Access Pattern disclosure on Searchable Encryption: Ramification, Attack and Mitigation

Mohammad Saiful Islam, Mehmet Kuzu, Murat Kantarcioglu
Jonsson School of Engineering
and Computer Science
The University of Texas at Dallas
{saiful, mehmet.kuzu, muratk}@utdallas.edu

Abstract

The advent of cloud computing has ushered in an era of mass data storage in remote servers. Remote data storage offers reduced data management overhead for data owners in a cost effective manner. Sensitive documents, however, need to be stored in encrypted format due to security concerns. But, encrypted storage makes it difficult to search on the stored documents. Therefore, this poses a major barrier towards selective retrieval of encrypted documents from the remote servers. Various protocols have been proposed for keyword search over encrypted data to address this issue. Most of the available protocols leak data access patterns due to efficiency reasons. Although, oblivious RAM based protocols can be used to hide data access patterns, such protocols are computationally intensive and do not scale well for real world datasets. In this paper, we introduce a novel attack that exploits data access pattern leakage to disclose significant amount of sensitive information using a modicum of prior knowledge. Our empirical analysis with a real world dataset shows that the proposed attack is able to disclose sensitive information with a very high accuracy. Additionally, we propose a simple technique to mitigate the risk against the proposed attack at the expense of a slight increment in computational resources and communication cost. Furthermore, our proposed mitigation technique is generic enough to be used in conjunction with any searchable encryption scheme that reveals data access pattern.

1. Introduction

Searching over remote encrypted data (commonly referred to as Searchable Encryption) has been an active area of research for the last few years. Due to increased popularity of cloud based services, more and more sensitive data (e.g., health care records, personal emails etc.) are stored encrypted in the cloud. But, the advantage of cloud data storage is lost if the user can not selectively retrieve segments of their data. Therefore, we need secure and efficient search schemes to selectively retrieve sensitive data from the cloud. The need for such protocols are also recognized by researchers from major IT companies such as Microsoft [14].

Although, there are quite a few proposed solutions for searchable encryption schemes, the basic settings remains the same. There is a set of clients and an untrusted server. A client, (e.g., *Alice*), has a set of sensitive documents which she wants to store in a remote server owned by *Bob*. Due to the sensitive nature of the documents, *Alice* does not want *Bob* to learn the content of any of her documents. Since, *Bob* cannot learn the content of the documents, storing the documents in the server in plaintext is unacceptable. Therefore, the document set is stored encrypted using a secure encryption scheme. To facilitate search on encrypted data, an encrypted index structure is stored in the server along with the encrypted data. Authorized users in this setting have access to a trapdoor generation function. Therefore, they can generate valid trapdoors for any arbitrary keyword. This trapdoor is used in the server to search for the intended keyword. It is assumed that the server does not have access to the trapdoor generation function, and therefore, can not ascertain the keyword searched for. However, it is imperative to hide the corresponding keyword of a given query[1] from an adversary. Otherwise, the adversary learns the set of documents that contains the given keyword and the set of documents that does not. In the context of the search over remote encrypted data, it is generally assumed that *Bob* is 'honest but curious', i.e., *Bob* tries to learn as much knowledge as he can without deviating from the protocol.

A 'Searchable Encryption scheme' is qualified as secure, if it satisfies the following necessary conditions.

[1] We use the term 'query' to refer to a trapdoor sent by a user to the remote server as a search request.

1. Query generation function is only known to the authorized data users.

2. The search mechanism works only in conjunction with a valid query generation function.

The first condition specifies that only *Alice* is able to generate a valid query for a given keyword. Furthermore, for a given $\langle keyword, query \rangle$ pair, no entity other than *Alice* has a mechanism to predict the validity of the pair with non-negligible advantage. The second condition guarantees that the search mechanism only works for a given query only if the valid query generation mechanism has been used.

There are many proposed solutions to date that satisfies the conditions outlined earlier. Some models like that of Oblivious ram (e.g., [11]) do not leak any information to the attacker, but are too expensive to be practical on large datasets. Other symmetric key encryption based models, such as [7, 8, 10, 20] proposes efficient solutions to the problem, but leaks data access pattern. That is, in all of those search schemes, given a query x of keyword w, an attacker does not learn w, but she knows which are the documents that contains w. Although, this fact has been noted in the literature (e.g., [8, 10]), none of the previous works systematically analyzed the implications of revealing data access pattern. In this paper, we identify the potential risks of access pattern disclosure via a novel attack. Our empirical analysis with a real world email dataset shows that significant amount of sensitive information can be compromised by the proposed attack. Finally, we propose a simple noise addition technique to mitigate risk against this type of attacks.

We summarize the contributions of this paper as follows.

1. To the best of our knowledge, this is the first study that investigates the implications of data access pattern disclosure in searchable encryption schemes.

2. We formalize a query identity inference attack model based on access pattern disclosure.

3. We prove that such an inference attack is $\mathcal{NP} - complete$ in general and give a heuristic solution that can identify approximately 80% of the queries with minimal background knowledge.

4. Finally, we propose a simple noise addition technique to limit the effect of inference attacks due to access pattern disclosures and empirically show the efficiency of our proposed model.

2. Related Work

In [11], Goldreich et. al. presented their oblivious RAM model, where a user can securely perform search over encrypted data without revealing their access patterns. Although, their technique is theoretically sound, it incurs poly-logarithmic overhead on all parameters. They also proposed a lighter version of the Oblivious RAM protocol which works in 2 rounds, but this protocol comes with a very high square root overhead [8]. Therefore, even this lighter version is computationally very expensive for large datasets. Boneh et. al. proposed a public key encryption based keyword search in [4], where data access pattern is revealed. Later, Boneh et. al. presented a public key solution that can hide access patterns in [5]. Again, these public key encryption based solutions are computationally expensive and not always practical for large datasets.

Song et. al. proposed a special symmetric key based searchable encryption technique in [20]. The authors themselves acknowledged the fact that certain statistical analysis can be performed to successfully leak valuable knowledge over the encrypted documents but did not provide a detailed analysis of such attacks.

In [10], Goh et. al. proposed a security definition to formalize the security requirements of *Searchable Symmetric Encryption* (SSE), and proposed a search technique based on Bloom Filters. Unfortunately, Bloom Filters can lead to false positives, a fact the authors themselves pointed out in [10]. Also, since this technique requires separate indexes for each of the documents, it is not very space (or time) efficient.

At the same time as [10] , Chang et. al. proposed a simulation based definition of SSE in [7]. But, their proposed definition does not take into account how an adversary can generate queries adaptively, i.e. an adversary can choose his subsequent queries after seeing the outcome of all the queries he has submitted so far. This shortcoming have been addressed by Curtmola et. al. in [8]. Furthermore, they also proposed SSE techniques that satisfy their improved definitions. We remark that although their security definition allows adversaries to generate queries adaptively, it does not guarantee hiding of access pattens of any particular user. It has been mentioned in [8, 10] that none of these symmetric key based SSE hides access pattern from the server.

In [17], Kumar et. al. used statistical techniques to compromise the anonymization of query logs using token-based hashing. The authors used a reference query log to infer statistical properties of words in the log-file. They used these properties to process an anonymized query log and tried to invert the underlying hash values to their respective tokens. Although, both their methods and ours employ statistical properties of words, the inference method is quite different. Again, the context of these attacks and even the attack themselves are quite different.

Again, access pattern is a fundamental issue in the context of *Steganographic File Systems*. The concept of a steganographic file system was first proposed by Anderson

et. al. in [1]. The main goal of such a system is to hide the very existence of the files it stores from an attacker so that the users can be protected from compulsion attacks[2]. Xuan et. al. presented a steganographic system in [23] that can hide data access from an attacker who has continuous access to the underlying storage system. Unfortunately, it has been later shown by Troncoso et. al. in [21] that the proposed file system is susceptible to traffic analysis attacks. Although, the attack presented in [21] may seem related to our work, the fundamental objective of these two models is quite different. In [21], the authors proposed a pattern recognition model which tries to identify *block access patterns* in the midst of a seemingly random sequence of block accesses. That is, their attack model tries to identify the block access patterns during the repeated access of a given file. Once a pattern is identified, the model successfully extracts the blocks of that file, and thus proves the existence of that file. Therefore, the scope of this attack model is significantly different from ours. Their attack model tries to discover the data access pattern from a sequence of seemingly random data accesses. Our model, on the other hand, discovers additional sensitive information using the data access patterns. Quite naturally, the attack methodologies are also quite different for these two attack models.

Therefore, to our best of knowledge, none of the previous works in the literature analyzed the potential security risks due to access pattern disclosure in **searchable encryption schemes**. We show that an attacker can gain significant advantage by utilizing access patterns with the help of some background knowledge on the document set. We further elaborate this point in the following section with an example.

A preliminary version of this work has appeared in the $18th$ *ACM Conference on Computer and Communications Security* ($CCS'11$) as a 3-page poster abstract [12]. Compared to the CCS paper, we provide a generalized version of the attack model, a complete mitigation scheme, formal proofs and additional experiments in this final version of the paper.

3. Motivation

Let us assume *Alice* stores a set of sensitive documents regarding *Major League Baseball* to a remote server (e.g., *Bob*) using a searchable encryption technique that leaks data access pattern. Furthermore, an attacker *Mallory* can intercept data access pattern of *Alice* for an arbitrarily unbounded amount of time. Let us also assume that the underlying searchable encryption protocol is safe in the sense that it does not reveal any information about the document contents other than the user access patterns. We argue that *Mallory* can infer some valuable knowledge about query identities using these access patterns with the help of some background knowledge on the keyword distribution of the underlying document set.

Considering the type of the documents stored, let us assume an highly likely scenario, where the words 'New', 'York' and 'Yankees' appear in any given document with a higher probability than any other subset of words of length 3 in the document set. Now, let us assume *Alice* sends search request for these three words interspersed among a set of other keywords in a particular period of time. After all the communication rounds are finished, *Mallory* sees a list of trapdoors and the set of documents they appear in. Quite naturally, *Mallory* can calculate the probability of any two of these queried words appearing together in a document by noticing the number of documents the corresponding trapdoors appear together. Now, by observing these probabilities, *Mallory* can easily isolate the three trapdoors representing the keyword set {'New', 'York', 'Yankees'} because of their high probabilities. Furthermore, it is likely that the pair of words 'New' and 'York' appear together in some documents to refer to the state or the city, not the baseball team. Therefore, the probability of both 'New' and 'York' appearing in a document will be higher than that of 'New' and 'Yankees' or 'York' and 'Yankees'. This important observation enables *Mallory* to uniquely identify the trapdoor for the word 'Yankees'. Furthermore, if *Mallory* learns the identity of the trapdoor for 'New', she will also be able to infer the trapdoor for 'York'. Therefore, it is quite evident that a modicum of prior knowledge can lead to the revelation of many encrypted queries and thus lead to significant document contents disclosure.

The above mentioned trivial motivating example considers the information revealed by only a few number of queries. But, an attacker with significantly large computing power can intercept hundreds of $\langle query, response \rangle$ pairs and combine these access patterns with every piece of information she has about the document contents to launch a very successful attack.

In this paper, we show a practical demonstration of such an attack where an attacker can build a model to identify a significant portion of the queries by combining access patterns with some known background knowledge. Furthermore, we show that conceptually very simple noise addition techniques can successfully thwart such attacks from happening.

4. Simplified Model

To explain the attack concept described in this paper, we present a simple model to simulate 'search over remote encrypted data'. A similar type of model was briefly mentioned in [20]. Before describing the model, we like

[2]In compulsion attack, the users are forced to hand over the decryption keys of an encrypted file.

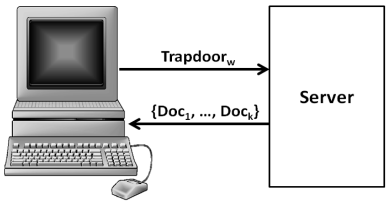

Figure 1. Simplified model of search over remote encrypted data from Mallory's point of view.

to elaborate one very important fact. Our attack model **does not depend on the underlying searchable encryption scheme**. Our attack model succeeds as long as the searchable encryption scheme reveals data access patterns to the attacker. Therefore, the model we describe in the next paragraph is internal to us, and hence should not be interpreted as a necessary condition for the attack to be successful.

In our simplified model, *Alice* has a set of keywords she wishes to search over the document set. At first, *Alice* builds a modified *Inverted Matrix* over the set of documents. That is, *Alice* builds a binary matrix, where the rows are indexed by the keyword set, while the columns are indexed by the document set. Throughout this paper, we refer to this binary matrix for an particular instance of 'search over encrypted data' as *Index Map* (ID). The $(i,j)^{th}$ entry of this Index Map is 1 iff the i^{th} keyword appears in the j^{th} document, and 0 otherwise. In our model, the server *Bob* performs searches for different keywords based on this $Index\ Map$. Since, sending the Index Map in plaintext reveals document contents, *Alice* encrypts the rows of this matrix independently. Furthermore, she also applies a trapdoor function on the set of keywords. Finally, *Alice* sends each of the encrypted rows of the matrix along with the trapdoor value of the corresponding keyword.

When *Alice* wants to search for a particular keyword, she applies the keyword to the trapdoor function and sends *Bob* the result. *Bob* performs a simple text matching and sends *Alice* the encrypted row of the Index Map. *Alice* decrypts the row, and asks for the appropriate set of documents. The search concludes by *Bob* sending back the requested documents.

We like to underscore the statement that we **do not propose a new search scheme** by the above discussion. Almost identical schemes like this one has been outlined by various papers including [20] and [10]. Rather, we simply outline a simple scheme that can simulate the access pattern disclosures of **existing solutions** to the 'search over remote encrypted data' so that we may use this scheme to explain our attack concept.

Now, we can visualize the above scheme as a two step process. *Alice* wants to search a keyword w on an encrypted document set stored in the server. She evaluates $Trapdoor_w$ and sends it to the server. The server sends her the document set that contains the word w. It is worth noting that we purposefully omitted some intermediate communication steps between *Alice* and *Bob*. The rationale behind this omission is, any third party attacker who has access to the communication channel, can omit those intermediate communication steps and see the whole protocol as described in Fig. 1. Furthermore, we argue that any solution to the 'search over encrypted text' that does not hide the access pattern of the client, can be viewed as the protocol shown in Fig. 1. To see why, please note that an attacker, during keyword search process, can record the documents accessed to answer the query. Of course, protocols will have different processes inside the black box, but when viewed from a third party attacker, their interaction with *Alice* will be the same as Fig. 1.

Hence, we assume that *Mallory* intercepts messages of the form $\langle Trapdoor_w, \{Doc_1, \cdots Doc_k\}\rangle$. As we will show next, Malory can infer the underlying keyword w of the query $Trapdoor_w$ that appears in each of the documents in the set $\{Doc_1, \cdots, Doc_k\}$ with the help of some background knowledge.

5. Notations

Table 1 summarizes a short list of notations we use in this paper.

In this paper, we informally treat a document as a set of keywords. So, notations like $x \in d$ are extensively used to mean that keyword x appears in the document d. Furthermore, we assume $\mathcal{D}$ and $\mathcal{K}$ are a total ordering on the document set and keyword set respectively. That is why, we uniquely identify a document by $\mathcal{D}_i$ for $i \in [1,n]$. Similarly, we can uniquely identify a keyword by specifying $\mathcal{K}_i$ for $i \in [1,m]$.

We mathematically denote the i^{th} keyword $\mathcal{K}_i$ as an m bit row vector $[\mathcal{K}_i^1, \mathcal{K}_i^2, ..., \mathcal{K}_i^m]$, such that $\mathcal{K}_i^i = 1$ and $\mathcal{K}_i^j = 0$ for $\forall j \neq i$. Again, we denote $\mathcal{Q} = \langle \mathcal{Q}_1, \cdots \mathcal{Q}_l \rangle$ be the ordered sequence of queries submitted by *Alice* for a given period of time. Here, $\forall i\ \exists j, \mathcal{Q}_i = Trapdoor_{\mathcal{K}_j}$. We refer $R_q = \langle d_1, \cdots, d_n \rangle$ as the result sent by the server in response to a query $q = Trapdoor_{\mathcal{K}_j}$ such that $d_i = 1$ iff the i^{th} document contains the keyword corresponding to the query q and $d_i = 0$ otherwise. That is, if $R_q = \langle d_1, \cdots, d_n \rangle$ be the response for query q and

Table 1. Notations

Notation	Meaning
$\|x\|$	The cardinality of the set x.
x^T	The transpose of the matrix x.
$\mathcal{D}$	An ordered sequence of all the documents stored in the server.
n	The number of documents in $\mathcal{D}$. That is, $n = \|\mathcal{D}\|$.
$\mathcal{K}$	An ordered sequence of all the keywords.
m	The number of keywords in $\mathcal{K}$. That is, $m = \|\mathcal{K}\|$.
$\mathcal{D}_i$	The i^{th} document in $\mathcal{D}$.
$\mathcal{K}_i$	The i^{th} keyword in $\mathcal{K}$.
$Trapdoor_w$	The output of the trapdoor function with argument $w \in \mathcal{K}$.
R_q	The result sent by the server in response to a query q.
$\mathcal{Q}$	A sequence of queries in a given time.

$q = Trapdoor_{\mathcal{K}_j}$, then each d_i is defined by the following function.

$$d_i = \begin{cases} 0 & : \text{if } \mathcal{K}_j \notin Doc_i \\ 1 & : \text{if } \mathcal{K}_j \in Doc_i \end{cases}$$

Since, $\mathcal{Q} = \langle \mathcal{Q}_1, \cdots \mathcal{Q}_l \rangle$ is the sequence of queries the client poses to the server at some given interval, the server sends the corresponding documents as the query answer in response to each of these queries. It should be noted that in our settings, the attacker can intercept the queries $\mathcal{Q}_i$, can uniquely identify them, but do not know which keywords they are associated with. Similarly, he/she can intercept the documents, but is not able to learn the contents of the documents.

6. Threat Model

In our model, the attacker, *Mallory* observes a sequence of l queries $\mathcal{Q} = \langle \mathcal{Q}_1, \cdots \mathcal{Q}_l \rangle$ submitted by the client to the server. Naturally, *Mallory* has also access to the sequence of query responses $R_{\mathcal{Q}_i}$, $\forall i \in [1, l]$, since he has full access to the communication channel. The goal of *Mallory* is to successfully ascertain the sequence of keywords $\mathcal{K}_A = \langle \mathcal{K}_{a_1}, \cdots \mathcal{K}_{a_l} \rangle$, where $\mathcal{K}_A \subset \mathcal{K}$ and $\forall i \in [1, l]$, $Trapdoor_{\mathcal{K}_{a_i}} = \mathcal{Q}_i$. That is, *Mallory* wishes to uniquely identify the underlying keywords of each of the queries $\mathcal{Q}_i$. In this paper, we show that *Mallory* has a very high probability of succeeding if she has access to the following background knowledge on the document set.

- *Mallory* knows the underlying keywords for k of the queries in the sequence $\mathcal{Q}$. That is, *Mallory* has access to the set $K_Q \subset \mathcal{K}_A \times \mathcal{Q}$, where $K_Q = \{\langle x, y \rangle \,|(x \in \mathcal{K}_A) \wedge (y \in \mathcal{Q}) \wedge (y = Trapdoor_x)\}$ and $k = |K_Q|$. We later show that *Mallory* can be successful with very high probability even when $k << l$. Furthermore, we empirically show that *Mallory* has high probability of being successful even when she does not know the identity of any of the queries in $\mathcal{Q}$, i.e., $k = 0$.
- *Mallory* has an $m \times m$ matrix M. Please recall that m is the number of possible keywords in our model. Each $(i, j)^{th}$ cell in matrix M contains the expected probability of both i^{th} and j^{th} keywords appearing in any random document $d \in \mathcal{D}$. That is, $M_{i,j} = Pr\left[(\mathcal{K}_i \in d) \wedge (\mathcal{K}_j \in d)\right]$, where d is a document sampled uniformly from the set $\mathcal{D}$.

An attacker can simulate the matrix M by carrying out a probabilistic analysis over the large publicly available online datasets. Sophisticated frequency analysis techniques can also be used to approximate M. Again, an inside attacker may already have access to a sizable subset of the original dataset in his/her disposal. Therefore, he can create the background matrix from this subset. In this case, there is a significant probability that this matrix is a reasonable approximation of the background matrix M our model requires. We later empirically show that our model works reasonably well for a noisy background knowledge matrix. Therefore, we argue that the background matrix M does not have to be completely accurate. Rather, any close approximation of M is sufficient for an successful attack. For example, there are quite a substantial number of `WikiLeaks` documents in public domain now. An adversary can use a related subset of documents to come up with a matrix that can approximate a background matrix M for a dataset that involves dimplomatic affairs. We plan to explore the ways to build the background matrix and their effects on our model in further details in the future.

Again, obtaining a set of known queries might prove to be a difficult task for an attacker under some scenario. But, we still find it appropriate to add it as a background knowledge in our model for the following reasons. First, we empirically show that our model performs quite well even when the adversary does not have access to a set of known queries. Therefore, an attacker only uses this known

query set when he has access to it. He still does quite well when he does not have access to this set. Second, under some scenario, it is quite possible for an inside attacker to know a known query set. And finally, we believe that any secure searchable encryption scheme should not reveal any additional query content given a set of known queries. The Oblivious RAM [11] is one such secure protocol. Therefore, we find it justified to consider the set of known queries as a part of the background knowledge the adversary has in our model.

To best explain our proposed model, we introduce the attack concept with a simplified but effective version of the model in the following section. Later, we proceed to present a more generalized attack model in the subsequent section of this paper.

7. Simplified Attack Model

In this section, we describe how *Mallory* can formulate an attack to identify query identities as an optimization problem with constraints. Here, *Mallory* has access to the communication channel, and thus intercepts a set of l queries $\mathcal{Q} = \langle \mathcal{Q}_1, \cdots, \mathcal{Q}_l \rangle$ and has access to K_Q and M as defined earlier. Let us assume that *Mallory* already knows the corresponding keywords for each of the queries in the set $\mathcal{S} \subset \mathcal{Q}$. That is, $\mathcal{S} = \{y | \exists x \, \langle x, y \rangle \in K_Q\}$. Now, the goal for *Mallory* is to assign keywords to all the queries $q \in (\mathcal{Q} - \mathcal{S})$ such that the joint keyword distribution as seen by the message responses R_q fits our prior knowledge, namely the background knowledge matrix, M.

Hence, *Mallory* tries to find a sequence of l indices $\langle a_1, \cdots, a_l \rangle$ s.t. *Mallory* believes that $\forall j : \mathcal{Q}_j = Trapdoor_{\mathcal{K}_{a_j}}$, given the matrix M. We present the simplified attack model as an optimization problem by Eq. 1.

$$\underset{\langle a_1, \cdots, a_l \rangle}{\operatorname{argmin}} \sum_{\mathcal{Q}_i, \mathcal{Q}_j \in \mathcal{Q}} \left(\frac{R_{\mathcal{Q}_i} \cdot R_{\mathcal{Q}_j}^T}{n} - \left(\mathcal{K}_{a_i} \cdot M \cdot \mathcal{K}_{a_j}^T \right) \right)^2 \tag{1}$$

$$Constraints: \quad \forall j \text{ s.t. } \mathcal{Q}_j \in \mathcal{S}, a_j = x_j \text{ s.t. } \langle \mathcal{K}_{x_j}, \mathcal{Q}_j \rangle \in K_Q$$
$$\forall j, \parallel \mathcal{Q}_j \parallel = 1$$

The first constraint in $Eq.$ (1) guarantees that the known queries will be assigned to their correct known keywords. While the second one makes sure that all the queries in the set has an assignment of a valid keyword format, i.e., each query $\mathcal{Q}_j$ conforms to the query format outlined in section 5.

The result of this constraint satisfying optimization problem is an assignment of keywords to the queries that achieves minimum distance from our background knowledge matrix M.

To explain the model described in $Eq.$ (1), let us consider the following example. Suppose, $\mathcal{Q}_s$ and $\mathcal{Q}_t$ are two encrypted queries. Therefore, *Mallory* can calculate the probability of the underlying keywords appearing together in a given document by $\beta = \frac{R_{\mathcal{Q}_s} \cdot R_{\mathcal{Q}_t}}{n}$, here $\cdot$ operation denotes the "dot" product between two vectors. Now, for any two given keywords $\mathcal{K}_f$ and $\mathcal{K}_g$, *Mallory* can calculate the probability of these two keyword appearing together by $\gamma = M_{f,g}$. When, the keywords K_f and K_g are presented by their proper bit-vector representation, γ can be equivalently written as $\gamma = (\mathcal{K}_f \cdot M \cdot \mathcal{K}_g^T)$. Here, $\cdot$ operation denotes matrix multiplication. Naturally, *Mallory* will assign $\mathcal{K}_f$, $\mathcal{K}_g$ to the queries $\mathcal{Q}_s$, $\mathcal{Q}_t$ iff the observed probability from the query response β is close to the known probability γ. This closeness can be measured by a simple arithmetic distance function $(\beta - \gamma)^2$, where a lower value of this function is preferred over a higher value. So, the aim of *Mallory* will be to assign keywords to queries such that this distance function is minimized. Our model given by $Eq.$ (1) is just a formalization of this objective function.

8. Generalized Model

In this section, we outline the generalized version of the proposed attack model. Here, instead of taking the joint probability distribution of keyword pairs, we consider joint probability distributions up to r keywords. In this model, instead of the background matrix M, *Mallory* has access to a family of r functions $F = \{F_i\}$. The i^{th} function in the family F, denoted as $F_i : \mathcal{K}^i \rightarrow [0, 1]$ takes i keywords as arguments and returns the probability of all of these i keywords appearing in a document d sampled uniformly from the document set $\mathcal{D}$. That is, F_i can be defined by the following equation.

$$F_i(\mathcal{K}_1, \cdots, \mathcal{K}_i) = Pr\left[(\mathcal{K}_1 \in d) \wedge \cdots \wedge (\mathcal{K}_i \in d) \right] \tag{2}$$

Before we present our generalized attack model, let us define a crude version of "*dot*" product for p equal length vectors[3].

Definition The MSP (*M*ulti *S*calar *P*roduct) function, denoted by $\odot$, takes as argument p equal length binary vectors $V_1, \cdots, V_p$ and returns a positive integer. Let us assume that each vector V_i is a q bit vector s.t. $V_i = \langle V_i^1 \cdots V_i^q \rangle$. Then, $\odot$ is defined as follows.

$$\odot(V_1, \cdots V_p) = \sum_{i=1}^{q} \prod_{j=1}^{p} (V_j^i) \tag{3}$$

[3]The value of p can be arbitrarily greater than 2.

It should be noted that the MSP function defined above degenerates to the conventional "*dot*" product operation for the limiting case of $p = 2$.

Also in this generalized model, *Mallory* aims to find a sequence of l indices $\langle a_1, \cdots, a_l \rangle$ s.t. *Mallory* believes that $\forall j : Q_j = Trapdoor_{\mathcal{K}_{a_j}}$, given the function family F.

Let us define c_i for $i \in [1, k]$ for a given sequence of indices $\langle a_1, \cdots, a_l \rangle$ in the following way.

$$c_i = \sum_{\mathcal{Q}_1, \cdots \mathcal{Q}_i \in \mathcal{Q}} \left(\frac{\odot(R_{\mathcal{Q}_1} \cdots R_{\mathcal{Q}_i})}{n} - (F_i(\mathcal{K}_{a_1}, \cdots, \mathcal{K}_{ai})) \right)^2 \tag{4}$$

Here, $\odot(R_{\mathcal{Q}_1} \cdots R_{\mathcal{Q}_i})$ returns the number of documents where all of the underlying keywords corresponding to the query response sequence $R_{\mathcal{Q}_1} \cdots R_{\mathcal{Q}_i}$ appear in. Again, let us define $w = (w_1, w_2 ..., w_k)$ be a real-valued weight vector. Our model can be expressed in terms of $Eq.\ (4)$ in the following way.

$$\underset{\langle a_1, \cdots, a_l \rangle}{\operatorname{argmin}} \sum_{i=1}^{k} w_i \cdot c_i \tag{5}$$

Subject to: $\forall_i \forall j$ s.t. $\mathcal{Q}_j \in \mathcal{S}, a_j = x_j$ s.t. $\langle \mathcal{K}_{x_j}, \mathcal{Q}_j \rangle \in K_Q$

$$\forall_i \forall_j, \parallel \mathcal{Q}_j \parallel = 1$$

Unfortunately, the optimization problem presented in our generalized model is $\mathcal{NP} - Complete$. In fact, we show in $Theorem\ 1$ that even a smaller subset of the given problem, i.e., the optimization problem given for the simplified model in $Eq.\ (1)$, is $\mathcal{NP} - Complete$.

Theorem 1. *Finding an optimal assignment of keywords to a given set of queries w.r.t. the objective function defined by Eq. (1) is $\mathcal{NP} - Complete$.*

For interested readers, the proof of $Theorem\ 1$ is presented in Appendix A.

Therefore, it is quite evident that solving the optimization problem outlined earlier is computationally infeasible for large dataset. Quite naturally, we propose an efficient approximation of this problem that uses *Simulated Annealing* [15]. Simulated Annealing is a probabilistic metaheuristic which is frequently used to find good approximation to the global optimum of some given function when the search space is large. It turns out that if we run the optimization problem long enough, a significant subset of the query set $\mathcal{Q}$ can be successfully revealed. A detailed description of the algorithm of our optimizer for the simplified attack model is given in Fig. 2 at page 8.

9. Experiment Results

In this section, we present an empirical evaluation of our proposed simplified attack model on various instances of 'search over remote encrypted data'. In our experiments, we use a real world dataset, namely, the Enron dataset as our corpus [16]. We also use a well known stemming technique such as *Porter Stemming Algorithm* [19] to ascertain the root of every keyword. Our experimental results suggest that even the simplified model can predict the identity of over 80% queries for large datasets. On the other hand, the execution time of the implemented model suggests that the proposed approximation of the model is quite efficient. Rest of this section is organized as follows. We describe our methodology first, then describe various experiments and explain their results.

9.1. Experimental Setup

9.1.1 Dataset Used

As we have already mentioned, we use *Enron* dataset as our corpus [16]. This dataset was originally prepared by the CALO Project (A $\underline{C}$ognitive $\underline{A}$ssistant that $\underline{L}$earns and $\underline{O}$rganizes). This dataset has already been extensively used in various study in the past [3, 6, 9]. *Enron* dataset contains emails of about 150 different users. The dataset is organized into various folders. We have chosen all the 30109 emails contained in the $\_sent\_mail$ folder of all the users as our experimental dataset. This dataset was made public by the Federal Energy Regulatory Commission during its investigation.

The most important characteristic of this dataset is that each of the documents is a real email sent by a person. One of the motivations [20] behind 'search over encrypted data' is to store emails to a remote server in encrypted format and search the emails from time to time. Therefore, we find it appropriate to run our experiments on real email dataset like that of enron. The first few lines for each of the emails contains some metadata about that mail. Since, these lines are not part of the original email, we strip these lines off the email documents as a preprocessing step.

9.1.2 Stemming Algorithm

Since, most of the emails are personal in nature, they capture informal communication made between two individuals. Therefore, we use a stemming algorithm, namely the *Porter Stemming Algorithm* [19] to find the root of each of the words in the document set. Throughout this section, it will be implicitly assumed that all the processing step on the keywords have been done after the keywords have gone through the porter stemming algorithm. It should

Algorithm 1 Optimizer

Require: V : variable List
D : domain List
K : known assignments
Mp, Mc : pair similarity matrices

V and D contains the cipher and plain keywords respectively that are not part of the background.

K contains the background knowledge in the form of known cipher-plain keyword pairs.

Mp, Mc contains the joint frequency distrubition of plain and cipher keyword pairs respectively.

```
valList ← copy D
for all cipher var_i ∈ V do
    val_i ← select random member of valList
    add { vari = vali} to initState
    remove val_i fom valList
end for
add K to initState
return ANNEAL (initState, D,  Mp, Mc)
```

Any state of the problem contains cipher-plain keyword pairs such that each cipher keyword is assigned to a unique plain keyword. Initial state is constructed by the union of mappings in backgrund and random mapping between cipher and plain keywords that are not part of the background.

Simulated annealing is applied to find an approximate solution to the formulated optimization problem.

Algorithm 2 ANNEAL

Require: initState, D, Mp, Mc
initTempareture, coolingRate. rejectThreshold

Simulated annealing parameters

```
currentState ← initState
succReject  ← 0
currT ←  initTempareture
while  (temp ≠ 0  and succReject < rejectThreshold) do
    currentCost ← 0, nextCost ← 0
    nextState ← copy currentState
    (x, y)  ← select random pair from nextState
    y' ← select random member of D different from y
    remove {x = y} from nextState
    add {x = y'} to nextState
     if (z,y') is a member of initState then
        remove {z = y'} from nextState
        add {z = y} to nextState
    endif
    for all cells i,j in Mc
        (i,k) ← currentState.get(i), (i,k') ← nextState.get(i)
        (j, l) ← currentState.get(j), (j, l') ← nextState.get(j)
        currentcost += (Mc[i,j] - Mp[k,l])^2
        nextcost += (Mc[i,j] - Mp[k',l'])^2
    end for
    E =nextCost – currentCost
    if (E < 0) then
        accept new state
    else
        accept new state with probability exp(-E/currT)
    end if
    if new state is accepted then
      succReject ← 0, currentState ← nextState
   else
      succReject++
    end if
    currT= coolingRate*currT
end while
return currentState
```

Search continues until tempreature reaches zero or the system is frozen (i.e. no new state is being accepted for a large period of time).

Next state is generated by choosing a random cipher-plain keyword pair from the current state and assigning a new plain keyword to the selected cipher keyword

If plain keyword y' that is assigned to selected cipher x is previously assigned to another cipher z, old plain keyword y that was assigned to x is assigned to z

Cost of the current and neighbor states are evaluated against the formulated evaluation function. state.get(i) returns the cipher,plain pair corresponding to cipher i. k,k' are the plain keywords assingned to cipher i in current and next states respectively. l,l' are plain keywords for cipher j in current and next states

Standard step for simulated annealing approaches. If next state is better than the current, accept new state. Even if new state is worse than the current state, accept new state with some probability to get rid of local minimums

Decrement temparature according to exponential cooling scheme, coolingRate is a constant close to but smaller than 1

Figure 2. Algorithm for Optimizer.

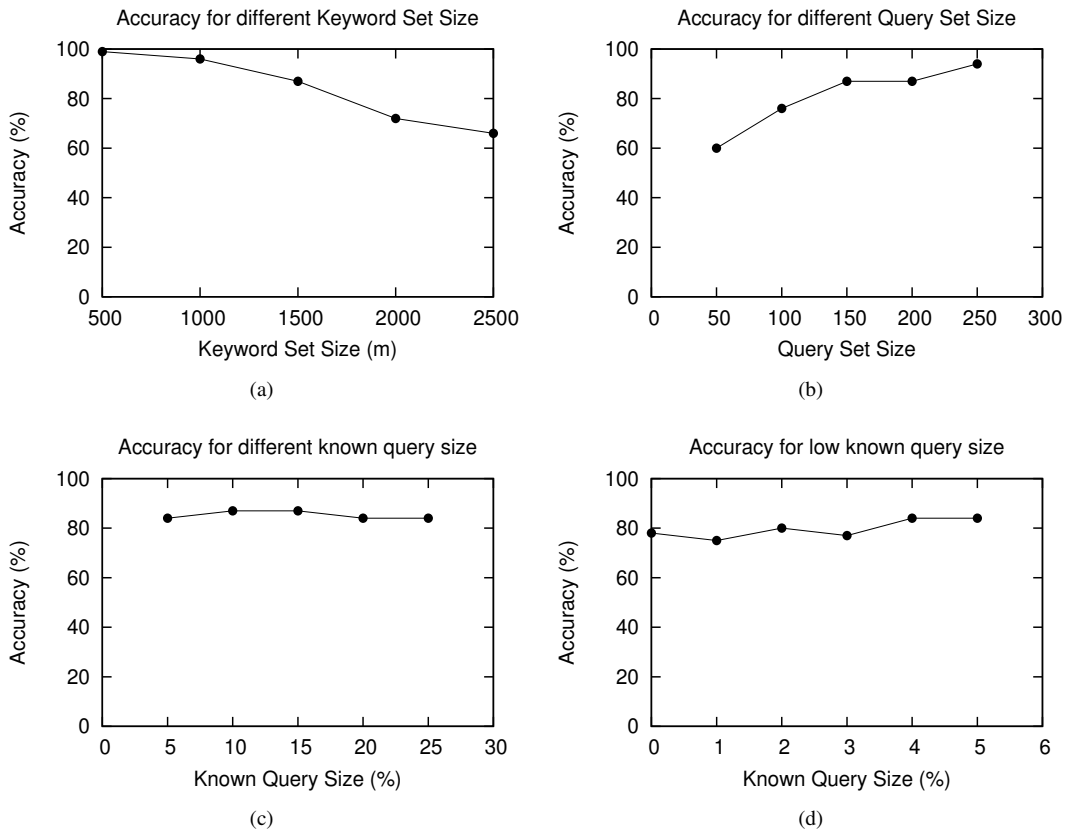

Figure 3. Accuracy for various (a) Keyword Set Size. (b) Query Set Size. (c) Known Query Set Size. (d) Low Known Query Set Size.

be noted that we have used an implementation of Porter's Stemming [19] written by its author Martin Porter.

9.1.3 Keyword Generation

Our whole corpus of 30109 documents contains 77000 unique keywords after getting rid of most common 200 words like 'the', 'a', 'from' etc. We sort these keywords based on the decreasing number of their occurrences and always use the first x number of words as our keywords. We refer to this x as the keyword set size. It should be noted that the value of x differs from experiment to experiment. But, unless noted otherwise, the keyword set size are always chosen as the most frequent x words out of the 77000 unique words.

9.1.4 Query Generation

Query patterns of individual users are expected to vary quite significantly. Therefore, it is hard to adopt a methodology that is able to capture the idiosyncratic behaviors of all the users. To make matter worse, we did not find any real-world query set on the *Enron* dataset. Therefore, we use *Zipfian Distribution* to create synthetic query sets from our keyword set. Fortunately, our attack scheme does not use query frequency. Therefore, our attack can be equally applicable to any other query distribution. Furthermore, we suppress query repetition from the attack model to undermine the effects of our query generation process on the model accuracy. That is, we generate the query set in such a way that no keyword is chosen as a query more than once. It should be noted that such suppression does not limit the applicability of our model in anyway. This is because, an attacker can always identify the duplicate queries and discard them before applying to the attack model.

To generate the query set, we sort the keyword set in non-increasing order of occurrences. If a particular keyword appears in the j^{th} position in this list, we call the rank of this keyword j. According to the *Zipfian distribution*, the probability of this keyword being chosen as query (Pr_j) is given by the following[4].

[4]If the keyword chosen is already in the query set, we discard it to suppress query repetition.

$$Pr_j = \frac{\frac{1}{j}}{N_x} = \frac{1}{j \times N_x}.$$

Here, the denominator $N_x = \sum_{i=1}^{x} \frac{1}{j}$ is the normalization factor. According to the Zipf's law, in a corpus of natural language utterances, the frequency of appearance of an individual word is inversely proportional to its rank [24]. Since, we run our experiments on natural language corpus, we argue that the queries might follow zipfian distribution. But, we like to stress that our attack scheme **does not use query frequency**. Therefore, our attack can be applicable to **any other query distribution**.

9.1.5 Execution Time

We use a serial implementation of our model for all the experiments mentioned in this paper. This implementation took no more than 14 hours to finish for any of the experiments in a AMD Phenom $II\ X6\ 1045T$ Windows 7 machine clocked at $2.70\ GHz$ with 8 GB of system memory. Our model is supposed to run even faster if we allow parallel computation. Therefore, we argue that the attack model we present in this paper is quite efficient.

9.2. Experiment & Results

9.2.1 Accuracy over varying Keyword Set Size

Fig. $3(a)$ in *page* 9 shows the accuracy (%) for various keyword set size. Here, we have chosen the keyword set size to be multiples of 500 with the maximum value of 2500. Keywords were generated according to the *Keyword Generation* step described earlier. The number of queries for this experiment was set at 150 and 15% of the queries were known beforehand. It is quite evident that for relatively smaller keyword set size, the attack model can correctly identify almost all the queries. But, as the keyword set size increases, the identification accuracy is decreased. But, even for keyword set size of 2500, the model can successfully identify most of the queries.

9.2.2 Accuracy over varying Query Set Size

Fig. $3(b)$ in *page* 9 shows the accuracy (%) for various query set size. Here, we have chosen query set size to the multiples of 50 with the maximum being 250. Queries were generated using the *Zipifian* distribution as described earlier. Our model works fairly well for a small query set size of 50. But, as the query set size goes up, the accuracy of the model goes higher as well. This is due to the fact that higher query set size indicates that higher portion of the background matrix M is used to predict the query identity.

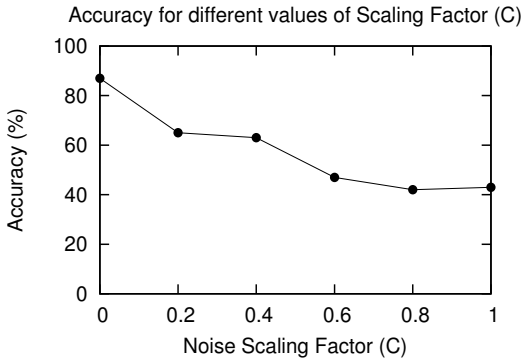

Figure 4. Accuracy (%) for different Noise Scaling Factor C.

9.2.3 Accuracy for varying Known Query (%)

In this experiment, we have run experiments for varying known query (% of Query set size). We start off with 5% known query and increase it upto 25%. It is quite apparent from Fig. $3\text{-}(c)$ in *page* 9 that increasing the known query size does not improve accuracy that much. Rather, the accuracy measurement are almost similar for different known query sizes. Fig. $3\text{-}(d)$ in *page* 9 shows the accuracy for very low known query percentage. We can see that the model can successfully identify nearly 80% of the queries correctly even if there are no known queries. This proves that the model is quite useful even without the knowledge of known queries. For this experiment, we fix the keyword set size to 1500 and the query set size to 150.

9.2.4 Accuracy for Noisy Matrix

All the experiments described up to this point has an implicit assumption that the attacker has access to a perfect background matrix M. But, getting a perfect matrix M may proved to be difficult under some scenario, even impossible under others. That's why, we have investigated the accuracy of our model under a noisy matrix M. Let, $\sigma^2 = Var\{M_{i,j}\}$. That is, σ^2 is the variance of the set of elements in the matrix M. We modeled the noise for the elements of the matrix M as a $Normal\ Distribution$ with mean 0 and variance $C \times \sigma^2$. Here, C is a constant, which we refer to as 'noise scaling factor'. That is, the noise will increase as C increases. We added noise to the element of the matrix M according to the distribution $\mathcal{N}(0, C \cdot \sigma^2)$. For this experiment, we fixed our keyword set size to be 1500, x, the keyword set size is chosen to be 1500 for this experiment and 15% of the queries were assumed to be known beforehand.

Query set size to be 150 and (%) known query to be 15%. We varied noise scaling factor C and present the result in Fig. 4. It can be deduced from Fig. 4 that our model works fairly well for low values of C. Even when the noise scaling factor is 1.0, our model predicts a reasonable number of query identities accurately.

10. Preventing Inference Attack

In this section, we develop a framework to thwart 'Inference Attacks' described earlier. Our proposed framework aims to make query responses as similar as possible at the expense of a few false positives. We assume, even a very small false negative rate is unacceptable in the context of search over remote encrypted text. But, a few false positives, on the other hand, is acceptable in the sense that the client can easily detect these false positives and discard them upon decrypting the documents.

It should be noted that our proposed mitigation scheme only aims to thwart the inference type of attacks described earlier. It does not guarantee the overall security of a searchable encryption scheme. For example, even with our mitigation scheme in place, an attacker may still be able to extract some sensitive information by applying a different attack model that uses query frequencies and some other useful domain knowledge. Guarantee of the overall security requires rigorous study of all the vulnerable aspects of a searchable encryption schemes. We leave such a study as a possible future work. Therefore, in the following sections, we propose a mitigation scheme that provides security only against the inference attack described earlier in this paper.

10.1. Our Framework

To explain our framework, we like to use the simplified model we have described in *Section* 4. But, it worths noting that the framework, does not depend on the underlying model in anyway. That is, our concept can be **easily applied to any searchable encryption scheme**.

In our simplified model, we assume that the server holds a bitmap for each of the query words. Of course, these bitmaps are stored encrypted for the sake of privacy. Once a query is serviced by a server, the client decrypts the bitmap and request the relevant documents. Thus, the server has a mechanism to relate a particular encrypted keyword and the list of documents that contain the keyword. Our approach formulates a mechanism to add fake documents in the bitmaps so as to prevent attacks described above by the server, or any third party eavesdropper.

Let us assume $ID \in m \times \{0,1\}^n$ be an index matrix s.t. each $(i,j)^{th}$ entry $ID_{i,j}$ is defined in the following way.

$$ID_{i,j} = \begin{cases} 0 & : \text{if } \mathcal{K}_i \notin \mathcal{D}_j \\ 1 & : \text{if } \mathcal{K}_i \in \mathcal{D}_j \end{cases}$$

Here, $\mathcal{K}_i$ is the i^{th} keyword and $\mathcal{D}_j$ is the j^{th} document. Again, we denote the i^{th} row of the matrix ID by ID_i. Given two bit strings A and B of same length k, we define the function $d_H : \{0,1\}^k \times \{0,1\}^k \rightarrow \mathcal{N}$ to be the function that returns the $Hamming\ Distance$ between its arguments.

11. Privacy Definition

Definition $(\alpha, t) - secure\ index$. We say an $m \times n$ binary index matrix ID is $(\alpha, t) - secure$ for a given $\alpha \in [1, m]$ and $t \in \mathcal{N}$, if there exists a set of partitions $\{S_i\}$ s.t. the following holds.

1. The partition set is complete. That is, $\bigcup_i S_i = [1, m]$.
2. The partitions in the set are non-overlapping. That is, $\forall i, j,\ S_i \cap S_j = \emptyset$.
3. Each partition has at least α rows. That is, $\forall j,\ |S_j| \geq \alpha$.
4. Finally, the hamming distance between any two rows of a given partition j is at most t. That is, $\forall i_1, i_2 \in S_j$, $d_H(ID_{i_1}, ID_{i_2}) \leq t$.

It should be noted that an $m \times n$ matrix ID is $(m, 0) - secure$ if all the rows of the matrix ID is identical to each other. On the other extreme, any matrix ID is $(\alpha, n) - secure$ for any value of $\alpha \leq m$. Informally speaking, an $(\alpha, 0) - secure$ index matrix guarantees that for each keyword, there are at least $\alpha - 1$ keywords which appear exactly in the same set of documents. Therefore, it's hard for an attacker to distinguish a keyword given the query response of that particular keyword.

Theorem 2. *Let ID be an $m \times n$ $(\alpha, 0) - secure$ index for some $0 < \alpha \leq m$. Given the response of a query $q_i = Trapdoor_w$ for some keyword w and the complete ID matrix, an attacker can successfully identify the keyword w with probability at most $\frac{1}{\alpha}$ by just using background information related to ID matrix.*

Proof The attacker has the complete ID matrix and the query response of q_i. Now, the query response of q_i is a row of ID matrix. Let's assume, without loss of generality, the response of q_i is the j^{th} row of the ID matrix for some unique j s.t. $1 \leq j \leq m$. Since, the matrix ID is $(\alpha, 0) - secure$, there are at least $\alpha - 1$ rows of ID which are exactly identical to each other. Hence, the attacker can find this set S of at least α rows that are exactly similar to ID_j. Since, we assume that our trapdoor function is secure,

the attacker can at best select an element from the set S randomly. Therefore, the attacker can successfully identify w with probability at most $\frac{1}{\alpha}$. ∎

$Theorem$ 2 states that an attacker can successfully identify a keyword from a query response with probability at most $\frac{1}{\alpha}$ even in the unlikely case of having full access to the index matrix. Here, we assume that the query repetition is visible to the attacker. Naturally, the attacker has significantly smaller chance of success when he does not have full access to the complete index matrix, rather has access to some partial information, for example, the background knowledge matrix used in previous sections. Therefore, we argue that making an index matrix $(\alpha, 0)-secure$ can make keyword frequency based attack significantly harder.

12. Securing Index Matrix

In this section, we outline our approach to transform a given $m \times n$ index matrix into a $(\alpha, 0) - secure$ matrix. It should be noted that an existing index matrix may not be modified by putting $0's$ in place of $1's$ for any row indexed by a keyword w. Otherwise, search results for this keyword w will be incomplete. That is, the resultset will not contain some documents in which the keyword belongs to, namely, the documents corresponding to those $1's$ which has been made $0's$. Furthermore, the user may well be unaware of these documents. Therefore, we assume that $false\ negatives$ are not allowed, which in turn prohibits replacing $1's$ with $0's$ in any row of the matrix. On the other hand, replacing $0's$ with $1's$ may induce some false positives in the resultset, which can be detected later by the user upon decrypting the documents. Therefore, the user can safely discard these $false\ positives$. Hence, introducing some false positives in the resultset will only affect performance, not accuracy.

Definition The function $t_\alpha : \{0,1\}^{m \times n} \rightarrow \{0,1\}^{m \times n}$ takes an $m \times n$ binary index matrix ID and returns a $(\alpha, t) - secure$ matrix ID' s.t. the function $(\sum_{i=1}^{m} w_i \cdot d_H(ID_i, ID'_i))$ is minimized for a given weight vector w, and if $\forall i, j$, $(ID_{i,j} = 1) \Longrightarrow (ID'_{i,j} = 1)$ holds.

It should be noted that in the process of making rows of the index matrix similar, we are adding false positives. This will result in increased communication overhead as well as processing overhead. The function t_α is defined in a way so that it converts a given ID matrix to the closest $(\alpha, t) - secure$ matrix. We define the cost of an transformation as follows.

Definition If an $m \times n$ index matrix ID is transformed into an $m \times n$ $(\alpha, 0) - secure$ index matrix ID' by applying the function t_α, the cost of such a transformation is denoted by the function c_t, defined as follows.

$$c_t(ID, ID') = \sum_{i=1}^{m} d_H(ID_i, ID_i').$$

Informally speaking, the function c_t returns the total number of elements $ID_{i,j}$ where a 0 bit has been changed to a 1 bit to obtain the matrix ID'. Thus, minimizing the c_t function will ultimately result in a less number of false positives to be injected into the $(\alpha, 0) - secure$ matrix ID'. It can be easily showed that t_α function converts a given ID matrix to a $(\alpha, 0) - secure$ matrix s.t. the cost function c_t is minimized.

Based on this observation, we can convert a given index matrix ID to a $(\alpha, 0) - secure$ matrix ID' by applying the t_α function on the matrix ID.

The main hurdle of finding an optimal conversion of a given matrix to a $(\alpha, 0) - secure$ matrix is that finding an optimal partitioning of rows is hard. It turns out, the problem of finding such optimal partitioning belongs to $\mathcal{NP} - hard$ [13]. Therefore, we propose an approximation algorithm given in $Algorithm$ 1 to convert an index matrix ID to a $(\alpha, 0) - secure$ index matrix.

Although, $Algorithm$ 1 returns a $(\alpha, 0) - secure$ index matrix, the c_t measure is not optimally minimized. But, since the optimal reduction of c_t is computationally infeasible, $Algorithm$ 1 is a very good approximation of the optimal conversion. To test the efficiency of our approach, we have implemented $Algorithm$ 1 and present our results in the following section. We use agglomerative hierarchical clustering algorithm [2] with average distance. Furthermore, we use $cosine$ distance as our distance measure.

Input : A index matrix ID
Output: A $(\alpha, 0) - secure$ index matrix ID'

Run clustering algorithm on the rows of ID to get p clusters.;
while *Any cluster C_p have less than α elements* **do**
  combine the cluster C_p to the nearest cluster.
end
for *each of the cluster C_p* **do**
  Feed the set of elements of C_p to the t_α function to obtain a new set of elements S ;
  Replace the set C_p by S in ID ;
end
return ID;

Algorithm 1: Approximation algorithm to transform a index matrix into a $(\alpha, 0) - secure$ matrix

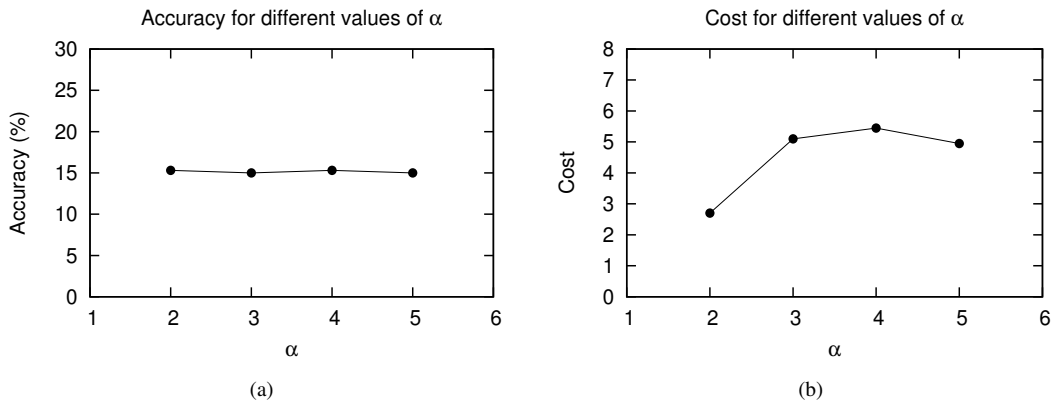

Figure 5. Accuracy and cost for different α values.

13. Experiment & Results

In this section we present the efficiency of our approach of evading attack model described in the earlier sections for different values of α. We run the experiments for $\alpha \in \{2, 3, 4, 5\}$. Fig. $5(a)$ shows the accuracy for different values of α. For these experiments, we have fixed the keyword set size to be 1500, query set size to be 150 and the $(\%)$ known query to be 15% (i.e., best possible scenario for attacker as identified by previous experimental results). Furthermore, we assume the background matrix M to be perfect, that is, no element of M contains any noise (again, this is optimal from attackers point of view). From Fig. 5-(a), it is evident that the accuracy is almost equal to that of the $(\%)$ known query. That is, the attacker failed to successfully identify queries any other than the known queries. Fig. 5-(b) presents the increased cost because of the converted ID' matrix. We define cost as the ratio of increase in number of documents retrieved by the new ID matrix to the number of documents retrieved by the old matrix. That is, Let, $p =$ (number of documents returned for the old matrix ID), and $q =$ (number of documents returned for the new matrix ID'). Then, the cost is defined as, $cost = \frac{q-p}{p}$. It is apparent from Fig. 5-(b), that the percent increase in the cost is increased as the value of α is goes up.

To compare the cost of our proposed mitigation strategy with ORAM based solutions, we analyzed the ORAM construction given in [22]. To our knowledge, the ORAM solution proposed by Williams et. al. in [22] is one of the most efficient ORAM constructions available in the literature.

The protocol in [22] has a computational overhead of $O(\log n \log\log n)$ per request for n data items. This overhead includes the constant factor of $1.44c$ for an allowed error probability of 2^{-c} along [5] with other constant factors in the big O [18]. Based on this analysis, we can accurately approximate the value of the constant associated with this overhead to be around 100. Now, let us assume the number of data items to be 1024 or (2^{10}). That is, $n = 1024$. Therefore, $\log n = 10$. Then, even the most efficient ORAM presented in [22] requires more than $100 \times 10 = 1000$ data items to be retrieved for a single data access. Furthermore, this overhead grows much larger when the number of data items (n) increases. In our approach, on the other hand, the required overhead is only $3 - 5$ accesses per request and does not change as n increases. Therefore, our approach is more efficient than the ORAM constructions. But, unlike our case, ORAM approach hides all the access pattern disclosure.

Therefore, it is evident from the above discussion that adding noise to the index matrix is quite efficient in thwarting *Inference Attacks*. But, this approach incurs some additional computational overhead. However, this overhead is still quite negligible when compared to that of the most efficient ORAM constructions. Therefore, we argue that noise-addition based mitigation scheme is quite useful where efficiency is a major concern. However, if an user is only concerned about privacy, efficient versions of ORAM is perhaps a better choice.

14. Conclusions

In this paper, we investigate the potential vulnerability posed by the disclosure of data access patterns during 'search over encrypted text'. We formalize a model that can be used to launch an inference attack utilizing this vulnerability and empirically show their efficiency in successfully predicting query identities. Furthermore, we present a simple noise addition scheme to thwart such attacks with the expense of retrieving a slightly more documents than needed. We conclude that 'hiding access pattern' is extremely im-

[5] If we assume, $c = 64$, then the constant 1.44c is 92 itself. Please see [18] for more details.

portant in encrypted keyword search and therefore is a necessary characteristics of a secure encrypted search scheme.

15. Acknowledgements

This work was partially supported by Air Force Office of Scientific Research MURI Grant FA9550-08-1-0265, National Institutes of Health Grant 1R01LM009989, National Science Foundation (NSF) Career Grant CNS-0845803, NSF Grants CNS-0964350,CNS-1016343.

References

[1] Anderson, Needham, and Shamir. The steganographic file system. In *IWIH: International Workshop on Information Hiding*, 1998.

[2] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[3] M. Berry and M. Browne. Email surveillance using non-negative matrix factorization. *Computational & Mathematical Organization Theory*, 11(3), 2005.

[4] D. Boneh, G. Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *proc. of EUROCRYPT*, 2004.

[5] D. Boneh, E. Kushilevitz, and R. Ostrovsky. Public key encryption that allows PIR queries. In *proc. of CRYPTO*, 2007.

[6] K. Borgwardt, H. Kriegel, and P. Wackersreuther. Pattern mining in frequent dynamic subgraph. In *ICDM*, pages 818–822. IEEE Computer Society, 2006.

[7] Y. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *International Conference on Applied Cryptography and Network Security (ACNS), LNCS*, volume 3, 2005.

[8] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *proc. of the 13th ACM Conference on Computer and Communications Security, CCS 2006*, pages 79–88. ACM, 2006.

[9] J. Diesner, T. Frantz, and M. Carley. Communication networks from the enron email corpus. *Computational & Mathematical Organization Theory*, 11(3), 2005.

[10] E. Goh. Secure indexes. *Cryptology ePrint Archive*, (Report 2003/216), 2003.

[11] O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious RAMs. *JACM: Journal of the ACM*, 43, 1996.

[12] M. S. Islam, M. Kuzu, and M. Kantarcioglu. Poster: Inference attacks against searchable encryption protocols. In *CCS*, pages 845–847. ACM, 2011.

[13] B. Jackson, J. Scargle, C. Cusanza, D. Barnes, D. Kanygin, R. Sarmiento, S. Subramaniam, and T. Chuang. Optimal partitions of data in higher dimensions. In *proc. of the Conference on Intelligent Data Understanding*, pages 98–108. NASA Ames Research Center, 2010.

[14] S. Kamara and K. Lauter. Cryptographic cloud storage. In *Financial Cryptography Workshops*, volume 6054, pages 136–149. Springer, 2010.

[15] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–679, May 1983.

[16] B. Klimt and Y. Yang. Introducing the enron corpus. In *CEAS*, 2004.

[17] R. Kumar, J. Novak, B. Pang, and A. Tomkins. On anonymizing query logs via token-based hashing. In *Proceedings of the 16th International Conference on World Wide Web, WWW*, pages 629–638. ACM, 2007.

[18] B. Pinkas and T. Reinman. Oblivious RAM revisited. *IACR Cryptology ePrint Archive*, 2010:366, 2010.

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

[20] D. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *IEEE Symposium on Security and Privacy*, 2000.

[21] C. Troncoso, C. Díaz, O. Dunkelman, and B. Preneel. Traffic analysis attacks on a continuously-observable steganographic file system. In *Information Hiding*, volume 4567, pages 220–236. Springer, 2007.

[22] P. Williams, R. Sion, and B. Carbunar. Building castles out of mud: practical access pattern privacy and correctness on untrusted storage. In *Proceedings of the 2008 ACM Conference on Computer and Communications Security, Alexandria, Virginia, USA, October 27-31, 2008*, pages 139–148. ACM, 2008.

[23] X. Zhou, H. Pang, and K.-L. Tan. Hiding data accesses in steganographic file system. In *ICDE*, pages 572–583. IEEE Computer Society, 2004.

[24] G. Zipf. *Selected studies of the Principle of Relative Frequency in Language*. Harvard U.P., 1932.

Appendix A

Proof of Theorem 1 Let $C = \{e_1, e_2, ..., e_n\}$, $P = \{p_1, p_2, ..., p_m\}$ be the encrypted and plain keyword lists and M_C, M_P be the pair similarity matrices for E and P respectively such that $n \leq m$. Let $B = \{(i,j)|f(e_i) = p_j, 1 \leq n, 1 \leq j\}$ be a set of pairs where f is a bijection from E to P and $cost : B \mapsto Integer$ be the evaluation function which is defined as follows:

$$\begin{aligned} cost(B) &= \sum_{i=1}^{n}\sum_{j=i}^{n}(M_C[i,j] - M_P[k,l])^2 \\ &s.t. \quad (i,k) \in B, (j,l) \in B \end{aligned}$$

Notice that constructed setting is equivalent to formulated optimization problem. We can formulate the decision version of the problem as follows: Given C, P along with M_C and M_P, does there exist any B such that $cost(B) \leq k$?

It is easy to see that the problem is in NP. The certificate is the mapping set (B) and a certifier can check if the cost(B) is at most the given bound (k) in polynomial time.

We now show that the problem is NP-Hard by a reduction from the Hamiltonian path problem. Given an instance of Hamiltonian path problem specified by graph G(V, E), we can construct an input $< C, P, M_C, M_P, k >$ for decision version of the optimization problem as follows.

$$C = V \tag{6}$$

$$P = \{1, 2, ..., n\}, n = |V| \tag{7}$$

$$M_C[i,j] = \left\{ \begin{array}{ll} w & \text{if } (v_i, v_j) \in E \text{ and } i < j \\ 0 & \text{otherwise} \end{array} \right\} \tag{8}$$

$$where, 1 \leq i \leq n, 1 \leq j \leq n \tag{9}$$

$$M_P[i,j] = \left\{ \begin{array}{ll} w & \text{if } j = i+1 \\ 0 & \text{otherwise} \end{array} \right\} 1 \leq i \leq n, 1 \leq j \leq n \tag{10}$$

$$k = \left\{ \begin{array}{ll} 0 & \text{if } |E| < |V| - 1 \\ w^2(|E| - |V-1|) & \text{otherwise} \end{array} \right\} \tag{11}$$

We claim that there is a mapping set B for the input $< C, P, M_C, M_P, k >$ such that $cost(B) \leq k$ if and only if G has a Hamiltonian path.

Without loss of generality, suppose G has a Hamiltonian path v_1, v_2, ..., v_n. Since G has an Hamiltonian path, it should have at least $|V| - 1$ edges. Thus, $k = w^2(|E| - |V - 1|)$. In this case, cost of assignment set $\{(1,1), ...(i,i), ..., (n,n)\}$ is bounded by k. In such a case,

$$\begin{aligned} cost(B) &= \sum_{i=1}^{n} \sum_{j=i}^{n} (M_C[i,j] - M_P[k,l])^2 \\ s.t. &\quad i = j \end{aligned}$$

case i : $j = i + 1$

By the construction of M_P, $M_P[i,j] = w$, similarly $M_C[i,j] = w$ for this case since there exists an edge between (v_i, v_{i+1}). Therefore, $\sum_{i=1}^{n} \sum_{j=i}^{n} (M_C[i,j] - M_P[i,j])^2 = 0$ for this case.

case ii : $j \neq i + 1$

By the construction of M_P, $M_P[i,j] = 0$, $M_C[i,j] = w$ if $(v_i, v_j) \in E$ and $i < j$, $M_C[i,j] = 0$ otherwise for this case. There exist $|V| - 1$ edges (v_i, v_j) such that $j = i + 1$ which implies the existence of $(|E| - |V - 1|)$ edges such that $i \neq j+1$. Hence, $M_C[i,j] = w$ for $(|E|-|V-1|)(i,j)$ pairs. Therefore $\sum_{i=1}^{n} \sum_{j=i}^{n} (M_C[i,j] - M_P[i,j])^2 = w^2(|E| - |V - 1|)$ for this case.

Cases i and ii covers all possible (i,j) pairs. Therefore $cost(B) = w^2(|E| - |V - 1|) = k$

Conversely, suppose $cost(B) \leq k$ for some B, then G has a Hamiltonian path. For the sake of contradiction, suppose G has no Hamiltonian path, then

case i : $|E| < |V| - 1 \rightarrow k = 0$

By the construction of M_P, $M_P[i, i+1] = w$ for each (i, i+1) pair for $i, i + 1 \in P$. If v_k is matched with i and v_l is matched with $i + 1$ for $v_k, v_l \in C$, then $(M_C[k,l] - M_P[i, i+1])^2 = 0$ if and only if $(v_k, v_l) \in E$. Since there exist $|V|-1$ (i, i+1) pairs, we need $(|V-1|)$ edges to satisfy $cost(B) = 0$ for any B. However, $|E| < |V| - 1$ for this case and $cost(B) > 0$

case ii : $|E| \geq |V| - 1 \rightarrow k = w^2(|E| - |V - 1|)$

Notice that cost(B) becomes smaller if consecutive elements $(i, i + 1)$ of P is matched with some $v_k, v_l \in C$ such that $(v_k, v_l) \in E$. However, there exist at least one $(i, i + 1)$ pair for $i, i + 1 \in C$, such that i is matched with v_k and $(i + 1)$ is matched with v_l for $v_k, v_l \in C$ and $(v_k, v_l) \notin E$. Otherwise G has a Hamiltonian path which is not the case. Therefore, at most $|V| - 2$ consecutive pairs in P can be matched with some v_k, v_l for $v_k, v_l \in C$ such that $(v_k, v_l) \notin E$. Notice that $cost(B) \geq w^2(|E| - |V - 1| + 2)$ in such a case. Hence $cost(B) > w^2(|E| - |V - 1|) = k$ for this case.

Both case i and ii shows that $cost(B) > k$ for any B if G has no Hamiltonian path. Therefore, we can conclude that G should have a Hamiltonian path if there exist some assignment set B such that $cost(B) \leq k$. ∎