A Leakage-Abuse Attack Against Multi-User Searchable Encryption

Cédric Van Rompay, Refik Molva, Melek Önen
EURECOM, France
{vanrompa,molva,onen}@eurecom.fr

May 11, 2017

Abstract

Searchable Encryption (SE) allows a user to upload data to the cloud and to search it in a remote fashion while preserving the privacy of both the data and the queries. Recent research results describe attacks on SE schemes using the access pattern, denoting the ids of documents matching search queries, which most SE schemes reveal during query processing. However SE schemes usually leak more than just the access pattern, and this extra leakage can lead to attacks (much) more harmful than the ones using basic access pattern leakage only. We remark that in the special case of Multi-User Searchable Encryption (MUSE), where many users upload and search data in a cloud-based infrastructure, a large number of existing solutions have a common leakage in addition to the well-studied access pattern leakage. We show that this *seemingly small* extra leakage allows a very simple yet powerful attack, and that the privacy degree of the affected schemes have been overestimated. We also show that this new vulnerability affects existing software. Finally we formalize the newly identified leakage profile and show how it relates to previously defined ones.

1 Introduction

Cloud computing allows companies and individuals to outsource their data and computation. The benefits of cloud services such as increased availability and flexibility come at a high cost in terms of new security and privacy challenges. In particular a user may not be willing to let the Cloud Service Provider (CSP) access the content of her data, because the CSP could be controlled by an external attacker or could in some extreme cases be a potential adversary itself. While traditional encryption seems to be a solution to this problem, a straightforward application of encryption would hinder most advantages of cloud computing since the CSP would be unable to perform some operations over encrypted data.

Searchable Encryption (SE) schemes allow a cloud user to outsource some encrypted data and to further delegate the search operations over this encrypted data to the CSP. Starting with the seminal work of Song et al. [29], SE has since been an active research topic [17, 23, 7, 16] (for an extended state of the art see the survey of Bösh et al. [5]).

In most SE schemes, a small amount of information leakage is often tolerated in order to achieve some level of efficiency. In particular it is commonly accepted that schemes leak the *access pattern*, denoting the ids of files that match a query, since doing so allows to search in sub-linear time [9]. As a consequence a large number of attacks on SE systems [15, 6, 12, 32] focus on access pattern leakage in order to affect the largest possible set of schemes. However each scheme tends to have its own specific leakage, often including but not limited to the access pattern. In this paper we show that in the multi-user setting, the majority of the so-called Multi-User SE (MUSE) schemes share a common information leak, on top of access pattern leakage. We show that this small extra leakage leads to a simple yet very serious privacy issue, exposing those schemes to an attack that overrides all existing access-pattern-based attacks. The cause of this situation seems to be the false belief that leaking "slightly more" than the access pattern can only enable attacks "slightly more powerful" than access-pattern attacks; as pointed by Zhang et al. [32]: *"In truth, the ramifications of different types of leakage are poorly understood"*.

Contributions: Through formal discussions, proof-of-concepts on real-world software and statistical experimentations, we show how the security of most existing MUSE schemes has been widely overestimated and that the solution to the security exposures pointed in this paper would require radical changes to their design. Our results also call for more efforts towards a systematization of knowledge on leakage profiles in SE, and as a result we formally define the newly identified leakage profile and show how it relates to the ones defined in the literature.

The rest of the paper is organized as follows: In Section 2 we remind the classification of single-user SE schemes by Cash et al. and existing attacks; In Section 3 we highlight a design flaw in many MUSE schemes that exposes them to a simple but powerful leakage-abuse attack; In Section 4 we present a new leakage profile that helps analysing the security of MUSE schemes by capturing the newly identified threat; Finally in Section 5 we give directions for future work and suggestions for possible countermeasures.

2 Existing Leakage-abuse attacks in Searchable Encryption

2.1 Leakage Profiles in Searchable Encryption

In a typical Searchable Encryption (SE) scenario a user encrypts her data before uploading it to the CSP. The user can further search over her outsourced data by sending an encrypted query to the CSP. The CSP applies the received query on the encrypted data and sends back the corresponding response. Either this response is in plaintext, or the user may need to decrypt it to obtain the result of her search. In this paper we restrict our study to *keyword search*, where search queries aim at testing the presence of a keyword w in a sequence $W = (W[1], W[2], \dots)$ of keywords called *index*.

In their seminal paper, Curtmola et al. [9] initiate an approach for the security analysis of SE schemes that consists in the use of *leakage profiles*. Let a *history* of a protocol be an acceptable input to the functionality implemented by the protocol; a leakage profile is a function taking as input a history of a protocol and which output characterizes the information an adversary can learn by taking part in the execution of the scheme.

Definition 1. A protocol is said to have a leakage profile $\mathcal{L}$ (or to be secure w.r.t leakage profile $\mathcal{L}$) if there exists an efficient algorithm called *simulator* and noted Sim which on input $\mathcal{L}(\mathcal{H})$ with $\mathcal{H}$ a history of the protocol, outputs a view $\mathsf{Sim}(\mathcal{L}(\mathcal{H}))$ that is indistinguishable from the view of an adversary in a real execution of the protocol with input $\mathcal{H}$.

For SE protocols, the history consists of the indexes $W_1, W_2, \dots$ that were uploaded and the queries $w_1, w_2, \dots$ that were sent. The practical consequence of proving that a scheme has a leakage profile $\mathcal{L}$ is the guarantee that the adversary cannot compute anything that cannot be computed from $\mathcal{L}(\mathcal{H})$.

The main statement of Curtmola et al. [9] is that a SE scheme cannot be trivially insecure if it does not reveal more than the *access pattern* of queries, that is the ids of indices matching a query, and the *search pattern* that is the information of whether two queries are the same or not. These two notions of access pattern and search pattern have since been frequently used in order to describe the leakage of SE schemes.

2.2 The Existing Leakage Profile Classification

Cash et al. [6, Section 2.1] remark that a large number of papers follow the method of Curtmola et al. [9] as they resort to leakage profiles for the security analysis of SE schemes. Cash et al. [6] generalize this approach by suggesting a classification of most existing SE schemes based on their leakage profile. To this end they define four profiles named from L1, that reveals the least information, to L4 that reveals the most information. We remind the definitions of the L1 to L3 profiles, and omit L4 as it reveals too much information to be useful for our study.

The **L1 leakage profile** or *query-revealed occurrence pattern* regroups SE schemes whereby the CSP sees the access pattern of each search query (the *occurrence pattern* of a keyword denotes the indices where this keyword occurs in the dataset). Let $\{W_i\}$ be the set of uploaded indices and $(w_1, w_2, \ldots, w_Q)$ the Q queries received by the CSP, the information contained in the L1 profile can be represented as:

$$\{i \mid w_1 \in W_i\}, \{i \mid w_2 \in W_i\}, ..., \{i \mid w_Q \in W_i\}$$

The **L2 profile** or *fully-revealed occurrence pattern* corresponds to schemes where the CSP can see the occurrence patterns directly from the uploaded data, before having received any query. This justifies the naming of the profile as "fully-revealed" instead of "query-revealed" as in L1. Let $\{w_1, w_2, \ldots, w_n\}$ be the set of all keywords in the uploaded data, the information contained in the L2 profile can be represented as:

$$\{i \mid w_1 \in W_i\}, \{i \mid w_2 \in W_i\}, ..., \{i \mid w_n \in W_i\}$$

Note that the information revealed by the L1 profile gets closer to the information revealed by the L2 profile as the number of received queries grows.

Finally, schemes within the **L3 profile** or *fully-revealed occurrence pattern with keyword order* reveal the position of keywords in the plaintext index in addition to their co-occurrence. For example, the CSP can see if the third keyword of W_i is the same as the fifth keyword of W_j. Let $\{w_1, w_2, \ldots, w_n\}$ be the set of all keywords as in L2, the information contained in the L3 profile can be represented as:

$$\{(i,l) : W_i[l] = w_1\}, \ldots, \{(i,l) : W_i[l] = w_n\}$$

2.2.1 Prior Knowledge and Existing Leakage Abuse Attacks

Cash et al. [6, Section 2.1] remark that defining the leakage profile of a scheme *"does not clarify what an attacker can learn from the information that is leaked"*. Consequently, they present practical attacks against the different leakage profiles in order to give an idea of how much privacy can be expected from each profile. The information leaked by the profiles is not sufficient for the CSP to recover any data or queries; However in any practical situation, the CSP will be able to obtain some information in a way that is independent from what is leaked by the scheme. For instance the CSP might know that the hosted data is mainly English text, or that the files are financial reports, etc. Cash et al. call this external information *prior knowledge*. The attacks they present, called *leakage-abuse attacks*, show the extent to which privacy can be violated when a given leakage profile is combined with a given prior knowledge.

We now quickly recall the main existing leakage-abuse attacks. An attack named "count attack" targeting the L1 profile is presented in [6]. The prior knowledge required by the count attack consists in the probability of co-occurrence of keywords (for instance "New" is very likely to occur if "York" is present) as well as the number of indices containing each keyword. With this information, the count attack allows the CSP to recover the content of some queries. The authors show that the count attack outperforms the similar "IKK attack" presented by Islam et al. [15]. Cash et al. also present a simple attack against the L3 profile requiring that the CSP knows the plaintext of a small number of documents. Since L3 reveals the order of keywords and their occurrence patterns, the CSP is able for each keyword in a known document to see where this keyword occurs in other documents. Active attacks, for instance where the adversary can plant arbitrary documents in the system, were also presented by Cash et al. [6, Section 5.2] and by Zhang et al. [32], but we only consider passive attacks in this paper.

A recent paper by Grubbs et al. [12] present many ways of attacking a software framework named Mylar [24] whose building blocks include a SE protocol by Popa and Zeldovich [25]. Among the attacks presented in [12], many of them only leverage implementation or design issues of Mylar and are not related to the underlying cryptographic protocols [12, Section 6], or require an active attacker [12, Section 8]. The few attacks that fall in the scope of the present paper [12, Section 7] only make a basic use of simple access pattern leakage and can be related to previous attacks against L1 profile. We also note that the leakage abuse attacks in [12] either leverage specific properties of the application under attack and cannot be extended to cover any application using the underlying SE scheme, or target application that do not even use any SE scheme ("MDaisy" and "MeteorShop" applications).

3 Leakage-Abuse Attacks against MUSE

Recent papers on leakage-abuse attacks [15, 6, 32, 21, 22, 12] show that the threat caused by prior knowledge is becoming a matter of concern in single-user SE. This threat is even more serious in the multi-user setting where the dataset is owned and accessed by a number of users and users do not trust other users except the ones they explicitly authorize to search their data. In this section we show that a large number of MUSE schemes share a common design flaw which, combined with the specificity of the multi-user setting, creates a very serious vulnerability that has been neglected so far.

3.1 Multi-User Searchable Encryption

A MUSE system involves a set $R = \{r_i\}$ of users called *readers* and a set $S = \{s_j\}$ of users called *writers*. Each writer s_j uploads an encrypted version C_j of her index W_j to the CSP (for simplicity, we assume that each writer stores a single index). Each writer can authorize an arbitrary set of readers to search her index; the authorizations are represented by an *authorization graph* whereby vertices are the readers and writers and an edge between a reader r_i and a writer s_j represents the fact that r_i has been authorized to search the index of s_j. The history of a MUSE protocol is slightly different from the one of a single-user SE protocol: it consists of the index of each writer, the queries which consist of pairs (w, r) where w is the queried keyword and r is the querying reader, and of the authorization graph represented by a function $Auth$ which on input a reader outputs the ids of the indices this reader was allowed to search.

MUSE is a recent but active research topic [4, 31, 25, 11, 33, 3, 30, 14, 27, 24]. We note however that despite the obvious risk that the CSP obtains some prior knowledge in a MUSE system through colluding users, only few schemes on MUSE [25, 27, 24] consider this risk in their security model.

3.2 Common Weakness for Many MUSE Schemes

A large number of MUSE schemes [4, 31, 25, 11, 33, 3, 30, 14, 24] share a common algorithmic structure regarding the application of queries on encrypted indices. In this structure, which we call *iterative testing*, an encrypted index denoted as C_i is built by encrypting each keyword of W_i separately. Further below we call an encrypted keyword a *hash*. The order of the hashes in C_i does not need to be related to the keywords in W_i, and as a result C_i can be described as follows:

$$C_i \leftarrow \mathsf{ReOrder}(\mathsf{Enc}(w) \forall w \in W_i)$$

where Enc is the keyword encryption function of the iterative-testing-based MUSE protocol and $\mathsf{ReOrder}$ is some reordering algorithm, typically a lexicographical ordering. A query is then applied on C_i by applying the query against each hash separately. This implementation of keyword search is intuitive and efficient, but it lets the CSP discover a significant amount of information: Not only the scheme leaks the ids of matching indices (the *access pattern*); but for each matching index, the scheme also reveals *which hash* the query matched. This leakage can be represented as:

$$\begin{array}{l} \{(i, l) : i \in Auth(r) \ \wedge \ C_i[l] \text{ matches } w\} \\ \text{for each query } (w, r) \end{array}$$

This leakage looks like a query-revealed version of the L3 leakage profile, with the difference that the revealed information is the position of the matching hash in the encrypted index C_i and not the position of the searched keyword in the index W_i. Now these two positions do not need to be correlated in iterative testing, and in most schemes the order of the hashes in an encrypted index is arbitrary. For this reason, it is commonly assumed that revealing the exact location of the matching hash gives no useful information to the CSP, and that such a leakage is equivalent to simply leaking the access pattern, i.e. to a L1 profile. This assumption is explicit in Cash et al [6]; it is also implicit in works that study iterative-based MUSE schemes while only mentioning "access pattern" attacks [12, 25, 4].

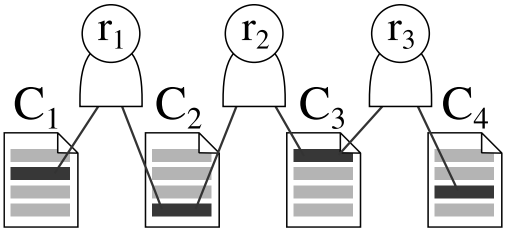


Figure 1: An illustration of the leakage caused by the iterative testing structure.

```
**For** each encrypted query $q$ observed by the adversary:
  Let $\mathcal{C}$ be a new equivalence class containing $q$;
  **For** each hash $C_i[l]$ matching $q$:
    **If** $C_i[l]$ is already in a class $\mathcal{C}_0$:
      Add all members of $\mathcal{C}$ to $\mathcal{C}_0$;
      Set $\mathcal{C} \leftarrow \mathcal{C}_0$;
    **Otherwise** (not in any existing class):
      Add the hash to $\mathcal{C}$;
Initialize the ouput value as an empty set;
**For** each $(x, w)$ in the prior knowledge, with $x$ a query or hash and $w$ a keyword:
  Let $\mathcal{C}$ be the class of $x$ (possibly containing only $x$);
  Add $(\mathcal{C}, w)$ to the output set
```

Figure 2: An example of attack algorithm leveraging the leakage caused by iterative testing

The leakage caused by iterative testing introduces a simple yet very serious vulnerability that would not be present in a scheme revealing *no more than* the access pattern. This vulnerability, illustrated in Figure 1, comes from the fact that hashes from different indices matching the same query, and conversely queries from different users matching the same hash, can only correspond to the same keyword. More precisely when the CSP observes a query matching a set of hashes, or a hash matching a set of queries, it can create an equivalence class representing the fact that these objects (both queries and hashes) all represent the same keyword. Equivalence classes can be merged when they intersect, resulting in equivalence relations even between hashes that are very "far" from each other in the authorization graph. Finally by learning the keyword corresponding to a single hash or query the CSP immediately obtains the content of *all hashes and queries* identified as "equivalent" to the hash or query that was revealed. Figure 2 presents the attack in a more formal way. Note that one could easily extract more information on the data and queries than what this attack does, but we prefer presenting a simple algorithm for the sake of clarity.

The prior knowledge required for our attack, mentioned in Figure 2, consists in hash-keyword and query-keyword pairs representing the information that "this hash (resp. query) corresponds to this plaintext keyword". We believe that the CSP can easily obtain some prior knowledge of this nature in the multi-user setting whereby some users can collude with the CSP. Indeed in such a setting the CSP has access to:

- the plaintext keywords corresponding to the hashes in the encrypted indices of colluding writers;
- the plaintext keywords corresponding to queries sent by colluding readers.

The only other input to the attack is the (perfectly legitimate) queries received by the CSP. The attack will then reveal the plaintext of some queries and hashes that can be very far, in the sense of the authorization graph, from the indices concerned by the prior knowledge. Said differently, the CSP obtains information colluding users did not have access to.

The simplicity and efficiency of this attack must be compared with the complexity of attacks that only exploit the basic access pattern leakage, i.e. attacks against L1 or L2 profiles: The IKK attack [15] uses *Simulated Annealing* and, according to Cash et al. [6], *"does not perform well even when just a small fraction of documents are unknown to the attacker"*; The count attack [6] improves over the IKK attack but remains quite complicated. Finally Grubbs et al. [12] do not provide evaluation results of the attacks they present but these results should not exceed the efficiency reached by the other existing attacks. As we show in Section 4, the attack we present outperforms all of these attacks by a large margin.

As a consequence, for SE schemes that are vulnerable to the attack presented in this paper, there is no more interest to study attacks based on access pattern leakage: these attacks are overridden due to the efficiency and simplicity of our new attack. Unfortunately, a large number of MUSE schemes [4, 31, 25, 11, 33, 3, 30, 14, 24] are in this situation since they do use the iterative testing structure and the adversary can easily obtain the required prior knowledge in the multi-user setting. In the next section we discuss the impact of this new attack on an existing MUSE scheme named MK (Multi-Key searchable encryption) as well as on an existing software framework named Mylar.

3.3 Application of the attack against MK and Mylar

Popa and Zeldovich were the first to address the situation whereby the CSP is colluding with some users in [25]: They aim at designing a MUSE scheme in which a user cannot obtain information on a document if she was not granted access to it, even in the case where she colludes with the CSP. To this end, Popa and Zeldovich present a scheme that is based upon the work of Bao et al. [4] but brings a major modification: While in the scheme of Bao et al. all indices are re-encrypted from the key of their corresponding writer to a master key owned by a Trusted Third Party (TTP), Popa and Zeldovich suggest to leave each index encrypted under the key of its owner and to instead re-encrypt each query into one different token per index to search. The resulting scheme, named *Multi-Key Search* or MK for short, provides more privacy against a colluding user than the scheme of Bao et al. because each index is encrypted under a different key.

Grubbs et al. [12] show that the definition of security of [25] is flawed by presenting an obviously insecure scheme that satisfies the definition. Additionally in the paper describing Mylar [24], the authors acknowledge that [25] does not protect against access pattern attacks. However none of these works manage to give a proper measure of how insecure the MK scheme is because they do not mention any attack like the one we present.

Indeed the MK scheme, despite its focus on user-collusion resistance, implements the iterative testing structure. This can be easily seen in the description of the scheme [25, Section 6]: in order to search an index, the algorithm $\mathsf{MK.Match}$ is iterated over each hash in the encrypted index and returns "True" if and only if the hash matches the query.

We now show how this weakness of the MK scheme allows to very efficiently break the privacy of the Mylar software framework that is based on MK. Mylar [24], described as *"a platform for building web applications which protects data confidentiality against attackers with full access to servers"*, uses the MK scheme to enable shared encrypted cloud-hosted data. It received a large attention both from the academia [19, 18, 28, 13, 20] and the industry[1].

A large number of security issues of Mylar have been very recently presented by Grubbs et al. [12] but very few of them are related to the underlying MK scheme; one could then hope that a better implementation of the framework could result in a quite secure solution, without having to change the cryptographic protocol used for MUSE operations. This would be a serious mistake, as we show that exploiting the weakness of the MK scheme suffices to break privacy in Mylar.

[1] http://bgr.com/2014/03/27/mylar-website-encryption-technology/
http://motherboard.vice.com/read/want-to-keep-data-private-encrypt-it-before-it-even-reaches-a-server
http://www.ibm.com/developerworks/cloud/library/cl-always-on-data-encryption-for-cloud-security
http://spectrum.ieee.org/computing/software/how-to-compute-with-data-you-cant-see

We only give the general idea and the practical results of our experiments here, but the source code of our implementation and the instructions necessary to reproduce the attack are publicly available [2]. Our proof-of-concept implementation targets EncChat, a "secure" chat application that has been built using Mylar. In the EncChat application, each message is considered as a document and an index is created out of the words contained in the message. This index is encrypted through the MK scheme and uploaded to the CSP. A user receives an authorization for each chat room it is given access to, and can search over all these chat rooms by sending a single encrypted query thanks to the use of the MK scheme. For each message searched, because of the iterative testing structure, the CSP sees precisely which hash matched the query. The CSP is then able to identify equivalent hashes across different messages and different chat rooms, as was illustrated in Figure 1. By corrupting a user, the CSP gets access to every message in every chat room this user had access to. Finally, by combining these two sources of information, the CSP can deduce the content of many messages and queries using the algorithm of Figure 2. Users can be affected even if they are very far from the corrupted user in the authorization graph.

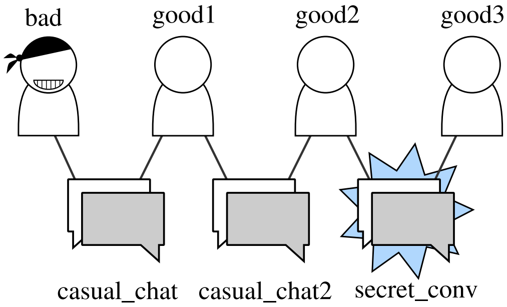


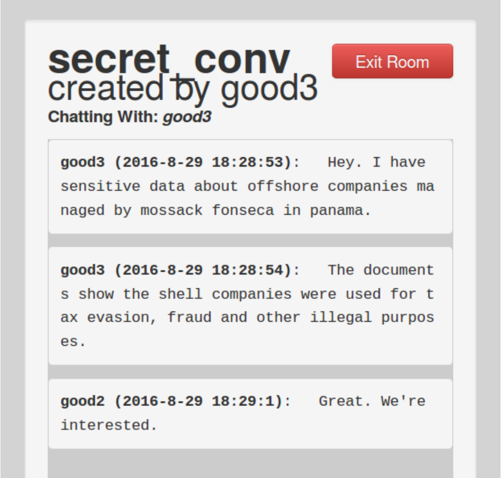
Figure 3: The users and chat rooms in our attack scenario against EncChat. The CSP by taking control of user "bad" is able to recover keywords in the chat room "secret_conv".

Figure 3 illustrates our attack scenario: two users "good2" and "good3" are having a very sensitive conversation in chat room "secret_conv". Only the two of them have access to this chat room. At the same time, "good2" is having another conversation with user "good1". This user is completely trustworthy and will provide no prior knowledge at all to the CSP. However user "good1" is in a chat room with a user named "bad" which does reveal information to the CSP. By observing the queries made by all users and by obtaining information through user "bad", the CSP is able to recover keywords in the chat room "secret_conv" as is shown in Figure 4. Note that we only present a small-scale scenario here which purpose is to confirm that our attack affects MK-based schemes like Mylar. The next section presents quantitative results that allow to estimate the amount of (in)security provided by any MUSE scheme implementing iterative testing.

4 A new leakage profile for MUSE

The vulnerability pointed out in this paper, albeit very simple, has been neglected by a large number of works focusing on the MUSE problem. We believe that one of the reasons why this vulnerability has been neglected before is the lack of systematization of the existing knowledge on leakage profiles: In the absence of a comprehensive (enough) framework for understanding information leakage in SE, one may try to resort to intuition to evaluate the dangerousness of a given leakage profile, and consider (by mistake) that the iterative-testing structure should provide as much privacy as any SE scheme known to "leak the access pattern".

In that sense the approach initiated by Cash et al. [6] consisting in categorizing leakage profiles is of major importance. Consequently, in this section we integrate the threat this paper presents in the framework of [6]. We define a new leakage profile named *keyword access pattern* or $\mathrm{L}_{\mathrm{KWAP}}$ that is computed by the algorithm in Figure 5.



secret_conv
created by good3
Chatting With: *good3*

good3 (2016-8-29 18:28:53): Hey. I have sensitive data about offshore companies managed by mossack fonseca in panama.

good3 (2016-8-29 18:28:54): The documents show the shell companies were used for tax evasion, fraud and other illegal purposes.

good2 (2016-8-29 18:29:1): Great. We're interested.

```
### REBUILT CONVERSATION ###
### CHATROOM: 'secret_conv' ###
good3 (2016-08-29 18:28:53): [fonseca, data,
                              mossack, panama]
good3 (2016-08-29 18:28:54): [evasion, tax]
```

Figure 4: The conversation "secret_conv" seen by user "good2" in the EncChat application, compared to the output of our attack program run by the CSP.

Where $Auth(r)$ indicates the indices for which reader r has authorization and σ_i is a randomly-chosen permutation of W_i. In the rest of the section we show that introducing the $\mathrm{L_{KWAP}}$ profile is necessary at least in the MUSE problem where the categories defined by Cash et al. do not capture its dangerousness in a precise enough manner We then show where the $\mathrm{L_{KWAP}}$ profile should be situated in the hierarchy defined by Cash et al.

4.1 Difference between $\mathrm{L_{KWAP}}$ and L1

First we confirm in the framework of Cash et al. [6] our claim that $\mathrm{L_{KWAP}}$ should not be considered as equivalent to L1.

The attack presented in Figure 1 is an (partial) index-recovery attack and is effective with a very small amount of prior knowledge; it would then be difficult to compare it with the count attack of Cash et al. [6] which is a query-recovery attack and requires a large amount of prior knowledge (at least 80% of the dataset corpus must be known by the CSP). We then design a query-recovery attack against $\mathrm{L_{KWAP}}$ that we think makes comparison between $\mathrm{L_{KWAP}}$ and L1 easier. Our attack is trivial: The CSP has some prior knowledge in the form of tuples $(w, (i, l))$, each such tuple meaning that the hash $C_i[l]$ represents the keyword w. In our setting an index is either entirely revealed to the CSP (all of its hashes are in the prior knowledge) or not revealed at all. As we already said, this kind of prior knowledge is immediate in a MUSE system based on iterative testing when some users collude with the CSP. The query recovery process is as follow: If a query matches one of the known hashes, the query is recovered; otherwise it is not.

We evaluated this attack experimentally, following a methodology that is as close as possible to the one of Cash et al. [6]: the dataset used is a subset of 30,000 mails from the ENRON email dataset [1], indices are built by taking all words in the corpus, removing stop words, applying the porter stemming algorithm [26], then selecting the 500 most common words obtained this way as the index vocabulary. The source code

```
Input: A MUSE history
For each index $W_i$:
    Pick a random permutation $\sigma_i$;
For each query $(w, r)$:
    Append $\{(i, l) : i \in Auth(r) \wedge W_i[\sigma_i(l)] = w\}$
    to result;
Return result.
```

Figure 5: Algorithm to compute $\mathrm{L}_{\mathrm{KWAP}}$ from a MUSE history

necessary to reproduce all experiments is publicly accessible [2].

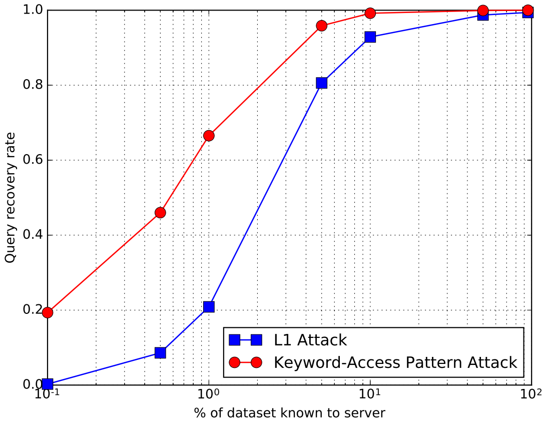


Figure 6: Results of query recovery attack against keyword-access pattern leakage, compared with attack against L1 leakage profile in the same scenario.

Experimental results show a very good success rate of the attack (red discs in Figure 6): for instance when the amount of revealed indices represent only 1% of all indices, the CSP is still able to recover about 65% of the queries. However comparing these results with the one of the count attack would not be fair, the prior knowledge of the count attack being very different from the one of our attack. We then design an attack against a L1 profile that uses the same prior knowledge as the one we designed against a $\mathrm{L}_{\mathrm{KWAP}}$ profile. The attack against a L1 profile consists in keeping track of the keywords that have unique occurrence pattern among the revealed indices: For instance, the CSP can remember that *only the keyword "dog" appears in* W_3 *and* W_5 *but not in* W_8. Queries targeting such keywords are recovered; other queries are not recovered. The performance of this attack (blue squares in Figure 6) is much lower than the corresponding attack against $\mathrm{L}_{\mathrm{KWAP}}$, especially when only a small fraction of the dataset is revealed to the CSP.

This significant difference already shows how different access-pattern leakage and keyword-access-pattern leakage are. To this initial difference must be added the fact that with a $\mathrm{L}_{\mathrm{KWAP}}$ profile, each new query may reveal the keyword corresponding to new hashes and thus increase the amount of knowledge of the CSP, making query recovery more efficient in return. This phenomenon, almost non-existent in a L1 profile, is very noticeable in the experiments presented in the next section.

4.2 Situating L_{KWAP} in the existing hierarchy of leakage profiles

In this section we show that the L_{KWAP} leakage profile can be considered as the query-revealed variant of the L3 profile, in the same way L1 is the query-revealed variant of the L2 profile. We insist again on the fact that, in a strict sense, a query-revealed variant of the L3 profile is different from a L_{KWAP} profile since the L3 profile reveal the position of keywords in an index while the L_{KWAP} profile reveals the position of hashes in an encrypted index.

Again we define two attacks and compare their efficiency. We compare the attack we described in Figure 2 with the attack against the L3 profile presented by Cash et al. [6] in which any keyword present in a revealed index is recovered in all the other indices. The prior knowledge is of the same type as in the previous section and the experimental methodology is the same except that the size of the vocabulary is 5,000, in order to stay close to the experiments of [6]. Because we need to define an authorization graph for the attack against L_{KWAP} that takes place in the multi-user setting, we consider 50 readers and give each of them access to 3,000 randomly-chosen index. Finally we reveal 20 indices out of the 30,000 as in the "Enron-20" experiment of [6].

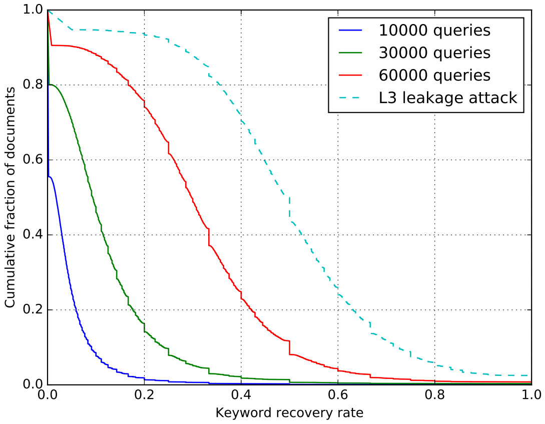


Figure 7: Results of our attack for 10,000, 30,000 and 60,000 observed queries, and results of the attack of [6] against L3 profiles for comparison.

Figure 7 shows the cumulative distribution of keyword recovery for the attack against L_{KWAP} with 10,000, 30,000 and 60,000 observed queries, and for the attack against L3. For instance we can see that with 60,000 observed queries, 90% of all indices have at least 10% of their keywords recovered and 50% of all indices have at least 30% of their keywords recovered. The damage on privacy caused by L_{KWAP} gets closer to the damage of a L3 profile as the number of observed queries grows; this allows us to conclude that, in terms of dangerousness, a L_{KWAP} profile can be considered as query-revealed version of a L3 profile. This experiment is also the opportunity to stress again how different L1 and L_{KWAP} are: with such a small ratio of revealed indices ($20/30,000 = 0.07\%$), an attack against L1 profile would produce close to no results as can be seen in Figure 6.

We thus suggest to add the L_{KWAP} leakage profile to the existing classification of Cash et al. [6], and to consider it as a profile that can lead to serious threats in some settings as MUSE. Doing so would help spreading the idea that mentioning access-pattern attacks makes no sense in a context where keyword-access-pattern attacks are possible, and should help noticing the kind of vulnerability we identify.

5 Countermeasures and Future Work

An interesting question for future work would be the presence of such vulnerability in single-user SE systems. Indeed even in state-of-the-art SE protocols as [8], the CSP can observe the memory addresses at which matches happen. However in a single-user setting the phenomenon of user collusion is not present any more, so an attack scenario where the adversary can exploit this extra leakage remains to be found.

A more urgent matter concerns the design of a MUSE system that is not affected by the vulnerability presented in this paper. As of today the only existing MUSE scheme that does not follow the iterative-testing structure seems to be the one of [27]. However this scheme is for now too inefficient to be considered practical: For instance the workload of a reader during a query scales linearly with the total number of indices being searched. While many papers are being published describing possible attacks using the access pattern, having an efficient protocol for the MUSE problem that reveals no more than the access pattern would actually be a significant improvement.

To build a MUSE protocol that reveals no more than the access pattern one must make sure than, in the view of the CSP, all queries matching the same indices are indistinguishable. Solutions could involve private information retrieval as [27] or oblivious transfer, but a scalable use of these protocols for MUSE remains to be found; another option could be based on a construction akin to cryptographic accumulators (see [10] for a good overview of the topic) where the CSP has no choice but to process an encrypted index as a single, indivisible entity.

Finally as more and more attacks using access pattern leakage are published, revealing the access pattern could soon be considered as too insecure for a MUSE system (or even a SE system) in practice. It would then be interesting to study protocols with higher privacy guarantees, with the possible cost of lack of efficiency and decreased functionality.

6 Conclusion

Recent work on leakage-abuse attacks paved the way for a better understanding of privacy in SE. Nevertheless, efforts are still needed before the various types of leakage in SE are properly understood. In particular, the growing popularity of access-pattern attacks must not hide the fact that most schemes reveal more than just the access pattern, and can thus be exposed to less general but more powerful attacks. This caveat is even stronger in the multi-user setting and our work shows that the MUSE problem still lacks a solution that is efficient and that offers an acceptable level of privacy at the same time. Our definition and study of the keyword-access pattern leakage profile should improve the security analysis of future MUSE schemes.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable feedback and comments. This work was partially supported by the TREDISEC project (G.A. no 644412), funded by the European Union (EU) under the Information and Communication Technologies (ICT) theme of the Horizon 2020 (H2020) research and innovation programme.

References

[1] Enron email dataset. `http://www.cs.cmu.edu/~enron/` Accessed on May 2016.

[2] Source code to reproduce our experiments. `http://www.eurecom.fr/~vanrompa/iterative-testing/`

[3] M. R. Asghar, G. Russello, B. Crispo, and M. Ion. Supporting complex queries and access policies for multi-user encrypted databases. In *CCSW'13, Proceedings of the 2013 ACM Cloud Computing Security Workshop, Co-located with CCS 2013, Berlin, Germany, November 4, 2013*, pages 77–88, 2013.

[4] F. Bao, R. H. Deng, X. Ding, and Y. Yang. Private Query on Encrypted Data in Multi-user Settings. In *Information Security Practice and Experience, 4th International Conference, ISPEC 2008, Sydney, Australia, April 21-23, 2008, Proceedings*, pages 71–85, 2008.

[5] C. Bösch, P. Hartel, W. Jonker, and A. Peter. A Survey of Provably Secure Searchable Encryption. *ACM Computing Surveys*, 47(2):1–51, Aug. 2014.

[6] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart. Leakage-Abuse Attacks Against Searchable Encryption. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 668–679, 2015.

[7] D. Cash, J. Jaeger, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Rosu, and M. Steiner. Dynamic searchable encryption in very large databases: Data structures and implementation. In *Proc. of NDSS*, volume 14, 2014.

[8] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, and M. Steiner. Highly-scalable searchable symmetric encryption with support for boolean queries. In *Advances in Cryptology–CRYPTO 2013*, pages 353–373. Springer, 2013.

[9] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. *Journal of Computer Security*, 19(5):895–934, 2011.

[10] D. Derler, C. Hanser, and D. Slamanig. Revisiting Cryptographic Accumulators, Additional Properties and Relations to Other Primitives. In *Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings*, pages 127–144, 2015.

[11] C. Dong, G. Russello, and N. Dulay. Shared and Searchable Encrypted Data for Untrusted Servers. In *Data and Applications Security XXII, 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security, London, UK, July 13-16, 2008, Proceedings*, pages 127–143, 2008.

[12] P. Grubbs, R. McPherson, M. Naveed, T. Ristenpart, and V. Shmatikov. Breaking Web Applications Built On Top of Encrypted Data. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1353–1364, 2016.

[13] F. Hahn and F. Kerschbaum. Searchable Encryption with Secure and Efficient Updates. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 310–320, 2014.

[14] Y. H. Hwang and P. J. Lee. Public Key Encryption with Conjunctive Keyword Search and Its Extension to a Multi-user System. In *Pairing-Based Cryptography - Pairing 2007, First International Conference, Tokyo, Japan, July 2-4, 2007, Proceedings*, pages 2–22, 2007.

[15] M. Islam, M. Kuzu, and M. Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. *Network and Distributed System Security Symposium (NDSS'12)*, 2012.

[16] S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner. Outsourced symmetric private information retrieval. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 875–888. ACM, 2013.

[17] S. Kamara and C. Papamanthou. Parallel and Dynamic Searchable Symmetric Encryption. In *Financial Cryptography and Data Security - 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers*, pages 258–274, 2013.

[18] N. Karapanos, A. Filios, R. A. Popa, and S. Capkun. Verena: End-to-end integrity protection for web applications. To appear in 37th IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 23-25, 2016.

[19] C. Lan, J. Sherry, R. A. Popa, and S. Ratnasamy. Embark: Securely outsourcing middleboxes to the cloud. To appear in 13th USENIX Symposium on Networked Systems Design and Implementation, NSDI 16, Santa Clara, CA, USA, March 16-18, 2016.

[20] A. Levy, H. Corrigan-Gibbs, and D. Boneh. *Stickler: Defending Against Malicious CDNs in an Unmodified Browser.* 2015.

[21] C. Liu, L. Zhu, M. Wang, and Y.-a. Tan. Search pattern leakage in searchable encryption: Attacks and new construction. *Inf. Sci.*, 265:176–188, 2014.

[22] M. Naveed, S. Kamara, and C. V. Wright. Inference Attacks on Property-Preserving Encrypted Databases. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 644–655, 2015.

[23] V. Pappas, F. Krell, B. Vo, V. Kolesnikov, T. Malkin, S. G. Choi, W. George, A. Keromytis, and S. Bellovin. Blind Seer: A Scalable Private DBMS. In *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, SP '14, pages 359–374, Washington, DC, USA, 2014. IEEE Computer Society.

[24] R. A. Popa, E. Stark, S. Valdez, J. Helfer, N. Zeldovich, and H. Balakrishnan. Building Web Applications on Top of Encrypted Data Using Mylar. In *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2014, Seattle, WA, USA, April 2-4, 2014*, pages 157–172, 2014.

[25] R. A. Popa and N. Zeldovich. Multi-Key Searchable Encryption. *IACR Cryptology ePrint Archive*, 2013:508, 2013.

[26] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[27] C. V. Rompay, R. Molva, and M. Önen. Multi-user Searchable Encryption in the Cloud. In *Information Security - 18th International Conference, ISC 2015, Trondheim, Norway, September 9-11, 2015, Proceedings*, pages 299–316, 2015.

[28] F. Schuster, M. Costa, C. Fournet, C. Gkantsidis, M. Peinado, G. Mainar-Ruiz, and M. Russinovich. VC3: Trustworthy Data Analytics in the Cloud Using SGX. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 38–54, 2015.

[29] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *Security and Privacy, 2000. S P 2000. Proceedings. 2000 IEEE Symposium on*, pages 44–55, 2000.

[30] P. Wang, H. Wang, and J. Pieprzyk. Common Secure Index for Conjunctive Keyword-Based Retrieval over Encrypted Data. In *Secure Data Management, 4th VLDB Workshop, SDM 2007, Vienna, Austria, September 23-24, 2007, Proceedings*, pages 108–123, 2007.

[31] Y. Yang, H. Lu, and J. Weng. Multi-User Private Keyword Search for Cloud Computing. pages 264–271. IEEE, Nov. 2011.

[32] Y. Zhang, J. Katz, and C. Papamanthou. All Your Queries Are Belong to Us: The Power of File-Injection Attacks on Searchable Encryption. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 707–720, Austin, TX, Aug. 2016. USENIX Association.

[33] F. Zhao, T. Nishide, and K. Sakurai. Multi-User Keyword Search Scheme for Secure Data Sharing with Fine-Grained Access Control. In *Information Security and Cryptology - ICISC 2011 - 14th International Conference, Seoul, Korea, November 30 - December 2, 2011. Revised Selected Papers*, pages 406–418, 2011.