M' \mid M, P\right]\\ =&\sum_{M'\in\mathcal{N}^{N\times N}}\mathbf{Pr}\left[\bar{M} \mid M', P\right]\mathbf{Pr}\left[M' \mid M\right],\end{aligned}$$

where N is the number of documents and $\mathcal{N}^{N\times N}$ is all N by N natural number valued matrices. In the third line of the equation, we used the law of total probability to turn the likelihood into a summation over all possible real co-occurrence matrices. The lines after break the probability into a sum of products of two probabilities. The first probability $\mathbf{Pr}\left[\bar{M}, M' \mid P\right]$ is the probability that $\bar{M}$ is the observed co-occurrence matrix and M' is the real co-occurrence matrix given P is the permutation. The second probability is the probability of getting M' as the real observed co-occurrence matrix knowing that M is the auxiliary co-occurrence matrix.

We assume the same structure of the auxiliary co-occurrence matrix M for all of our leakage functions so its derivation is shared by all three leakage functions. We note that only some of the real co-occurrence matrices generate a non-zero likelihood, as the sum of off-diagonal entries of a row must be less or equal to the diagonal entry for correctness. By writing a row of a matrix M without the i-th entry as $M_{i,\cdot}$, for those real co-occurrence matrices, we can derive the probability as:

$$\mathbf{Pr}\left[M' \mid M\right]=\sum_i\mathbf{Pr}\left[M'_{i,i} \mid M_{i,i}\right]\mathbf{Pr}\left[M'_{i,\cdot} \mid M'_{i,i}, M_{i,\cdot}\right].$$

In the second line of the above expression, the first term is the probability of getting $M'_{i,i}$ documents containing keyword kw_i, and the second term is the probability of observing the off-diagonal co-occurrence counts.

Derivation for PRT-EMMs. For **PRT-EMMs** [KM19], query response lengths may be truncated by a random amount. This means that based on the query response length in the auxiliary co-occurrence matrix M' and that in the observed co-occurrence matrix, an attacker can estimate how many documents in the off-diagonal entries are expected to be removed. For observed co-occurrence count between keywords kw_j and kw_j where $i \neq j$, the real process can be modelled as a sequential application of two hypergeometric distributions on the real co-occurrence count.

$$\prod_{i<j} \sum_{k} \mathbf{Pr}\left[\mathbf{HGeom}\left(M'_{P(i),P(i)}, M'_{P(i),P(j)}, \bar{M}_{i,i}\right) = k\right] \mathbf{Pr}\left[\mathbf{HGeom}\left(M'_{P(j),P(j)}, k, \bar{M}_{j,j}\right) = \bar{M}_{i,j}\right].$$

Derivation for FP-EMMs [PPYY19]. To simplify the first term of the likelihood decomposition, we assume independence of the entries in the observed co-occurrence matrix. Without loss of generality, we assume that all query response lengths are padded to m. This means we can express the probability as:

$$\prod_{i<j} \mathbf{Pr}\left[\mathbf{HGeom}\left(2N, 2m - 2M'_{P(i),P(i)}, 2m - 2M'_{P(j),P(j)}\right) = \bar{M}_{i,j} - M'_{P(i),P(j)}\right].$$

Derivation for DP-EMMs [PPYY19]. The first term of the likelihood decomposition for differentially private volume-hiding EMMs [PPYY19] is similar to that of the full padding version, except that the query response lengths are padded according to a Laplacian distribution as opposed to padding to the maximum query response length. Let n^* be the constant to offset the Laplacian random variable $\mathbf{Lap}(2/\epsilon)$, the first term of the likelihood decomposition can be expressed as:

$$\begin{aligned}
&\sum_{i=j} \mathbf{Pr}\left[\bar{M}, M' \mid P\right] \times \sum_{i<j} \mathbf{Pr}\left[\bar{M}, M' \mid P\right] \\
=&\sum_{i} \mathbf{Pr}\left[2M'_{P(i),P(i)} + n^* + \mathbf{Lap}(2/\epsilon) = \bar{M}_{i,i}\right] \\
&\times \sum_{i<j} \mathbf{Pr}\left[\mathbf{HGeom}(2N, 2\bar{M}_{i,i} - 2M'_{P(i),P(i)}, 2\bar{M}_{j,j} - 2M'_{P(j),P(j)}) = \bar{M}_{i,j} - 2M'_{P(i),P(j)}\right].
\end{aligned}$$

Derivation for DPAP-SE. Recall that in **DPAP-SE** [CLRZ18], the documents are split into shards and the keywords for the shards are randomized. This means that each diagonal entry of the observed co-occurrence matrix contain the counts from the real shards which have kept the keyword, and the counts from the other shards which have gained the keyword from the randomization process. Similarly, each off-diagonal entry of the observed co-occurrence matrix contain the counts from the real shards which have kept both of the keywords, and the other counts from the other shards which have gained one of the keywords or both of them from the randomization process. Let p be the probability that a shard keeps its keywords, q be the probability that a fake keyword is introduced to a shard, and m to be the number of shards, we can express the first term in the likelihood decomposition as:

$$\begin{aligned}
&\prod_{i=j} \mathbf{Pr}\left[\bar{M}, M' \mid P\right] \times \prod_{i<j} \mathbf{Pr}\left[\bar{M}, M' \mid P\right] \\
=&\prod_{i} \mathbf{Pr}\Big[\mathbf{Bin}\left(mM'_{P(i),P(i)}, p\right) + \mathbf{Bin}\left(mM'_{P(j),P(j)}, q\right) = \bar{M}_{i,i}\Big] \\
&\times \prod_{i<j} \mathbf{Pr}\Big[\mathbf{Bin}\left(mM'_{P(i),P(j)}, p^2\right) + \mathbf{Bin}\left(m\left(M'_{P(i),P(i)} - \sum_{k>1} M'_{P(i),P(k)}\right), pq\right) \\
&+ \mathbf{Bin}\left(m\left(M'_{P(j),P(j)} - \sum_{k>1} M'_{P(j),P(k)}\right), pq\right) \\
&+ \mathbf{Bin}\big(m\left(N - M'_{P(i),P(i)} - M'_{P(j),P(j)} + M'_{P(i),P(j)}\right), q^2\big) = \bar{M}_{i,j}\Big].
\end{aligned}$$

Approximation Techniques. As it can be seen, it is computationally infeasible to sum over all possible real co-occurrence matrices. We propose to sum over all possible real co-occurrence matrices such that $\mathbf{Pr}\left[M' \mid \bar{M}\right]$ is significant. In our experiment, we used symmetric endpoints on every entry of M' such that the resultant interval covers at least 95% of the probability density function. We use Normal approximation in the first term of the likelihood decomposition for **PRT-EMM** [KM19] to remove the need of convolution. To further improve the computational efficiency, we used simple rectangle rule to approximate large summations, such as the convolutions in the first term of the likelihood decomposition for **DPAP-SE** [CLRZ18].

Speeding up the `Score` function. A naïve implementation of the `Score` functions require l^2 computations per iteration, where l is the number of non-repeating queries observed. However, we note that the `Score` functions in our attacks take the shape $\prod_{i \le j} \mathbf{Pr}\left[\bar{M}_{P(i),P(j)}, M\right]$, and the neighbourhood function `Neighbour` only changes the assignment P for one or two values. Without loss of generality, let $P(a)$ be the changed assignment. It means only the probabilities with $P(a)$ involved are changed, that is, the new likelihood function can be written as

$$\prod_{\substack{i \le j \\ i,j \ne a}} \mathbf{Pr}\left[\bar{M}_{P(i),P(j)}, M\right] \times \prod_{i \le a} \mathbf{Pr}\left[\bar{M}_{P(i),P(a)}, M\right] \times \prod_{a<j} \mathbf{Pr}\left[\bar{M}_{P(a),P(j)}, M\right].$$

In our implementation, we maintain an $l \times l$ matrix where the i,j-th entry of the matrix records $\mathbf{Pr}\left[\bar{M}_{P(i),P(j)}, M\right]$. Only l (or $2l$ if the assignment is changed on two queries) of these entries (which corresponds to the probabilities in the second and third products) are updated according to the likelihood function, and the score function simply outputs the product of the entries of this matrix.

Algorithm 2 Neighbourhood Generation Algorithm for **PRT-EMM** [KM19] and **FP-EMM** [PPYY19]

```
1: procedure NEIGHBOUR($P, \bar{M}, M$)
2:     $i \xleftarrow{\$} \{1, \ldots, |\mathrm{kw}(\mathbf{DB})|\}$
3:     $j \xleftarrow{\$} \{1, \ldots, |\mathrm{kw}(\mathbf{DB})|\}$
4:     $P' \leftarrow P$
5:     $P'(i) \leftarrow j$
6:     if there exists $k$ such that $P(k) = j$ then
7:         $P'(k) \leftarrow P(i)$
8:     return $P'$
```

Algorithm 3 Neighbourhood Generation Algorithm for **DP-EMM** [PPYY19]

```
1: procedure NEIGHBOUR($P, \bar{M}, M$)
2:     $i \xleftarrow{\$} \{1, \ldots, |\mathrm{kw}(\mathbf{DB})|\}$
3:     $j \xleftarrow{\$} \{1, \ldots, |\mathrm{kw}(\mathbf{DB})|\}$
4:     $b_0 \leftarrow NM_{j,j} - 1.96NM_{j,j}(1 - M_{j,j}) - 1.96/\epsilon$
5:     $b_1 \leftarrow NM_{j,j} + 1.96NM_{j,j}(1 - M_{j,j}) + 1.96/\epsilon$
6:     if there exists $k$ such that $P(k) = j$ then
7:         $b_2 \leftarrow NM_{P(i),P(i)} - 1.96NM_{P(i),P(i)}(1 - M_{P(i),P(i)}) - 1.96/\epsilon$
8:         $b_3 \leftarrow NM_{P(i),P(i)} + 1.96NM_{P(i),P(i)}(1 - M_{P(i),P(i)}) + 1.96/\epsilon$
       /* Check the condition with k only if it exits */
9:     while $\neg(b_0 < \bar{M}i, i < b_1) \vee \neg(b_2 < \bar{M}_{k,k} < b_3)$ do
10:        Resample $i, j, k$
11:    $P' \leftarrow P$
12:    $P'(i) \leftarrow j$
13:    if there exists $k$ such that $P(k) = j$ then
14:        $P'(k) \leftarrow P(i)$
15:    return $P'$
```

C Detailed Pseudocodes

In this section, we present the detailed pseudocodes of the neighbourhood generation subroutines used for our simulated-annealing based attack. Algorithm 2 describes the neighbourhood generation subroutine used for our attack on the end-to-end SSE schemes built in a natural way from **PRT-EMM** [KM19] and **FP-EMM** [PPYY19]. Algorithm 3 describes the neighbourhood generation subroutine used for our attack on the end-to-end SSE scheme built in a natural way from **DP-EMM** [PPYY19]. Finally, Algorithm 3 describes the neighbourhood generation subroutine used for our attack on the end-to-end SSE scheme **DPAP-SE** [CLRZ18]. In the rest of the section, when we refer to **PRT-EMM**, **FP-EMM**, and **DP-EMM**, we refer to the end-to-end SSE schemes built in a natural way from the respective EMM scheme.

In order to reduce the size of the search space for **DP-EMM** and **DPAP-SE**, neighbours are picked such that the query response volumes (with padding) are 'close' to the one in the

Algorithm 4 Neighbourhood Generation Algorithm for **DPAP-SE** [CLRZ18]

```
1: procedure NEIGHBOUR($P, \bar{M}, M$)
2:     $i \xleftarrow{\$} \{1, \dots, |\text{kw}(\mathbf{DB})|\}$
3:     $j \xleftarrow{\$} \{1, \dots, |\text{kw}(\mathbf{DB})|\}$
4:     $b_0 \leftarrow kpNM_{j,j} + kqN(1 - M_{j,j}) - 1.96kNM_{j,j}(1 - M_{j,j}) - 1.96kN(p + q)$
5:     $b_1 \leftarrow kpNM_{j,j} + kqN(1 - M_{j,j}) + 1.96kNM_{j,j}(1 - M_{j,j}) + 1.96kN(p + q)$
6:     if there exists $k$ such that $P(k) = j$ then
7:         $b_2 \leftarrow kpNM_{P(i),P(i)} + kqN(1 - M_{j,j}) - 1.96kNM_{P(i),P(i)}(1 - M_{P(i),P(i)}) - 1.96kN(p + q)$
8:         $b_3 \leftarrow kpNM_{P(i),P(i)} + kqN(1 - M_{j,j}) - 1.96kNM_{P(i),P(i)}(1 - M_{P(i),P(i)}) - 1.96kN(p + q)$
       /* Check the condition with $k$ only if it exits */
9:     while $\neg(b_0 < \bar{M}i, i < b_1) \vee \neg(b_2 < \bar{M}_{k,k} < b_3)$ do
10:        Resample $i, j, k$
11:    $P' \leftarrow P$
12:    $P'(i) \leftarrow j$
13:    if there exists $k$ such that $P(k) = j$ then
14:        $P'(k) \leftarrow P(i)$
15:    return $P'$
```

auxiliary information. Specifically, we require all assignments to have padded query response volumes within 95% confidence interval of the auxiliary data (by treating the auxiliary data as samples from a random variable containing the same number of documents). To simplify the actual computation, we use Normal approximation and allow the padded query response volumes of the assignments to deviate by no more than 1.96x of the standard deviation derived from the auxiliary data.

D Experimental Data

D.1 General Information about Enron Email Corpus

The Enron email corpus [WWC] is a collection of over 600,000 emails generated by 158 employees of the Enron Corporation and acquired by the Federal Energy Regulatory Commission (FERC) during its investigation of the Enron scandal. At the conclusion of the investigation, and upon the issuance of the FERC staff report, the email corpus was released to the public for historical research and academic purposes. The Enron dataset is widely used as a target for cryptanalysis on structured encryption [IKK12,CGPR15,BKM20] as it is one of the only public, real-world datasets.

D.2 Pre-processing

We implemented our email processing and keyword extraction script in python using the Natural Language Toolkit [Pro] module as the tokeniser. The English stop words and other keywords with frequency higher than 5% are removed.

D.3 General Statistics of Enron Email Corpus

Figure 5 gives some general statistics of the Enron email corpus after pre-processing.

# documents	480,000
# keywords	33,366
# keyword-document pairs	17,415,721
Max. keyword frequency	23,989
Min. keyword frequency	1
Mean keyword frequency	522.0
Max. # keywords per document	3,483
Min. # keywords per document	1
Mean # keywords per document	36.8

Figure 5: General statistics of the Enron email corpus after pre-processing.

Figure 6 shows the frequency distribution of the 5,000 most frequent keywords after pre-processing.

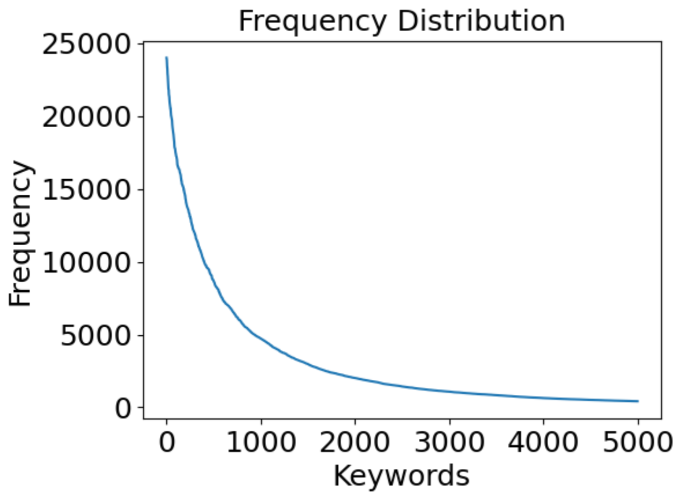


Figure 6: Frequency distribution of the 5,000 most frequent keywords.

E Measuring the Level of Noise in Auxiliary Information

In our attacks, we build auxiliary co-occurrence matrices M by sampling from the Enron email dataset. It is not clear immediately how close the co-occurrence matrices and the real ones are. Here, we propose two measurements for the level of noise.

Absolute distance. Inspired by the Kolmogorov–Smirnov test [Smi48], we define absolute distance to be the maximum absolute difference between the target co-occurrence matrix and auxiliary co-occurrence matrix:

$$D = \max_{i,j} \left| \frac{\bar{M}_{P(i),P(j)}}{N} - M_{i,j} \right| ,$$

where $\bar{M}$ is the co-occurrence matrix generated from the target database (without using any construction on top), M is the co-occurrence matrix generated for auxiliary information, P is the true keyword assignments between the queries and keywords, and N is the number of documents in the target database. Intuitively, more noisy auxiliary information means a larger absolute distance.

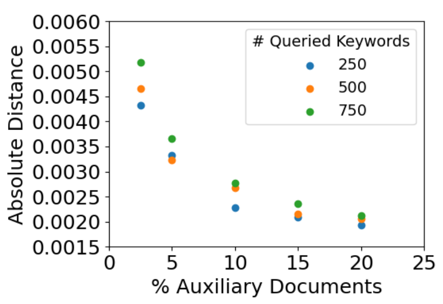
Modified Probability Score. The second measurement of the level of noise we propose is the probability score. As the name suggests, the measurement is simply:

$$\mathbf{Pr}\left[\bar{M} \mid M\right] .$$

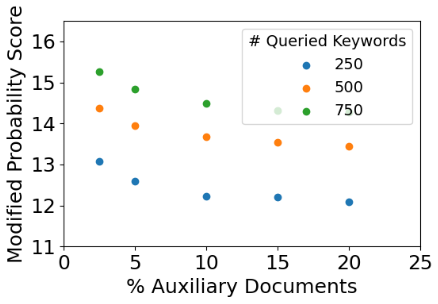
It is clear that less noisy auxiliary information produces a larger probability score.

The probability score is very small for our datasets, so we use $D = \log(-\log(\mathbf{Pr}\left[\bar{M} \mid M\right]))$ as a modified probability score instead. Less noisy auxiliary information produces a larger modified probability score just as before.

Measurements on the Level of Noise. The measurements on the level of noise for the auxiliary datasets used in our attacks can be found in Figure 7. It can be seen clearly that the absolute distance and modified probability score increase as less documents are used as auxiliary information.



(a) Absolute distance.



(b) Modified probability score.

Figure 7: Measurements of the level of noise of the auxiliary data in our experiments.