# Security Analysis on One-to-Many Order Preserving Encryption-Based Cloud Data Search

V.Manjunathan, D.Gnana Prakash, A.Raja, Mr.P.Sundaravadivel

*Abstract*— **For ranked search in encrypted cloud data, order preserving encryption (OPE) is an efficient tool to encrypt relevance scores of the inverted index. When using determinis-tic OPE, the ciphertexts will reveal the distribution of relevance scores. Therefore, Wang *et al.* proposed a probabilistic OPE, called one-to-many OPE, for applications of searchable encryp-tion, which can flatten the distribution of the plaintexts. In this paper, we proposed a differential attack on one-to-many OPE by exploiting the differences of the ordered ciphertexts. The experimental results show that the cloud server can get a good estimate of the distribution of relevance scores by a differential attack. Furthermore, when having some background information on the outsourced documents, the cloud server can accurately infer the encrypted keywords using the estimated distributions.**

*Index Terms*— **Searchable encryption, order preserving encryption, privacy, cloud computing.**

## I. Introduction

NOWADAYS users connected to the Internet may store their data on cloud servers and let the servers manage or process their data. They can enjoy convenient and efficient service without paying too much money and energy, as one of the most attractive feature of cloud computing is its low cost [1].

However, no matter how advantageous cloud computing may sound, large number of people still worry about the safety of this technology. If cloud server get direct access to all these users' data, it may try to analyse the documents to get private information. The initial purpose of this action may be kind. The server wants to provide better service by digging into these data and then displaying customer-oriented advertisement, which could be convenient but also annoying. Besides, when we consider sensitive data such as personal health records and secret chemical ingredients, the situation becomes even more serious [2]. Theoretically, the server is not supposed to have access to sensitive data at all; therefore we should ensure the server has no access to leaking these data to an untrusted third party. Thus, sensitive data have to be encrypted before being outsourced to a commercial public cloud [3].

However, encryption on sensitive data presents obstacles to the processing of the data. Information retrieval becomes difficult in the encrypted domain because the amount of outsourced files can be very large and traditional search patterns can not be deployed to ciphertext retrieval directly. Users need to download all the data, decrypt it all, and then search keywords like plaintext retrieval. To overcome this, Searchable Encryption (SE) [4] was proposed to make query in the encrypted domain possible while still preserving users' privacy. There are several problems in searchable encryption: fuzzy search, ranked search, multi-keyword search and so on. Song *et al.* [5] first proposed a search scheme only supporting single Boolean keyword search. After that plenty of searchable encryption methods [6]–[9] arose to improve efficiency and reduce communication overhead.

Applying order preserving encryption (OPE) [10] is one practical way of supporting fast ranked search. This algorithm was first proposed in 2004 to solve encrypted query problems in database systems. OPE is a symmetric cryptosystem, therefore it is also called order-preserving symmetric encryption (OPSE). The order-preserving property means that if the plaintexts $x_1 < x_2$, then the corresponding ciphertexts $E(x_1)$ and $E(x_2)$ satisfy $E(x_1) < E(x_2)$.

Boldyreva *et al.* initiated the cryptographic study of OPE schemes [11], [12], in which they defined the security of OPE and proposed a provably secure OPE scheme. However, the security definition and the constructions of OPE in [11] and [12] are based on the assumption that OPE is a deterministic encryption scheme which means that a given plaintext will always be encrypted as a fixed ciphertext. However, deterministic encryption leaks the distribution of the plaintexts, so it cannot ensure data privacy in most applications. For instance, in privacy-preserving keywords search, OPE is used to encrypt relevance scores in the inverted index [16]. As noted by Wang *et al.* [16], when using a deterministic OPE, the resulting ciphertext shares exactly the same distribution as the relevance score, by which the server can specify the keywords [14], [15]. Therefore, Wang *et al.* [16] improved the OPE in [11] and proposed a "One-to-Many OPE" in their secure keyword search scheme, where they tried to construct a probabilistic encryption scheme and conceal the distribution of the plaintexts.

However, we discover that the One-to-Many OPE [16] cannot ensure the expected security. In fact, although the ciphertexts of One-to-Many OPE conceals the distribution

Manuscript received January 14, 2015; revised March 19, 2015; accepted May 3, 2015. Date of publication May 20, 2015; date of current version July 22, 2015. This work was supported in part by the National Natural Science Foundation of China under Grant 61170234 and Grant 60803155, in part by the Strategic Priority Research Program through the Chinese Academy of Sciences under Grant XDA06030601, and in part by the Funding of Science and Technology on Information Assurance Laboratory under Grant KJ-13-02. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Wanlei Zhou.

The authors are with the CAS Key Laboratory of Electromagnetic Space Information, University of Science and Technology of China, Hefei 230026, China (e-mail: lee0525@mail.ustc.edu.cn; zhangwm@ustc.edu.cn; yangce@mail.ustc.edu.cn; ynh@ustc.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2015.2435697

TABLE I

EXAMPLE OF POSTING LIST OF THE INVERTED INDEX

| Keyword | $w$ | | | |
|---|---|---|---|---|
| | $F_1$ | $F_2$ | ... | $F_{f_w}$ |
| Relevance Score | 8.6 | 6.1 | ... | 7.3 |

of the plaintexts, an adversary may estimate the distribution from the differences of the ciphertexts. So in this paper, we propose a differential attack on the One-to-Many OPE. Our experimental results show that, when applying this attack to the secure keyword search scheme of [16], the cloud server can get an estimation of the distribution of the the relevance scores, and furthermore accurately reveal the encrypted keywords.

The rest of this paper is organized as follows. We first describe the plaintext search model and ciphertext search model in Section II. Then, in Section III, the basic OPE, One-to-Many OPE, and privacy requirement in cloud comput-ing are briefly reviewed. We elaborate on differential attack on One-to-Many OPE and further attack with background information of outsourced data in Section IV and Section V respectively. Finally the conclusion is given in Section VI.

## II. SEARCHING MODEL

### A. Plaintext Searching Model

In practice, to realize effective data retrieval on large amount of documents, it is necessary to perform relevance ranking on the results. Ranked search can also significantly reduce network traffic by sending back only the most relevant data. In ranked search, the ranking function plays an important role in calculating the relevance between files and the given searching query. The most popular relevance score is defined based on the model of $TF \times IDF$, where term frequency (TF) is the number of times a term (keyword) appears in a file and inverse document frequency (IDF) is the ratio of the total number of files to the number of files containing the term. There are many variations of $TF \times IDF$-based ranking functions, and in [16], the following one is adopted.

$$Score(w, F_d) = \frac{1}{|d|} \cdot (1 + \ln f_{d,w}) \cdot \ln \frac{1 + fd}{N} \quad (1)$$

Herein, $w$ denotes the keyword and $f_{d,w}$ denotes the TF of term $w$ in file $F_d$ ; $N/f_w$ denotes IDF where $f_w$ is the number of files that contain term $w$ and $N_d$ is the total number of documents in the collection; and $|F_d|$ is the number of indexed terms containing in file $F_d$ , i.e., the length of $F_d$ .

To realize fast search, the keywords, IDs of files, and the relevance scores are usually organized as an index structure named "Inverted Index". An example on posting list of the Inverted Index is shown in TABLE I. With a complete Inverted Index, the server can complete retrieval task by simply com-paring the relevance scores stored in the index which represent the importance level of each file for a certain keyword.

### B. Ciphertext Searching Model

Due to the special background of cloud computing, unlike traditional plaintext information retrieval, there are usually

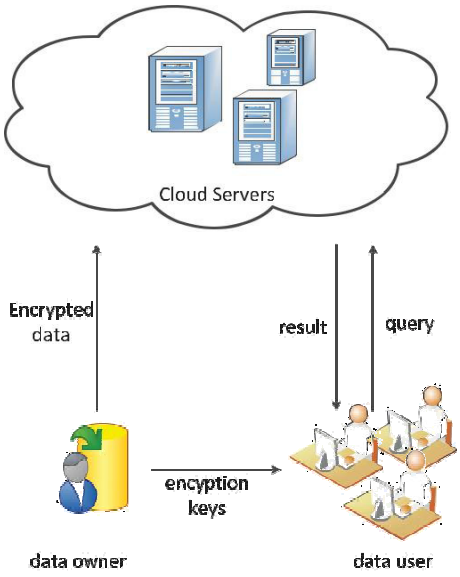



Fig. 1. Framework of retrieval over encrypted cloud data.

TABLE II

EXAMPLE OF ENCRYPTED POSTING LIST OF THE INVERTED INDEX

| Keyword | $hash(w)$ | | | |
|---|---|---|---|---|
| File ID | $F_1$ | $F_2$ | ... | $F_{f_w}$ |
| Relevance Score | $E'(8.6)$ | $E'(6.1)$ | ... | $E'(7.3)$ |

three entities in cloud data retrieval as shown in Fig. 1: data owner, remote cloud server and users. A data owner can be an individual or a corporation, i.e., it is the entity that owns a collection of documents $D_c = \{D_1, D_2 \ldots D_{Nd}\}$ that it wants to share with trusted users. The keyword set is marked as $W = \{w_1, w_2 \ldots w_{Nw}\}$. For security and privacy concerns, documents have to be encrypted into $\xi = \{E(D_1), E(D_2) \ldots E(D_{Nd})\}$ before being uploaded to the cloud server. Additionally, the plaintext index has to be encrypted into $I$ to prevent information leakage.

The encrypted form of the example of the posting list of the Inverted Index is shown in TABLE II, in which the keyword $w_i$ is protected by a Hash function $hash()$, and the relevance scores are encrypted by a encryption scheme $E()$.

We use TABLE II as an example to see how a cloud server conducts a secure search based on an encrypted index. In the search procedure, a user first generates a search request in a secret form — a trapdoor $T(w)$. In this example, the trapdoor is just the hash values of the keyword of interest.

Once the cloud server receives the trapdoor $T(w)$, it compares it with the hash values of all keywords in the index $I$, then the desired documents which are corresponding to keyword $w$ are found. Next, the server returns the matched file IDs: $F_1, F_2, \ldots,$ $F_{fw}$ to the user. Finally, the user can download all the encrypted documents based on the given IDs and decrypt them. A desirable system is supposed to return the documents in a ranked order by their relevance with the queried keyword, but using traditional encryption schemes will disorder relevance scores. Therefore, in [16] Order Preserving

Encryption (OPE) is applied to encrypt the relevance scores, which enables the server to quickly perform ranked search without knowing the plain relevance scores.

## III. OPE Vs. One-to-Many OPE

### A. OPE

OPE is a symmetric cryptosystem, so it is also called order-preserving symmetric encryption (OPSE). The order-preserving property means that if the plaintexts have such a relationship as $x_1 < x_2$, then the corresponding ciphertexts $E(x_1)$ and $E(x_2)$ satisfy $E(x_1) < E(x_2)$.

Boldyreva *et al.* [11] initiated the cryptographic study of OPE schemes, and they defined the security of an OPE scheme using the ideal object. Note that any order-preserving function $g$ from domain $D = \{1, 2, \ldots, M\}$ to range $R = \{1, 2, \ldots, N\}$ can be uniquely defined by a combination of $M$ out of $N$ ordered items. The ideal object is just a function that is randomly selected from all order-preserving functions, which is called a random order-preserving function (ROPF). Thus, with the spirit of pseudorandom functions, an OPE scheme is defined to be secure if the adversary cannot distinguish the OPE from the ROPF. In [11], the authors also constructed an efficient OPE scheme satisfying this secure criterion. The construction is based on the relation between the random order-preserving function and the hyper-geometric probability distribution (HGD), and a HGD simpler is used to select an order-preserving function in a pseudoran-dom manner.

In the OPE scheme of [11], the range $R$ is divided into some nonoverlapping interval buckets with random sizes. The random-sized bucket is determined by a binary search based on a random HGD sampler. In [16], the procedure of binary search is described as Algorithm 1, where *T apeGen()* is a random coin generator.

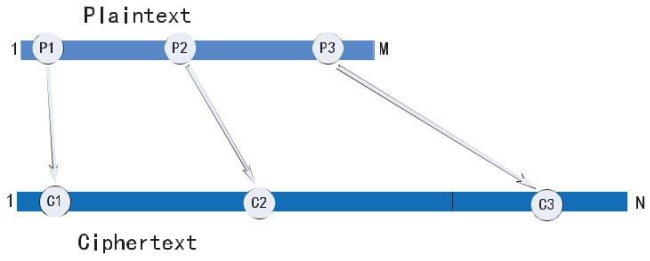
**Algorithm 1** BinarySearch

**Input:** $\{K, D, R, m\}$

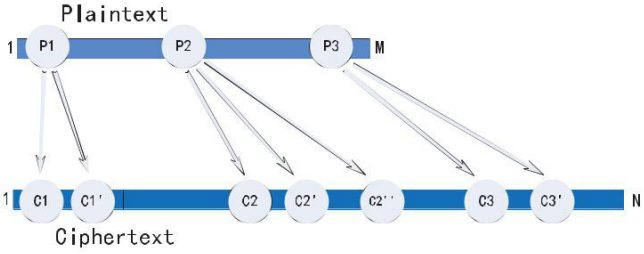
1: $M \leftarrow lengt\,h(D)$; $N \leftarrow lengt\,h(R)$
2: $d \leftarrow mi\,n(D) - 1$; $r \leftarrow mi\,n(R) - 1$
3: $y \leftarrow r + cei\,l(N/2)$
4: $coi\,n \xleftarrow{R} T\,apeGen(K, (D, R, y||0))$
5: $x \xleftarrow{R} d + H\,G\,D(coi\,n, M, N, y - r)$ 6:
$x = d + f$
7: **if** $m \le x$ **then**
8: $D \leftarrow \{d + 1, \ldots, x\}$ 9:
$R \leftarrow \{r + 1, \ldots, y\}$
10: **else**
11: $D \leftarrow \{x + 1, \ldots, d + M\}$
12: $R \leftarrow \{y + 1, \ldots, r + N\}$
13: **end if**

**Output:** $\{D, R\}$

After the binary search, a plaintext $m$ is mapped into a bucket in the range $R$, and then the OPE algorithm assigns a fixed value in the bucket as the encrypted value of $m$. The encryption process of Algorithm 1 is illustrated in Fig. 2(a),




(a) Deterministic OPE




(b) One-to-Many OPE

Fig. 2. Comparison between deterministic OPE and One-to-Many OPE.

which shows that a given plaintext $m_i$ will always be mapped to a fixed ciphertext $c_i$ belonging to a bucket selected by the binary search procedure, therefore it is a deterministic encryption.

### B. One-to-Many OPE

Wang *et al.* [16] noted that, in applications of privacy-preserving keyword search, if a deterministic OPE is used to encrypt relevance scores, the ciphertexts will share exactly the same distribution as its plain counterpart, by which the server can specify the keywords.

Therefore, Wang et al. [16] modified the original OPE [11] to a probabilistic one, called "One-to-Many OPE". For a given plaintext $m$, i.e., a relevance score, the "One-to-Many OPE" first employs Algorithm 1 to select a bucket for $m$, and then randomly chooses a value in the bucket as the ciphertext. The randomly choosing procedure in the bucket is seeded by the unique file IDs together with the plaintext $m$, and thus the same relevance score in the Inverted Index will be encrypted as different ciphertexts. The encryption process of "One-to-Many OPE" is described in Algorithm. 2 [16], which is also illustrated in Fig. 2(b).

**Algorithm 2** One-to-Many Order Preserving Encryption

**Input:** $\{K, D, R, m, i\,d(F)\}$
**while** $|D|! = 1$ **do**
$\{D, R\} = bi\,nar\,ysear\,ch(K, D, R, m)$
**end while**
$coi\,n \xleftarrow{coin} T\,apeGen(K, (D, R, 1||m, i\,d(F)))$
$c \xleftarrow{coin} R$
$c = r\,ound(coi\,n)$

**Output:** c

*Example 1:* To compare the encrypted results of OPE and One-to-Many OPE, we take the posting list of

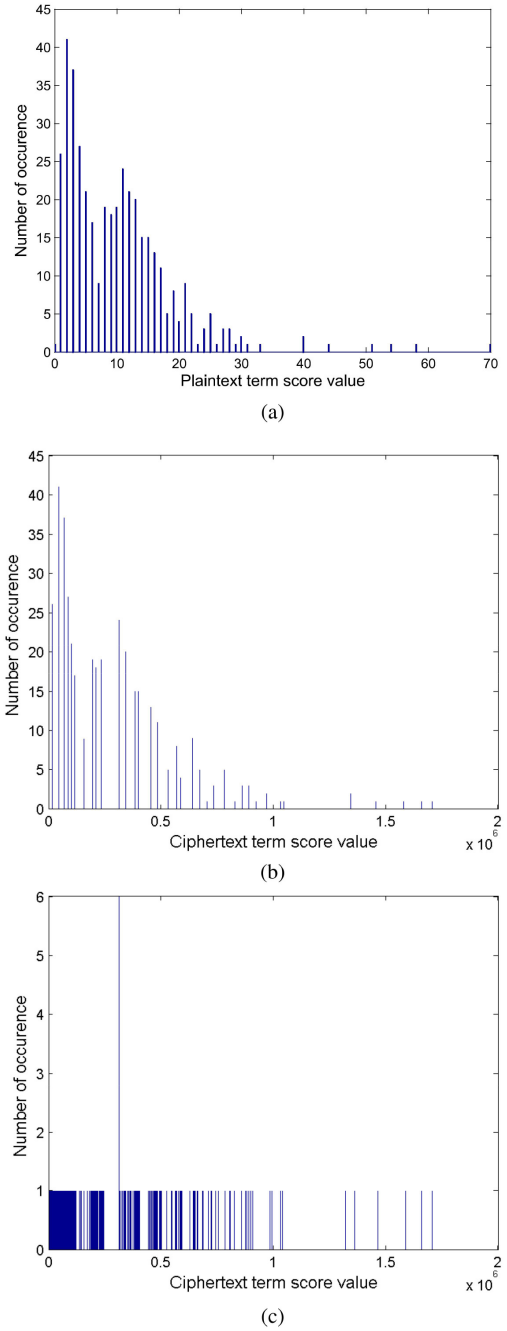



Fig. 3. Comparison between plaintext distribution and ciphertext distribution obtained by two kinds of OPE. (a) Plaintext distribution. (b) Ciphertext Distribution of deterministic OPE. (c) Ciphertext distribution of One-to-Many OPE.

keyword "weather" as example that is generated from the the TREC data [17]. The relevance scores are encoded into integers, from which we get the plaintext distribution shown in Fig. 3(a). The distributions of the encrypted results obtained by deterministic OPE and One-to-Many OPE are shown in Fig. 3(b) and Fig. 3(c) respectively.

Comparing Fig. 3(b) with Fig. 3(a), we can see that deterministic OPE makes the plaintexts and the ciphertexts share the same distribution, which would make it too easy for an attacker to get the exact keyword's information. As shown in Fig. 3(c), when the size of the ciphertext domain is large enough (such as $2 \times 10^6$), One-to-Many OPE can flatten the distribution of plaintexts. In fact, with One-to-Many OPE, almost all encrypted values appear only once except for a few appearing twice or more.

### C. Privacy Threat Models

The purpose of both OPE and One-to-Many OPE is to prevent information leakage to the cloud server. The cloud server is considered as "semi-honest", also called "honest but curious" [18]. Specifically, the cloud server will not attempt to remove encrypted data files or index from the storage, and it will also correctly follow the designed protocol specification and execute the procedure faithfully. However, it is curious to handle the stored data and tries to analyze the data to learn additional information.

When talking about the "honest but curious" model, usually there are two attack models Known Ciphertext Model and Known Background Model [19].

Known Ciphertext Model assumes that the cloud server can only get access to the encrypted files and the encrypted index. In this model the server can only dig into the ciphertexts without any other background information, and thus security means that the keywords and documents information are strictly protected and there is no indirect way to speculate these information.

Known Background Model is closer to the real-world situation in the cloud application. The cloud server is supposed to possess more knowledge than what can be accessed in the Known Ciphertext Model. It may intentionally collect related statistical information about the outsourced documents, and with this information the server can infer more sensitive information.

Next, we will propose attacking methods on One-to-Many OPE under these two threat models respectively.

## IV. Differential Attack on One-to-Many OPE Under Known-Ciphertext-Model

In Fig. 3(c), it can be seen that One-to-Many OPE has successfully hidden the distribution of the plaintexts, but the security of One-to-Many OPE has not endured strict cryptanalysis. In this section, we will show that, by analyzing the differences between the ciphertexts, the cloud server can get an estimation on the distribution of the plaintexts.

As shown in Fig. 2(b), each plaintext value $m$ is mapped into many possible ciphertexts belonging to a fixed bucket, and the ciphertext is randomly selected in the bucket. Therefore, the scatter of ciphertexts in a bucket will be dense for a plaintext value with high frequency, but will be sparse for a plaintext value with low frequency. Although the sizes of the buckets are randomly determined, the density of ciphertexts in each bucket will vary according to the frequency of the corresponding plaintext, and thus the profile of the plaintexts' frequency can be portrayed by the density of ciphertexts. Note that the density of ciphertexts can be revealed by the differences between the neighboring ciphertexts that we call "differential ciphertexts". In other words, if we can locate the change points of the distribution of the differential ciphertexts, we can

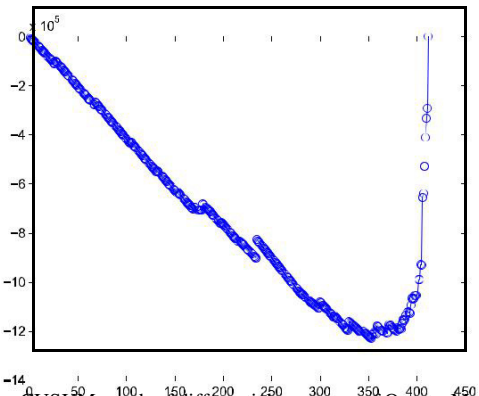


Fig. 4. CUSUM graph of differential sequence of One-to-Many OPE.

determine the boundaries of the buckets in the ciphertext range $R = \{1, 2, \ldots, N\}$. With these boundaries, the histogram of the plaintexts can be easily estimated by counting the number of ciphertexts belonging to each bucket. Therefore, the cloud server may reconstruct the distribution of plaintexts from the differential ciphertexts, which we call "differential attack".

The key issue in "differential attack" is locating the change point in the differential sequence of the ciphertexts. There are many statistical methods to realize such Change Point Analysis (CPA), and we use the cumulative sum (CUSUM) based CPA [21] to describe the procedure of "differential attack", which consists of six steps.

*1) Sort the Encrypted Values:* Suppose that the original ciphertext sequence is $c_1, c_2, \ldots, c_L$. Sort the ciphertext sequence in ascending order, and get $c_{i1} \le c_{i2} \le \cdots \le c_{iL}$.

*2) Generate the Differential Sequence:* The differential sequence $\{d_i, 1 \le i \le L-1\}$ of the ordered ciphertexts is obtained by calculating $d_1 = c_{i2} - c_{i1}, d_2 = c_{i3} - c_{i2}, \ldots, d_{L-1} = c_{iL} - c_{iL-1}$.

*3) Generate CUSMU Sequence:* To get the CUSUM of $d_i$ $(1 \le i \le L-1)$, we first compute their average value:

$$\bar{d} = \frac{1}{L-1} \sum_{i=1}^{L-1} d_i. \quad (2)$$

Set the initial value of cumulative sum as $S_0 = 0$. The other cumulative sum values are calculated in a recursion way such that

$$S_i = S_{i-1} + (d_i - \bar{d}), \quad i = 1, 2, \ldots, L-1. \quad (3)$$

The CUSUM is defined as the cumulative sum of each data minus the average value, so the final value should always be zero, i.e., $S_{L-1} = 0$. A CUSUM chart can be obtained by drawing the cumulative sum $S_i$ in order for $0 \le i \le L-1$. If there is a period of data which is greater than the average value, an ascending curve will occur on the chart; otherwise, a descending curve will occur on the chart. A change point on the chart refers to a sudden change in the curve. In Fig. 4, we depict the CUSUM chart of the differential sequence of ciphertexts obtained by One-to-Many OPE in Example 1, which shows that a change point took place.

*4) Locating One Change Point:* To be sure that a change indeed took place, we determine a confidence level by performing a bootstrap analysis. Define

$$S_{max} = \max_{0 \le i \le L-1} S_i, \quad (4)$$

$$S_{min} = \min_{0 \le i \le L-1} S_i, \quad (5)$$

$$S_{diff} = S_{max} - S_{min}. \quad (6)$$

$S_{diff}$ is an estimator of the changing magnitude, with which a single bootstrap is performed by following four steps:

a) Generate a bootstrap sample of $L-1$ units, denoted as $x_1, x_2, \ldots, x_{L-1}$, by randomly reordering the original $L-1$ differential values $d_1, d_2, \ldots, d_{L-1}$.
b) Calculate the CUSUM of the bootstrap sample, denoted as $S^0, S^1, \ldots, S^{L-1}$.
c) Calculate the changing magnitude of the bootstrap sample $S^{diff}$.
d) Determine whether $S^{diff}$ is less than the original changing magnitude $S_{diff}$.

The bootstrap sample mimics the behavior if no change has occurred. Perform $N_b$ bootstraps, where $N_b$ is large enough, and let $U$ be number of bootstraps for which $S^{diff} < S_{diff}$. Then the confidence level that a change occurred is defined as follows:

$$Confidence\ Level = \frac{U}{N_b}. \quad (7)$$

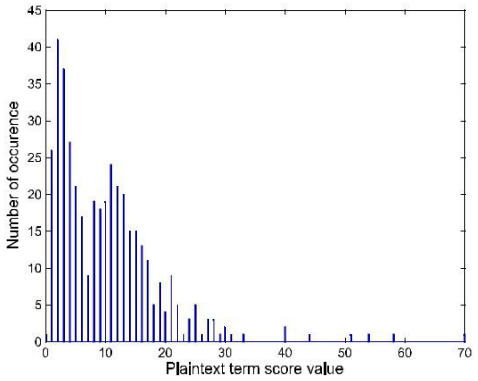
If the *Confidence Level* is larger than a pre-set threshold $\beta$ (typically $\beta = 0.9$ or 0.95), we determine that a change has occurred. After detecting a change, the change point $v_1$ is determined by

$$v_1 = \arg\max_{0 \le i \le L-1} |S_i|. \quad (8)$$

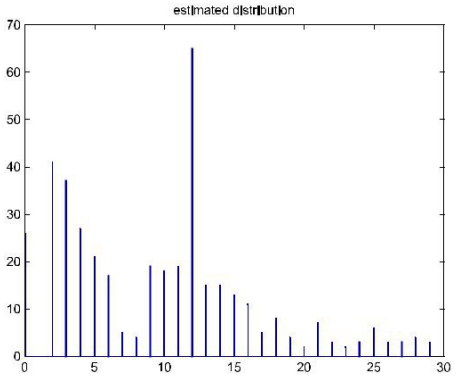
*5) Locating Other Change Points:* Assume that Step **4** outputs one change point $v_1$, which thus divides the sequence $d_1, d_2, \ldots, d_{L-1}$ into two subsequences: $d_1, d_2, \ldots, d_{v_1}$ and $d_{v_1+1}, d_{v_1+2}, \ldots, d_{L-1}$. Then we take change point analysis, i.e., Steps **3** and **4**, on these two subsequences respectively. Assume that change points, $v_2$ and $v_3$, are detected in the two subsequences respectively. Obviously, $v_2 < v_1 < v_3$, which can divide the original sequence into four subsequences. Take change point analysis on these four subsequences and output eight change points...and so on. This process will end when we cannot detect change points in any subsequence. Assume that $B$ change points in total are found, denoted by $v_1, v_2, \ldots, v_B$.

*6) Generating the Estimated Histogram of Plaintexts:* Sort the change points $v_1, v_2, \ldots, v_B$ in ascending order, and get $v_{i1} < v_{i2} < \cdots < v_{iB}$. Let $v_{i0} = 0$ and $v_{iB+1} = N$, and then the ciphertext range $R = \{1, 2, \ldots, N\}$ can be divided into $B+1$ disjointed intervals: $[v_{i0}+1, v_{i1}], [v_{i1}+1, v_{i2}], \ldots, [v_{iB}+1, v_{iB+1}]$. Count the number of ciphertexts dropped in each interval:

$$h_k = |c_j | v_{ik-1} < c_j \le v_{ik}|, \quad 1 \le k \le B+1 \quad (9)$$




(a) Original distribution




(b) Estimated distribution

Fig. 5. Comparison of Original and Estimated distribution.

Let $h_k$ be the height of the $k$th bin, and we get a histogram with $B + 1$ bins, which is just the estimation of the distribution of the plaintexts.

Fig. 5(b) is the estimated distribution of the relevance scores of keyword "weather" in Example 1. Comparing with Fig. 5(a), we can see that, by differential attack, the cloud server can find a similar profile of the distribution of the relevance scores with limited errors. Note that this attack is executed in Known Ciphertext Model, because the cloud server only needs to observe the ciphertexts.

## V. Further Attack Under Known-Background-Model

### A. Identifying the Keywords

In this subsection, we will show that, if the cloud server has some background knowledge of the stored data, it can even infer what the keyword is based on the estimated distribution of the relevance scores.

If the curious server knows what the encrypted documents are roughly about, it can collect many relative documents using a tool such as a web crawler, and get a mimic document collection.

For instance, suppose that a server wants to attack an encrypted dataset whose documents are from website *engli sh.cntv .cn/neWs/s por t s*, and the attacker has priori knowledge that these documents are about sports news. Then it can conduct a document mining work on another similar website *WWW.chi nadai l y.com.cn/s poor t s* to get a mimic document set. As sports news in a short period share high similarity, we can assume that the distributions of keywords from two data sets are remarkably similar and this imitation has high accuracy.

Based on these, the cloud server can then generate an Inverted Index for the mimic document collection. Assume that there are $N_W$ keywords of interest in this Inverted Index. For the $i$ th keyword $\bar{w}_i$, the cloud server can calculate the histogram of the relevance scores in the corresponding posting list, denoted as $H_i$. Take $H_i$ as the feature of the keyword $\bar{w}_i$.

On the other hand, for the encrypted keyword *hash(w)* in the encrypted Inverted Index, the cloud server can get an estimated histogram of the relevance scores by using differential attack, denoted as $H_W$. Then the cloud server can guess what *hash(w)* is by comparing $H_W$ with $H_i$ for $1 \le i \le N_W$.

If the top $k$% most similar features to $H_W$ are $H_{i1}, H_{i2}, \ldots, H_{iJ}$, the cloud server can be confident that the keyword $w$ belongs to the set $\{\bar{w}_{i1}, \bar{w}_{i2}, \ldots, \bar{w}_{iJ}\}$. When such inference is correct, the smaller $k$ or $J$ is, the more information on $w$ is leaked to the cloud server. In fact, if the cloud server has enough background knowledge, it can accurately identify what the keyword $w$ is, i.e., $J = 1$.

We adopt relative entropy here to evaluate the similarity between $H_W$ and $H_i$. Relative entropy is also called K-L divergence, which is a measurement of the similarity of two probability distributions. For discrete probabilistic distribution $P$ and $Q$, their relative entropy is defined as follows:

$$D_{KL}(P||Q) = \sum_i P(i) \ln \frac{P(i)}{Q(i)}, \quad (10)$$

where all $Q(i)$ and $P(i)$ should be larger than zero and the default value of $0 \ln 0$ is 0.

Because change point analysis on ciphertext differences contains inevitable deviation, the estimated distribution includes location errors and height errors, we should not directly use histograms of $H_W$ and $H_i$ to compute relative entropy. Instead, we should regularize both $H_W$ and $H_i$ to $z$ bins according to the procedure of CPA, and then compare the regularized histograms. Herein we call $z$ as *BINsize*. The detailed regularization procedure is as follows:

From the recursive change point analysis on ciphertext differences we can get first $z - 1$ change points: $v_1, v_2, \ldots, v_{z-1}$. Sort the change points $v_1, v_2, \ldots, v_{z-1}$ in ascending order, and get $v_{i_1} < v_{i_2} < \ldots < v_{i_{z-1}}$. Let $v_{i_0} = 0$ and $v_{i_z} = N$, and then the ciphertext range $[1, 2, \ldots, N]$ can be divided into $z$ disjointed intervals: $[v_{i_0} + 1, v_{i_1}], [v_{i_1} + 1, v_{i_2}], \ldots, [v_{i_{z-1}} + 1, v_{i_z}]$. Count the number of ciphertexts dropped in each interval, we can get a histogram with $z$ bins as $ep_1, ep_2, \ldots, ep_z$, which is the regularized distribution of $H_W$.

To regularize $H_i$, we denote $H_i = \{b_1, b_2, \ldots, b_M\}$, where $b_j$ is the occurrence of value $j$ for $1 \le j \le M$. We do change point analysis on the sequence $\{b_1, b_2, \ldots, b_M\}$ and record the first $z - 1$ change points as $u_1, u_2, \ldots, u_{z-1}$. Sort the change points $u_1, u_2, \ldots, u_{z-1}$ in ascending order, and

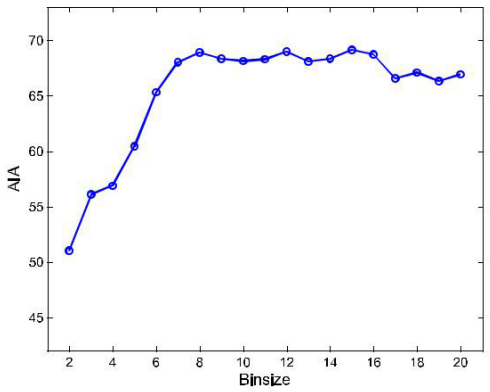



Fig. 6. Average identifying accuracy (AIA) for different BINsize.

get $u_{i1} < u_{i2} < \cdots < u_{iz-1}$ . These $z-1$ change points will divide $H_i$ into $z$ regularized bins as $e_1, e_2, \ldots, e_z$ , which is the regularized distribution of $H_i$ .

Normalize these bins as $P_i(i) = ep_i/(ep_1 + ep_2 + \ldots + ep_z)$ and $Q_W(i) = e_i/(e_1 + e_2 + \ldots + e_z)$ for $1 \le i \le z$, and the relative entropy of $H_W$ and $H_i$ is estimated as:

$$D_{KL}(H_W \| H_i) = \sum_{i=1}^{z} P_i(i) \ln \frac{P_i(i)}{Q_W(i)}, \quad (11)$$

For a given keyword $w$ whose estimated histogram by using differential attack is $H(w)$, and mimic histogram by the cloud server is $H(w)^-$, we define the identifying accuracy of keyword $w$ as

$$IA_W = \frac{\sum_{i=1}^{N_W} F(D_{KL}(H_W \| H_i), D_{KL}(H_W \| H_{W^-}))}{N_W}, \quad (12)$$

where

$$F(x, y) = \begin{cases} 1 & y < x \\ 0 & y \ge x \end{cases}. \quad (13)$$

Note that the maximum value of $IA(w)$ is $(N_W - 1)/N_W$ , and, when $IA(w)$ reaches the maximum value, the keyword $w$ can be uniquely identified by the cloud server. Furthermore, we define the average identifying accuracy (AIA) for $N_t$ keywords as follows.

$$AIA = \frac{\sum_{j=1}^{N_t} IA_{w_j}}{N_t}. \quad (14)$$

*B. Experimental Results*

We demonstrate a thorough experimental evaluation on the TREC data [17], which consists of 7594 documents and 18238 distinct keywords, from which we select $N_W = 2061$ keywords of interest. In other words, the Inverted Index consists of 2061 post listings in our experiments.

To imitate the background obtained by the cloud server, we randomly select a subset of $100\alpha$ percent of the whole document set. The keywords $w^-_i$ and corresponding feature $H_i$ are generated from this subset for $1 \le i \le N_W$ . Herein, we use $\alpha$ as a parameter to describe the similarity of the background acquired by the cloud server to the outsourced document collection. We call $\alpha$ the background strength.

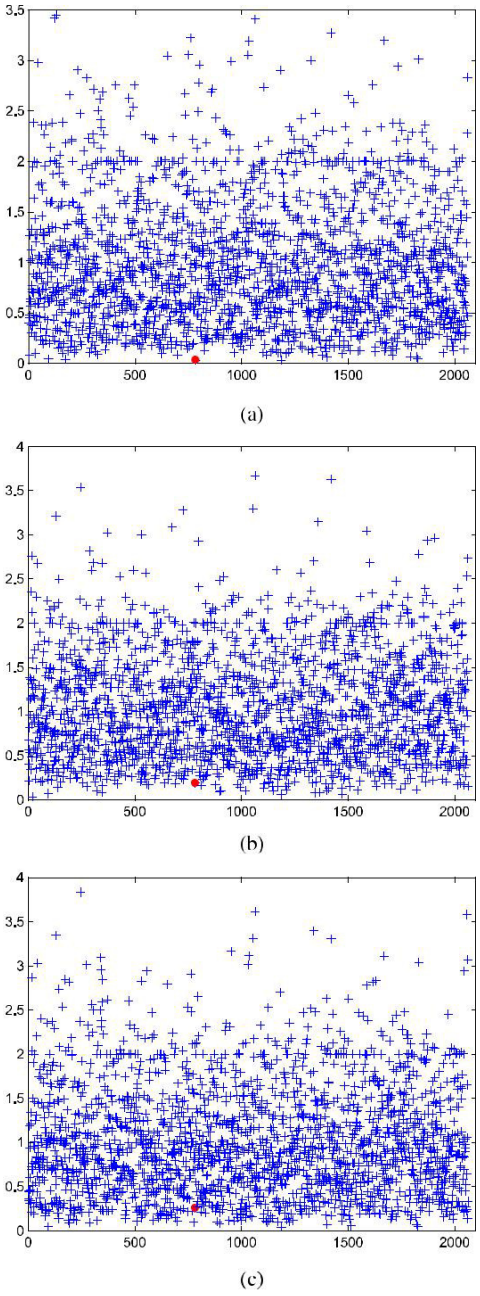


Fig. 7. Identifying keyword "weather" for different background strength $\alpha$. (a) $\alpha = 0.90$. (b) $\alpha = 0.66$. (c) $\alpha = 0.50$.

A large background strength means that the server has a distribution close to the real distribution of the relevance scores that have been encrypted, and vice versa. In this experiment, we choose $\alpha = 0.90$, 0.66 and 0.50.

First, the *BIN size z* in Eq. (11) is an important parameter that has to determined, and obviously choosing different *BIN sizes* will affect the identifying accuracy. We perform a series of experiments to see which range of $z$ would be the best for the attacker to get the most accurate result. In this experiment, we set $\alpha = 0.66$ for consistency, and select $N_t = 100$ representative keywords from the total $N_W$ keywords to estimate AIA. The AIA trend chart for different *BIN sizes* is depicted in Fig. 6, which shows that choosing $z = 8$ would

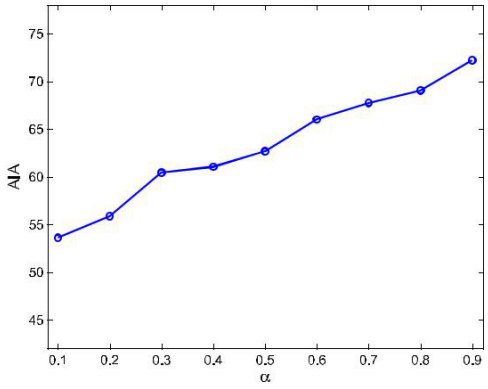



Fig. 8. AIA for different background strength $\alpha$.

TABLE III
NUMBER OF SUCCESSFULLY IDENTIFIED KEYWORDS OUT OF 100 TEST ONES FOR DIFFERENT BACKGROUND STRENGTH $\alpha$

| $\alpha$ | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|
| Count | 2 | 14 | 16 | 24 | 29 | 36 | 40 |

reach the best identifying accuracy. Therefore, in the following experiments we set $z = 8$. We should note that the choice of $z$ should vary with term frequency, otherwise setting $z$ too small or too big will cause measurement deviation. For keywords with higher term frequencies we should set bigger $z$ values, and vice versa.

Next, we try to recognize the keyword "$w$ = weather". The relative entropies between $H_W$ and $H_i$ for $1 \le i \le N_W$ are depicted in Fig. 7(a), Fig. 7(b), and Fig. 7(c) for $\alpha$ = 0.90, 0.66 and 0.5 respectively. In these figures, for the "correct" keyword "$w_i$ = weather", the relative entropy is marked by a red circle. As shown in Fig. 7(a), the minimum relative entropy appears at "$w_i$ = weather", that is, the server can successfully identify that $w$ is just "weather". Fig. 7(b) shows the relative entropy of "$w_i$ = weather" is among the top 2.28% results, which means when $\alpha$ = 0.66, the server can successfully identify $w$ as "weather" in a small range as 2.28% of all. When $\alpha$ comes to 0.50, the server can successfully identify $w$ as "weather" in a relatively bigger range as 7.67%.

Furthermore, we conduct extensional experiments on representative $N_t$ = 100 keywords selected from the total $N_W$ keywords including "weather" to make our results more convincing. The calculation of $AIA$ is repeated ten times with different encryption keys to avoid accidental error, and the average $AIA$ results are shown in Fig. 8. The $AIA$ curve is drawn to show the trend of $AIA$ values with different background strengths $\alpha$ from 0.1 to 0.9.

In addition, in TABLE III, we introduce the notion of "successfully identified". An attacked keyword is considered "successfully identified" if the relative entropy of the keyword falls in the top 5% results. We list the number of "successfully identified" keywords out of the 100 test ones for different background strengths $\alpha$ from 0.3 to 0.9.

Fig. 8 and TABLE III show that when there is more background knowledge, i.e., a larger $\alpha$, the server can infer the keyword information more accurately.

## VI. CONCLUSION

In ranked search of encrypted cloud data, probabilistic OPE is needed to preserve the order of relevance scores and conceal their distributions at the same time. One-to-Many OPE [16] is a scheme designed for such a purpose. However, in this paper, we demonstrate that the cloud server can estimate the distribution of relevance scores by change point analysis on the differences of ciphertexts of One-to-Many OPE. Furthermore, the cloud server may identify what the encrypted keywords are by using the estimated distributions and some background knowledge.

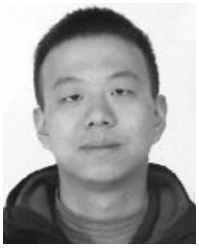
On the other hand, some methods can be used to resist the proposed attack. One is to improve the One-to-Many OPE itself. For instance, we can divide plaintexts having the same value into several sets and divide the corresponding bucket into several sub-buckets. By mapping each plaintext set into one sub-bucket, some new change points will appear in the differential attack, which will cover up the original distribution of plaintexts. Another possible method is to add noise into the inverted index by adding some dummy documents IDs and keywords, and forging corresponding relevance scores.

In our future work, we will elaborate these ideas to design secure methods of probabilistic OPE and schemes for search in encrypted data.

## REFERENCES

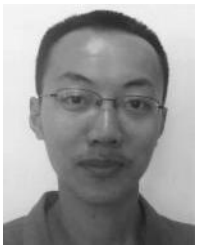
[1] P. Mell and T. Grance. (Jan. 2010). *Draft NIST Working Definition of Cloud Computing*. http://csrc.nist.gov/groups/SNS/cloudcomputing/index.html
[2] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," *J. Netw. Comput. Appl.*, vol. 34, no. 1, pp. 1–11, 2011.
[3] B. Krebs. (2009). *Payment Processor Breach May Be Largest Ever*. [Online]. Available: http://voices.washingtonpost.com/securityfix/2009/01/payment_processor_breach_may_b.html
[4] M. Abdalla *et al.*, "Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions," in *Advances in Cryptology*. Berlin, Germany: Springer-Verlag, 2005, pp. 205–222.
[5] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy*, May 2000, pp. 44–55.
[6] E.-J. Goh. (2003). "Secure indexes," Cryptology ePrint, Tech. Rep. 2003/216. [Online]. Available: http://eprint.iacr.org/
[7] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology*. Berlin, Germany: Springer-Verlag, 2004, pp. 506–522.
[8] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Applied Cryptography and Network Security*. Berlin, Germany: Springer-Verlag, 2005, pp. 442–455.
[9] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, 2006, pp. 79–88.
[10] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2004, pp. 563–574.
[11] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill, "Order-preserving symmetric encryption," in *Advances in Cryptology*. Berlin, Germany: Springer-Verlag, 2009, pp. 224–241.
[12] A. Boldyreva, N. Chenette, and A. O'Neill, "Order-preserving encryption revisited: Improved security analysis and alternative solutions," in *Advances in Cryptology*. Berlin, Germany: Springer-Verlag, 2011, pp. 578–595.
[13] L. Xiao and I.-L. Yen, "Security analysis for order preserving encryption schemes," in *Proc. 46th Annu. Conf. Inf. Sci. Syst.*, Mar. 2012, pp. 1–6.
[14] A. Swaminathan *et al.*, "Confidentiality-preserving rank-ordered search," in *Proc. ACM Workshop Storage Secur. Survivability*, 2007, pp. 7–12.

[15] S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, "Zerber$^{+R}$ : Top-k retrieval from a confidential index," in *Proc. 12th Int. Conf. Extending Database Technol., Adv. Database Technol.*, 2009, pp. 439–449.
[16] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling secure and effi-cient ranked keyword search over outsourced cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 8, pp. 1467–1479, Aug. 2012.
[17] P. Bailey, N. Craswell, I. Soboroff, and A.-P. de Vries, "The CSIRO enterprise search test collection," *ACM SIGIR Forum*, vol. 41, no. 2, pp. 42–45, 2007.
[18] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.
[19] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 829–837.
[20] S. Büttcher and C.-L. A. Clarke, "A security model for full-text file system search in multi-user environments," in *Proc. 4th Conf. USENIX Conf. FAST*, 2005, p. 13.
[21] W.-A. Taylor. (2000). *Change-Point Analysis: A Powerful New Tool for Detecting Changes*. [Online]. Available: http:// www.variation.com/cpa/tech/changepoint.html
[22] G. Box and T. Kramer, "Statistical process monitoring and feedback adjustment: A discussion," *Technometrics*, vol. 34, no. 3, pp. 251–267, 1992.
[23] G.-K. Zipf, *The Psycho-Biology of Language*. Oxford, U.K.: The MIT Press, 1965.



**Ke Li** received the B.S. degree from the Univer-sity of Science and Technology of China, in 2013, where he is currently pursuing the M.S. degree. His research interests include security, privacy and reliability in cloud computing.

**Weiming Zhang** received the M.S. and Ph.D. degrees from the Zhengzhou Information Science and Technology Institute, Zhengzhou, China, in 2002 and 2005, respectively. He is currently an Associate Professor with the School of Information Science and Technology, University of Science and Technology of China. His research interests include multimedia security, information hiding, and privacy protection.



**Ce Yang** received the B.S. degree from the University of Science and Technology of China, in 2011, where he is currently pursuing the Ph.D. degree. His research interests include security, privacy, and reliability in cloud computing.

**Nenghai Yu** received the B.S. degree from the Nanjing University of Posts and Telecommunications, in 1987, the M.E. degree from Tsinghua University, in 1992, and the Ph.D. degree from the University of Science and Technology of China, in 2004, where he is currently a Professor. His research interests include multimedia security, multimedia information retrieval, video processing, information hiding, and security, privacy, and reliability in cloud computing.