Secure Similarity Search

Hyun-A Park, Bum Han Kim, Dong Hoon Lee
CIST(Center for the information security technologies),
Graduate School of Information Management and Security,
Korea University
Email: kokokzi@cist.korea.ac.kr,
anewholic @cist.korea.ac.kr, donghlee@korea.ac.kr

Yon Dohn Chung
Department of Computer
Science and Engineering,
Korea University
Email:ydchung@korea.ac.kr

Justin Zhan
Carnegie Mellon CyLab Japan,
Carnegie Mellon Heinz School
Email:justinzh@andrew.cmu.edu

***Abstract*— One of the most substantial ways to protect users' sensitive information is encryption. This paper is about the keyword index search system on encrypted documents. It has been thought that the search with errors over encrypted data is impossible because 1 bit difference over plaintexts may reduce to enormous bits difference over cyphertexts. We propose a novel idea to deal with the search with errors over encrypted data. We develop two similarity search schemes, implement the prototypes and provide substantial analysis. We define security requirements for the similarity search over encrypted data. The first scheme can achieve perfect privacy in similarity search but the second scheme is more efficient.**

Keywords: index keyword search over encrypted data, similarity search, hamming distance

I. Introduction

When documents contain sensitive data, they are typically stored under encryption to protect privacy. This paper deals with the search system on encrypted data where a secure keyword index search protocol is provided. The protocol enables a legitimate querier to search the encrypted documents with the encrypted keyword in a server without decrypting them and revealing any information of the documents. The search system on encrypted data has been an active research area where a number of works have been produced.

Song et al. [34] proposed sequential scanning search over entire documents. Following this idea, the works have been focused on the keyword index search. Boneh et al. [4] developed the keyword search using a public key system. Chang et al. [18] proposed two index search schemes using the idea of pre-built dictionaries. Goh [13] proposed a secure index scheme, which formulates a security model for indexes known as semantic security (or no leakage of information) against adaptive chosen keyword attack (IND-CKA). Recently, Boneh et al. [11] proposed the public key systems supporting queries on encrypted data, using tokens produced by a secret key for testing any supported query predicate. They construct the systems for comparisons and subset queries as well as conjunctive versions of these predicates. However, this scheme does not consider the arithmetic operation, while Hacigumus et al. [23] proposed a method for range queries on encrypted data in DAS (Database As a Service) model using privacy homomorphism which allows basic arithmetic $(+, -, \times)$ over encrypted data. However, all of the above works about the search on encrypted data have not considered similarity search.

It is commonly thought that similarity search such as the search with errors over encrypted data is impossible because the search process over encrypted documents can be accomplished only through equality test. It is because that 1 bit difference over plaintexts may result in enormous bits difference over cyphertexts. For example, assume that a user mistypes 'abcd' into 'abcf'. In the general search on plaintexts, it can be searched for the form of 'abc*' or other similar form's keywords. In the search on the encrypted data, however, the result of one character's mistypo will be failure or preposterous. It motivates us to propose novel secure similarity search schemes which allow a limited number of 'errors' in the matches over encrypted data.

A. Key Idea and Contribution

For the purpose of the similarity search over encrypted data, we encrypt a keyword character by character. However, it is difficult to design a character-wise encryption algorithm since the domain is too limited. The total number of alphabets is only 26 so that the total number of outputs is also 26 independently of the encryption algorithm. It suffers dictionary attacks by an adversary.

To solve this problem, we design the encryption algorithm satisfying 'Cell Privacy' which means a character has different encryption values cell by cell. As the method for this, we encrypt a character together with all of the user's secret key, the document's identifier, and the field's identifier. Consequently, an adversary is not able to guess a character by observing the encrypted indexes. The dictionary attack by an adversary is impossible as well.

To achieve higher lever of security, it would be advisable to design the algorithm satisfying 'Query Privacy'. It makes the trapdoor by encrypting a keyword which a user want to search with a newly generated random value every query time. Even if the same character is queried repeatedly, no one can know the fact that it is the same character because of the newly generated random value every query time.

Based on these methods, we design two similarity searchable schemes over encrypted documents and define the security requirements for the similarity search over encrypted data. We then demonstrate that our first scheme SSS-I can guarantee the best security under our definition of 'Perfect Similarity

Search Privacy' which is explained in detail in Section III. The implementation of the prototypes shows that another scheme SSS-II is more efficient by accepting a little weaker security guarantee. The detailed methods are described in the following sections. In Section II, we address the settings of our schemes. We present the constructions of SSS-I and SSS-II in Section III and VI respectively. The implementation of our schemes is addressed in Section V. We provide our conclusion in Section IV.

II. Model

A. DAS model

Our scheme is based on DAS model [23], an instantiation of the computing model where clients are trusted and their data are stored at an untrustworthy server. We assume that an untrustworthy server is an adversary of our scheme. DAS model enables the system provider not to correctly interpret the data. The clients own the data, only have limited computational power and storage, and rely on the server for the mass computational power and storage. The server manages the clients' databases at the server. In our scheme, authorized clients are given encryption keys. The data is encrypted by the client before being sent to the server. The encryption key should not be given to any administrator, thus the server cannot decrypt the data. The queries are decomposed into client and server queries. The query sent to the server is executed against encrypted data. The results of the original query are obtained if the client decrypts it after implementing the server queries. Data privacy is assured under the conditions that the client does not share the encryption keys, the metadata or the unencrypted data with any party.

B. Similarity Search based on Approximate String Matching

As mentioned in Section I, we construct the similarity search schemes over encrypted documents. The similarity search is based on approximate string matching in this paper. We use 'hamming distance' as the approximate string matching method which defines the degree of similarity in our proposed scheme.

C. Algorithms

Our proposed schemes consist of six algorithms.

- **SysParam**(1^k)**:** Parameter generation algorithm $SysParam$ that is executed by the client takes an input as a secret parameter k and produces a system parameter λ.
- **KeyGen**(λ)**:** Key generation algorithm $KeyGen$ is executed by the client. It produces the user's search key set K.
- **IndGen**(K, W, d, J)**:** Index generation algorithm $IndGen$ takes the user's secret key K, a keyword list W, the document's identifier d, and the fields(column)'s identifier J and generates W's index I. It is done by the client.
- **Trapdoor**(K, w)**:** It is a trapdoor generation algorithm executed by the client. $Trapdoor$ takes a keyword w and the user's secret key K and returns the trapdoor T for w.
- **PattGen**(T, d)**:** Pattern generation algorithm $PattGen$ is executed by the server. It takes as input the trapdoor T and a document's identifier d and makes the pattern P for the similarity test.
- **SimMatch**(P, I)**:** Similarity string matching algorithm $SimMatch$ which is executed by the server takes as input the pattern P and an index I. It returns 'yes' if the similarity string matching test satisfies the similarity κ, or 'no' otherwise.

D. Premitives

Definition.1. Hamming Distance and Degree of Similarity

For two strings s and t, the Hamming distance $H(s,t)$ is defined as the number of places where the two strings differ, i.e., in number of characters. Accordingly, we define the degree of similarity as $H(s,t) \leq \kappa$, where $\kappa(\geq 0)$ is an integer [37].

Definition 2. PRF(Pseudo Random Function)

We say that '$F : K_f \times X \longrightarrow Y$ is (t, q, e) -secure pseudorandom function' if every oracle algorithm A making at most q oracle queries and with running time at most t has advantage $Adv_A < e$. The advantage is defined as $Adv_A =\mid Pr[A^{F_k} = 1] - Pr[A^R = 1] \mid$ where R represents a random function selected uniformly from the set of all maps from X to Y , and where the probabilities are taken over the choice of k and R [34].

Definition 3. PRG G_r(Pseudo Random Generator)

We say that '$Gr : K_{Gr} \longrightarrow S$ is a (t, e)-secure pseudorandom generator' if every algorithm A with running time at most t has advantage $Adv_A < e$. The advantage is defined as $Adv_A =\mid Pr[A(Gr(U_{K_{Gr}})) = 1] - Pr[A(U_S) = 1] \mid$. Where $U_{K_{Gr}}$, U_S are random variables distributed uniformly on K_{Gr}, S [34].

Definition 4. DDH (Decisional Diffie-Hellman)

Let G be a group of prime order q and g a generator of G. The DDH problem is to distinguish between triplets of the form (g^a, g^b, g^{ab}) and (g^a, g^b, g^c), where a, b, c are random elements of $\{1, ..., q-1\}$.

Consider the following experiment with a polynomial time adversary A : Flip a coin δ(to get 0 or 1), if $\delta = 1$, set $c = ab$, else choose c at random. The DDH problem is said to be hard if for any polynomial time adversary A, $|Pr(A(G, g^a, g^b, g^c) = \delta) - 1/2|$ is negligible.

E. Notions of Security

Definition 5. Security Game ICC (Indistinguishability of Ciphertext from Ciphertext)

It is the security proof model which gurantees semantic securrity(Indistinguishability) against CKA(Chosen Keyword Attack) and based on Goh [18] and Golle et.al.'s [20] security model.

- **Setup.** The challenger C creates a set W of q words $\in \{D_n\}$ and gives this to the adversary A. A chooses a polynomial number of subsets from W. This collection of subsets is called W^* and is returned to C. Upon receiving W^*, C runs algorithm **KeyGen** to generate the master key and encrypts each subset W^* running algorithm **IndGen**. Finally, C sends all indexes with their associated subsets to A.
- **Queries.** A is allowed to query C on a word w and receives the trapdoor T_w for w. With T_w, A can invoke algorithm **PattGen** and **SimMatch** on an index I to determine if $H(P_w, I) \leq \kappa$. (P_w is a pattern of the word 'w' for string matching test.)
- **Challenge.** After making some Trapdoor queries, A decides on a challenge by picking two keyword sets W_0, W_1, where $|W_0| = |W_1|$. A must not have queried the trapdoors for the keywords belonged to W_0 and W_1. Next, A gives W_0 and W_1 to C and C chooses $b \overset{\$}{\leftarrow} \{0,1\}$, invokes algorithm **IndGen** to obtain I_b for W_b, and returns I_b to A. The challenge for A is to determine b with the error rate $H(P_w, I) \leq \kappa$. After the challenge is issued, A is not allowed to query the trapdoors for the keywords belonged to W_0 and W_1 to C.
- **Response.** A eventually outputs a bit b', representing its guess for b. The advantage of A in winning this game is defined as $Adv_A = |Pr[b = b'] - 1/2|$, where the probability is over A and C's coin tosses. Also, adversary $A(t, \epsilon, q)$ is said to have an $\epsilon - advantage$ if $Adv_A > \epsilon$ after A takes at most t times and makes q trapdoor queries to the challenger.

For the purpose of simplicity, we define other variants of the security game ICC. In the first variant, the adversary chooses W_0 as well as a subset V of the keywords W_0. The challenger creates a document $W_1 = Rand(W_0, V)$. The goal of A is to distinguish between an encrypted index I_0 of W_0 and I_1 of W_1. It is called **Security Game ICR(Indistinguishability of Ciphertexts from Random)** and the detailed process is similar to ICC.

As the final security game, we consider an adversary who is able to distinguish between $W_o = Rand(W, V - \{w_t\})$ and $W_1 = Rand(W, V)$, for some keyword set W and query set V, and $w_t \in V$. It is called **Security Game ICLR(Indistinguishability of Ciphertexts from Limited Random)**. The detailed process is also similar to ICC and ICR.

Definition 6. Query Privacy

According to the security game ICC, given two words w_0 and w_1, we define the search scheme provides 'query privacy' if A cannot distinguish the trapdoors T_0 from T_1 for w_0 and w_1 with non-negligible advantage, for all polynomial time adversary A. The advantage is defined as $Adv_A = |Pr[Exp_A^{ind-cka-trp-1} = 1] - Pr[Exp_A^{ind-cka-trp-0} = 1]|$.

Definition 7. Index Privacy

According to the security game ICC, given two word-lists W_0 and W_1, we define the search scheme provides 'index privacy' if all polynomial time adversary A cannot distinguish the indexes I_0 from I_1 for the word-lists W_0 and W_1 with non-negligible advantage which is defined as $Adv_A = |Pr[Exp_A^{ind-cka-idx-1} = 1] - Pr[Exp_A^{ind-cka-idx-0} = 1]|$.

Definition 8. Cell Privacy

Under the condition that index privacy is satisfied, given two words $w_{i,0}$ and $w_{i,1}$ in a word-list W_i, we define the search scheme provides 'cell privacy' if all polynomial time adversary A cannot distinguish the indexes $I_{i,0}$ from $I_{i,1}$ for the words $w_{i,0}$ and $w_{i,1}$ with non-negligible advantage. The advantage is defined as $Adv_A = |Pr[Exp_A^{ind-cka-cell-1} = 1] - Pr[Exp_A^{ind-cka-cell-0} = 1]|$.

Definition 9. Perfect Similarity Search Privacy

It is defined that the search scheme provides 'perfect similarity search privacy' if both of the query privacy and cell privacy are achieved.

F. Notation

- n; the number of documents.
- m; the number of fields.
- D_i; the i-th document.
- W_i; the document D_i's keyword list
- $W_{i,j}$; the keyword of the j-th field in the i-th document, where $1 \leq i \leq n$ and $1 \leq j \leq m$. i.e. $W_i = \{W_{i,1}, W_{i,2}, ..., W_{i,m}\}$
- $w_{i,j}^s$; s-th character of the keyword $W_{i,j}$. i.e. $W_{i,j} = w_{i,j}^1 \| w_{i,j}^2 \| ... \| w_{i,j}^s$.
- s; the number of characters in the keyword
- d_i; the identifier of D_i which is randomly selected.
- j; the identifier of a field which is randomly selected.
- I_i; the index of W_i.
- $I_{i,j}$; the index of $W_{i,j}$ i.e. $I_i = I_{i,1}, I_{i,2}, ...I_{i,j}$

III. SSS-I

A. The construction of SSS-I

In this section, we construct our first Secure Similarity Search scheme, SSS-I. We use the Keyword field as in Golle et al.'s scheme [20], which consists of m keyword fields associated with each document. We assume that each field has a distinctive attribute so that the same keyword never appears in two different keyword fields. The matching information between each field and each attribute must be stored in the client to enable the queries by users. Our scheme SSS-I is constructed as follows:

1) **SysParam**(1^k)
It generates system parameter $\lambda = (G,\ g,\ f(\cdot),\ G_r,\ h(\cdot),\ HS(\cdot))$. G is a group of order q which is a large prime and g is a generator of a group G. $f : \{0,1\}^k \times \{0,1\}^* \rightarrow Z_q$ is a pseudo random function and G_r is a pseudo random generator. $h : \{0,1\}^* \rightarrow Z_q$ and $HS : \{0,1\}^* \rightarrow \{0,1\}^k$ are one way hash functions.

2) **KeyGen**(λ)
Key generation algorithm $KeyGen(\lambda)$ outputs a users' search key set $K \in \{0,1\}^k$ for index encryption.

3) **IndGen**(K, W, d, J)
It takes as input a user's secret key k_u, a keyword list W_i=$\{W_{i,1}, W_{i,2}, ..., W_{i,m}\}$, a document identifier d_i, and fields(columns)' identifier set $J = \{j_1, j_2, ..., j_m\}$, where $1 \leq i \leq n$ and $W_{i,j}$=$w^1_{i,j} w^2_{i,j} ... w^s_{i,j}$. It processes as follows.

   a) For the keyword $W_{i,j}$, compute;

   $$F_{K_u}(W_{1,1}) = f_{k_u}(w^1_{1,1}|1) \| f_{k_u}(w^2_{1,1}|1) \| ... \| f_{k_u}(w^s_{1,1}|1)$$
   $$F_{K_u}(W_{1,2}) = f_{k_u}(w^1_{1,2}|2) \| f_{k_u}(w^2_{1,2}|2) \| ... \| f_{k_u}(w^s_{1,2}|2)$$
   ...
   $$F_{K_u}(W_{i,j}) = f_{k_u}(w^1_{i,j}|j) \| f_{k_u}(w^2_{i,j}|j) \| ... \| f_{k_u}(w^s_{i,j}|j)$$

   b) Compute $I_{i,j}$;

   $$I_{i,j} = HS(g^{d_i h(k_u) f_{k_u}(w^1_{i,j}|j)}) \| HS(g^{d_i h(k_u) f_{k_u}(w^2_{i,j}|j)}) \| ... \| HS(g^{d_i h(k_u) f_{k_u}(w^s_{i,j}|j)})$$

   c) Produce an index I_i;

   $I_i = id_{i,1}, id_{i,2}, I_{i,1}, I_{i,2}, ... I_{i,m}$
   (where, $id_{i,1} = g^{d_i h(k_u)}$, $id_{i,2} = g^{-d_i}$. These are the document D_i's identifiers.)

4) **Trapdoor**(K, w)
It takes a user's secret key k_u and the keyword $W = w^1 w^2 .. w^s$ that a user wants to search. It generates the trapdoor for the keyword W as follows.

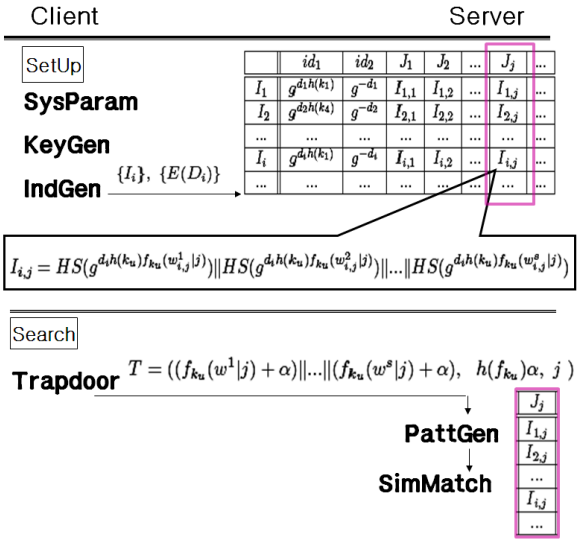

Fig. 1. System Process of SSS-I

$$T = (T_1, T_2, T_3)$$
$$= ((f_{k_u}(w^1|j) + \alpha) \| ... \| (f_{k_u}(w^s|j) + \alpha),\quad h(f_{k_u})\alpha,\ j\ (the\ field\ identifier))$$

$T_1 = (f_{k_u}(w^1|j) + \alpha) \| ... \| (f_{k_u}(w^s|j) + \alpha) = t^1 \| ... \| t^s$
α; the random value which is generated newly every query time by pseudo random generator.

5) **PattGen**(T, I)
It takes the trapdoor T and the document's index I_i and generates the pattern P for string matching. For j-th $I_{i,j}$ of each index I_i;

$$HS((id_{i,1})^{t^1} \cdot (id_{i,2})^{T_2}) \| ... \| HS((id_{i,1})^{t^s} \cdot (id_{i,2})^{T_2})$$

$$= HS((g^{d_i h(k_u)})^{(f_{k_u}(w^1|j)+\alpha)} \cdot (g^{-d_i})^{(h(f_{k_u})\alpha)}) \|$$
$$... \| HS((g^{d_i h(k_u)})^{(f_{k_u}(w^s|j)+\alpha)} \cdot (g^{-d_i})^{(h(f_{k_u})\alpha)})$$

$$= HS(g^{d_i h(k_u) f_{k_u}(w^1|j)}) \| ... \| HS(g^{d_i h(k_u) f_{k_u}(w^s|j)})$$
$$= P$$

6) **SimMatch**(P, I)
As input, it takes pattern P and j-th field's index $I_{i,j}$. It computes the hamming distance $H(P,\ I_{i,j})$ between pattern string P and index string $I_{i,j}$ for each I_i. If $H(P,\ I_{i,j}) \leq \kappa$, return 'yes' or 'no' otherwise.

B. The Security Analysis of SSS-I

Theorem 1. SSS-I can provide 'Query Privacy' according to the game ICC if G_r is (t, e)-secure pseudo random generator.

Proof. We prove it with contraposition. We assume that SSS-I cannot provide 'query privacy' of the Definition 6. according to the security game ICC. Then, there exists an algorithm $A(t, \epsilon, q)$ which wins the game ICC. We construct an algorithm β which can solve the problem about whether G_r is pseudo random or random generator with non-negligible probability. β uses an algorithm A as a subroutine and can query the oracle $\mathcal{O}_{G_r}$ so that it can obtain the value of random number α_q. Algorithm β simulates algorithm A as follows;

- **Setup.** Algorithm β creates a set W of q words $\in \{D_n\}$ and gives this to the adversary A. A chooses a polynomial number of subsets $\{W_i\}$ from W. We call this the collection of subsets $W^* = \{W_i\}$, where $W_i = \{W_{i,1}, W_{i,2}, .., W_{i,l}\}$. A sends them to β again. Upon receiving W^*, β runs algorithm **KeyGen** to generate search keys $K = \{k_u\}$ and invokes algorithm **IndGen** for keywords set $W_i \in W^*$. Where, an index string I_i is produced and an unique identifier d_i for each document D_i is assigned. After producing all the index stirings, β sends them to A.
- **Queries.** If A queries the trapdoor for a word $W_{i,j} \in W_i$, β runs algorithm **Trapdoor** for a word $W_{i,j}$ and produces $T_{i,j}$ and sends it to A. $T_{i,j}$ is obtained through the queries to the oracle $\mathcal{O}_{G_r}$ and again with $T_{i,j}$ as input value, A runs algorithm **PattGen** and **SimMatch** on each index $I_{i,j}$ $(1 \leq i \leq n)$ to determine if $H(P_w, I_{i,j}) \leq \kappa$. P_w is a pattern of the word $W_{i,j}$ for string matching test.
- **Challenge.** After making some queries, A chooses distinctive $W_{i,0},\ W_{i,1} \in W_i$. A cannot query the trapdoor for $W_{i,0},\ W_{i,1} \in W_i$ any more. Next, A gives $W_{i,0}$ and $W_{i,1}$ to β and β chooses $b \overset{\$}{\leftarrow} \{0, 1\}$, and then invokes algorithms **Trapdoor** for $W_{i,b}$. The result of this, $T_{i,b}$ is returned to A.
- **Response.** A finally outputs a bit b', representing its guess for b. If $b = b'$, then β outputs 1. Otherwise, β outputs 0. 'β outputs 1' means that G_r is a pseudo random generator.

We show that β can solve the problem about whether G_r is pseudo random or random generator with non-negligible probability. Accordingly, the advantage of β in winning this experiment is;

$$Adv_\beta = Pr[Exp_\beta^{PR} = 1] = Pr[b' = b]$$

$$= Pr[b' = b|b = 1] \cdot Pr[b = 1] + Pr[b' = b|b = 0] \cdot Pr[b = 0]$$

$$= Pr[b' = b|b = 1] \cdot \frac{1}{2} + Pr[b' = b|b = 0] \cdot \frac{1}{2}$$

$$= Pr[b' = 1|b = 1] \cdot \frac{1}{2} + Pr[b' = 0|b = 0] \cdot \frac{1}{2}$$

$$= Pr[b' = 1|b = 1] \cdot \frac{1}{2} + (1 - Pr[b' = 1|b = 0]) \cdot \frac{1}{2}$$

$$= \frac{1}{2} + \frac{1}{2}(Pr[Exp_A^{ind-cka-1} = 1] - Pr[Exp_A^{ind-cka-0} = 1])$$

$$= \frac{1}{2} + \frac{1}{2} Adv_A^{ind-cka} = \frac{1}{2} + \frac{1}{2}\epsilon$$

Theorem 2. SSS-I can provide 'Index Privacy' according to the game ICC in random oracle model if DDH problem is hard.

For example, the indexes for a common keyword $W = w^1 w^2$ of D_1 and D_2 of a user u, are $HS(g^{d_1 h(k_u) f_{k_u}(w^1 \| j)}) \| HS(g^{d_1 h(k_u) f_{k_u}(w^2 \| j)})$ and $HS(g^{d_2 h(k_u) f_{k_u}(w^1 \| j)}) \| HS(g^{d_2 h(k_u) f_{k_u}(w^2 \| j)})$. They have different values respectively because of unique identifiers d_1 and d_2 and field identifier j. Hence, although two indexes are for a common keyword, an adversary cannot know the fact.

Proof. According to [20], the existence of an adversary that wins the game ICC with non-negligible probability implies the existence of an adversary that wins the game ICLR with non-negligible probability. Let A be an adversary that wins the game ICLR with advantage ϵ. We construct an adversary Δ that uses A as a subroutine and breaks DDH with non-negligible advantage.

Δ invokes the algorithms **SysParam** and **KeyGen**. Let $g^\delta, g^\tau, g^\gamma$ be a Diffi-Hellman triplet, the challenge is to determine whether $\gamma = \delta\tau$. Δ guesses a value $w_{i,j}^z$ for the character $w_{i,j}^t$ that A will choose in the game ICLR, by picking $w_{i,j}^z$ uniformly independently at random in $\{W_1, W_2, ..., W_m\}$.

Δ simulates the algorithm **IndGen** as follows. Δ associates every keyword $W_{i,j} = w_{i,j}^1 w_{i,j}^2 ... w_{i,j}^s$ with a random value $X_{i,j} = x_{i,j}^1 x_{i,j}^2 ... x_{i,j}^s$. Also search key K with K', and $g^{K'}$ with g'. For $W_i = \{W_{i,1}, W_{i,2}, ..., W_{i,m}\}$, Δ chooses a random value δ_i for each W_i and outputs;

$$\begin{aligned} I_i = (&g^{\delta_i K'}, g^{-\delta_i}, HS(g^{\delta_i x_{i,1}^1 K'}) \| ... \| HS(g^{\delta_i x_{i,1}^s K'}), ..., \\ &HS(g^{\delta_i x_{i,z}^1 K'}) \| .. \| HS((g^\tau)^{\delta_i x_{i,z}^s K'}), ..., \\ &HS(g^{\delta_i x_{i,m}^1 K'}) \| .. \| HS(g^{\delta_i x_{i,m}^s K'}) \\ = &((g')^{\delta_i}, g^{-\delta_i}, ..., HS((g')^{\delta_i x_{i,z}^1}) \| ... \| HS(((g')^\tau)^{\delta_i x_{i,z}^s}), ... \\ &..., HS((g')^{\delta_i x_{i,m}^1}) \| ... \| HS((g')^{\delta_i x_{i,m}^s})) \end{aligned}$$

If A queries for a keyword $W_{i,j}$, Δ outputs the trapdoor T_w and runs algorithm **PattGen** and **SimMatch**.

Finally, A selects a challenge keyword set $W_i = \{W_{i,i}, .., W_{i,m}\}$ along with a query set $V \subseteq W_i$ and a character $w_{i,j}^t \in V$. If $w_{i,j}^z \neq w_{i,j}^t$, Δ returns a random value in reply to the DDH challenge. With the probability about $1/m \times s$, we have $w_{i,j}^z = w_{i,j}^t$ and that case Δ proceeds as follows. Let $I_{i,j}^t = HS(((g')^\tau)^{\delta_i x_{i,j}^t})$. For $W_{i,j} \in V$, $w_{i,j}^s \neq w_{i,j}^t$, let $I_{i,j}^s = R_{i,j}^s$(random value). For $W_{i,j} \notin V$, let $I_{i,j} = HS(g^{\delta_i x_{i,j}^1 K'}) \| ... \| HS(g^{\delta_i x_{i,j}^s K'})$. Δ returns to A the following; $I_i = ((g')^{\delta_i}, g^{-\delta_i}, I_{i,1}, .., I_{i,m})$.

Check that this index is an encryption of W in every keyword $W_{i,j} \notin V$. If $\gamma = \delta\tau$, this index is also an encryption of W_i for a character $w_{i,j}^t$; otherwise it is not.

Δ is again allowed to ask for index of keyword sets, with the restriction that A may not make a query that are distinguishing $W_o = Rand(W_i, V - \{w_{i,j}^t\})$ from $W_1 = Rand(W_i, V)$.

Finally, A outputs a bit b'. If $b' = 0$, Δ guesses that $g^\delta, g^\tau, g^\gamma$ is not a DDH triplet. If $b' = 1$, Δ guesses that $g^\delta, g^\tau, g^\gamma$ is a DDH triplet. Since the encryption will be random for character $w_{i,j}^s$ if and only if the challenge is not a DDH tuple, Δ solves the DDH challenge with the same advantage that A has in winning game ICLR.

Theorem 3. Under the condition that 'Index Privacy' is achieved, SSS-I can provide 'Cell Privacy' according to the game ICC if f is pseudo random function.

By the assumption that 'Index Privacy' is achieved, the same character in different documents has the different index value for each document. It is because that each index is encrypted with the distinctive document's identifier.

Next, we consider the same character in different fields of a document. Assume that the same character is 'a'. 'a' has the different index value for each field in the document since 'a' is encrypted with the distinctive field's identifier. Hence, the character or the keyword has the different values in the different cell.

The proof can be done similarly to Theorem 1. For simplification, we consider only a character because it can imply the case of a keyword. In the stage of challenge, select the distinctive fields j_o, j_1 where 'a_{i,j_o}' and 'a_{i,j_1}' are encrypted to generate the index value. Then the challenge is to determine b of $I_{i,b}$ for a_{i,j_b}. We leave out the detailed proof since the rest are almost the same as the proof of Theorem 1.

Theorem 4. SSS-I can provide 'Perfect Similarity Search Privacy' if it guarantees both of the 'Query Privacy' and 'Cell Privacy'.

By Theorem 1 and 3, it is clear that SSS-I can provide 'Perfect Similarity Search Privacy'.

Note 1. Our scheme is firstly proposed so that there is no scheme to compare with. Hence, we implement the prototype and check the correctness and efficiency of our proposed scheme in Section V.

Note 2. In SSS-I, α in the trapdoor generation equation, which is a randomly generated number every query time, is removed in the pattern formative equation. These equations enable us not to have to update all documents every query time such as in Golle et al.'s scheme [20] where a random number is generated every query time and all documents in a server have to be updated every query time to prohibit any information from being leaked by the accumulated results. However, our proposed scheme SSS-I can leave out such inefficient processes by forming the efficient trapdoor and pattern equations and at the same time it can keep the same level of security as Golle et al.'s scheme.

IV. SSS-II

A. Construction of SSS-II

Our second scheme SSS-II also uses m keyword fields but there is no rule that each field has a specific attribute. Therefore, the clients do not have to store the matching information between each field and each attribute. Since SSS-II has the same algorithms **SysParam**(1^k) and **KeyGen**(λ) as SSS-I, we start from the algorithm **IndGen**(K, W, d, J).

1) **IndGen**(K, W, d, J)
It takes as input a user's secret key k_u, a keyword list W_i=$\{W_{i,1}, W_{i,2}, ..., W_{i,m}\}$, a document identifier d_i, and fields(columns)' identifier set $J = \{j_1, j_2, ..., j_m\}$, where $1 \le i \le n$ and $W_{i,j}$=$w_{i,j}^1 w_{i,j}^2 ... w_{i,j}^s$. It processes as follows.

   a) For keywords $W_{i,j}$, compute;
   $$F_{K_u}(W_{1,1}) = f_{k_u}(w_{1,1}^1|1+1)\|f_{k_u}(w_{1,1}^2|2+1)\|.... \\ ...\|f_{k_u}(w_{1,1}^s|s+1) = f_{1,1}^1\|f_{1,1}^2\|...\|f_{1,1}^s$$
   $$F_{K_u}(W_{1,2}) = f_{k_u}(w_{1,2}^1|1+2)\|f_{k_u}(w_{1,2}^2|2+2)\|.... \\ ...\|f_{k_u}(w_{1,2}^s|s+2) = f_{1,2}^1\|f_{1,2}^2\|...\|f_{1,2}^s$$
   ...
   $$F_{K_u}(W_{i,j}) = f_{k_u}(w_{i,j}^1|1+j)\|f_{k_u}(w_{i,j}^2|2+j)\|.... \\ ...\|f_{k_u}(w_{i,j}^s|s+j) = f_{i,j}^1\|f_{i,j}^2\|...\|f_{i,j}^s$$

   b) Compute $I_{i,j}$;
   $$I_{i,j} = HS(f_{i,j}^1|d_i)\|HS(f_{i,j}^2|d_i)\|..\|HS(f_{i,j}^s|d_i)$$

   c) Produce an index I_i;
   $$I_i = d_i, I_{i,1}, I_{i,2}, ...I_{i,m} \\ = d_i,\ HS(f_{i,1}^1|d_i)\|...\|HS(f_{i,1}^s|d_i), \\ HS(f_{i,2}^1|d_i)\|...\|HS(f_{i,2}^s|d_i), ... \\ ...., HS(f_{i,m}^1|d_i)\|..\|HS(f_{i,m}^s|d_i)$$

2) **Trapdoor**(K, w)
It takes a user's secret key k_u and the keyword $W = w^1 w^2 .. w^s$ that a user want to search. It generates m trapdoors for the keyword W as follows. (m is the total number of fields.)
   $$T = (T_1,\ T_2, ..., T_m)$$
   $$T_1 = f_{k_u}(w^1|1+1)\|f_{k_u}(w^2|2+1)\|.... \\ ...\|f_{k_u}(w^s|s+1) = t_1^1\|t_1^2\|...\|t_1^s$$
   $$T_2 = f_{k_u}(w^1|1+2)\|f_{k_u}(w^2|2+2)\|....$$

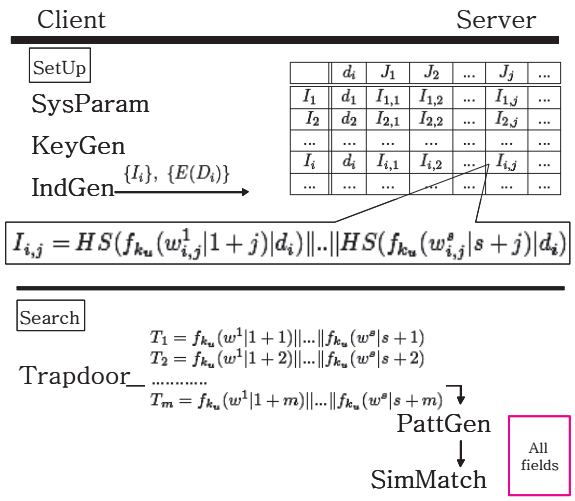

Fig. 2. System Process of SSS-II

$$...\|f_{k_u}(w^s|s+2) = t_2^1\|t_2^2\|...\|t_2^s$$

.....

$$T_m = f_{k_u}(w^1|1+m)\|f_{k_u}(w^2|2+m)\|....$$
$$...\|f_{k_u}(w^s|s+m) = t_m^1\|t_m^2\|...\|t_m^s$$

3) **PattGen**(T, I)
It takes trapdoor T and the document's index I_i and generates the pattern $P = (P_1,\ P_2, .., P_m)$ for string matching.

$$T_1;\ HS(t_1^1|d_i)\|HS(t_1^2|d_i)\|...\|HS(t_1^s\|d_i) = P_1$$

$$T_2;\ HS(t_2^1|d_i)\|HS(t_2^2|d_i)\|...\|HS(t_2^s\|d_i) = P_2$$

.....

$$T_m;\ HS(t_m^1|d_i)\|HS(t_m^2|d_i)\|...\|HS(t_m^s\|d_i) = P_m$$

4) **SimMatch**(P, I)
As input, it takes pattern $P = (P_1,\ P_2, .., P_m)$ and index I_i. It computes the hamming distance $H(P_1,\ I_{i,1})$, $H(P_2,\ I_{i,2})$,...,$H(P_m,\ I_{i,m})$ respectively for each I_i. If at least one of them satisfies the condition $H(P_j,\ I_{i,j}) \leq \kappa$, return 'yes' or 'no' otherwise.

B. Analysis

The index formation, i.e., the encryption requirement for cell privacy of SSS-II is to include all values of the user's secret key, the identifier of the document, and the identifier of the field. In addition to these, the character's ordering number, by which even the same character in the same cell has different encryption value. Anyway, we can say SSS-II can provide 'Cell Privacy'. However, it cannot provide 'Query Privacy' because it does not use random factor in the trapdoor generation. For this reason, one attack may be supposed as follows.

We assume that the malicious server manager stores the queried trapdoors and their results matched with 'yes'. By using those trapdoor values through a considerable number of queries, the server manager can produce another database. It is filled with the trapdoor values matched with 'yes' results. However, the problem is that the trapdoor value does not include the document's identifier. This means that it cannot satisfy 'Cell Privacy'. Accordingly, within a field, the server manager can obtain the total of $26 \times the\ number\ of\ users \times the\ number\ of\ characters(i.e.\ the\ length\ of\ the\ keyword)$ encryption values.

For example, the 1st field 's trapdoors have the form as like this; $f_{k_u}(w^1|1+1)\|f_{k_u}(w^2|2+1)\|...\|f_{k_u}(w^s\|s+1)$. The diversity of these values depends on users' secret keys and the length of the keywords while other factors, the field's identifier and the ordering number of each character are same. There is no document identifier. Since it is a character-wise encryption, the total number of distinctive encryption values is $26 \times the\ number\ of\ users \times\ the\ length\ of\ the\ keywords$. We assume that the number of users is 1000. Since the length of the keywords is various keyword by keyword, we assume the average length is 'e'. Then the adversary can obtain $26,000 \times e$ distinctive encryption values. Each alphabet can take $1,000 \times e$ values among $26,000 \times e$ encryption values. Hence, the probability that an adversary can success in a dictionary attack is $1/({}_{26,000\times e}C_{1,000\times e})^{26}$, i.e. negligible. The formulation of an adversary's success probability is $1/({}_{26\times u\times e}C_{u\times e})^{26}$ if the number of users is u and the average length of the keywords is e. Consequently, the security of SSS-II depends on the number of users and the length of the keywords.

As we mentioned before, the specific attribute is not assigned to each field in SSS-II. It makes the 'trade-off' happen. The positive aspect is that SSS-II has no information leaked by the identifiers of the fields in the trapdoors, where the identifier may be enough to allow the server to infer unintended information about the documents. The negative is inefficiency. No attribute-assignment for each field make us generate m(the number of trapdoors) trapdoors for one keyword. This enforces the server to produce m patterns and to test the similarity between m patterns and m indexes respectively. If SSS-II use the specific m keyword fields such as SSS-I, the performance will be more efficient. We will provide the prototype implementation in next section.

V. Prototype Implementation

In this section, we describe the implementation of our proposed schemes. It processes the transactions on Intel Core 2 Duo 2.13 GHz processor with 2 GB RAM. We use Oracle 10g as a database system and OCCI (Oracle C++ Call Interface) as the interface between Oracle DBMS and SSS client to reduce the interface latency. We use OpenSSL cryptography modules for cryptographic operations such as SHA-1, AES, and Elliptic Curve operation. Since an exponentiation calculation is very heavy and the size of a group element is generally long, we use a group over 'Elliptic Curve' to solve that problem. We

TABLE I

IMPLEMENTATION ENVIRONMENT AND PARAMENTERS

Agent	Processor	Intel Core 2 Duo 2.13 GHz
	RAM	2 GB
	Language	C++
	Crypto. Eng.	OpenSSL 0.9.8e
Database	Product	Oracle 10g
	Interface	OCCI (Oracle C++ Call Interface)
Cryptographic Parameter	Curve	WTLS Curve 3
	Size	163 bits
	Hash Function	SHA-1 (160 bits)
	PRF	AES (128 bits)
Data Set	# of MAX keywords (=m)	10
	# of MAX characters (=s)	15
	# of documents	$500/1,000/5,000/$ 10,000 / 50,000 / 100,000
	# Hamming distance $H(P,I) \leq \kappa$	$\kappa = 1$

use 'Koblitz curve', where the underlying field $GF(2^{163})$ is defined by generating the polynomial $x^{163}+x^7+x^6+x^3+x+1$. This curve has been used in many standards and identified in WAP WTLS standard as WTLS Curve 3. As for the degree of similarity, Hamming Distance $H(P,I) \leq \kappa$, we set κ as 1. The detailed implementation parameters are presented in Table 1.

To benchmark the performance of SSS-I and SSS-II, we make a data set then estimate the time which is required for the search process of our two schemes. Firstly, we build two tables for SSS-I and SSS-II and select random keywords using the standard C language random function $rand()$ where $s = 15$ and $m = 10$. Next, SSS generates the index strings $I = \{I_{i,j}\}$ related to the keywords then inserts the document into the database. We repeat this procedure $100,000$ times, *i.e.*, $100,000$ documents are inserted into the database. At last, if a user requests a document with the specific keyword, SSS client performs the search process with a server. We repeat the experiment with respect to the data set size. The experimental results of the search process using SSS-I and SSS-II are shown in Figure 1. For example, SSS-I search process takes about 6.6 sec and SSS-II 0.73 sec for 10,000 documents. In addition, we experiment the modified version SSS-IIM which is the SSS-II using the specific m keyword fields such as SSS-I. Since in SSS-IIM, the server only has to search for the field queried in the trapdoor, it takes about 0.58 sec.

To check the correctness of our scheme, we make four test files. Each of them includes the keywords (complement), (compliment), (complement and compliment), and (complemente) respectively. Those files are inserted into the database which is generated with random function. The first we query the 'complement' and then 'compliment'. The first result is all of the four test files and the second one is the former three files including (complement), (compliment), and (complement and compliment) respectively. It shows the correctness of our scheme.

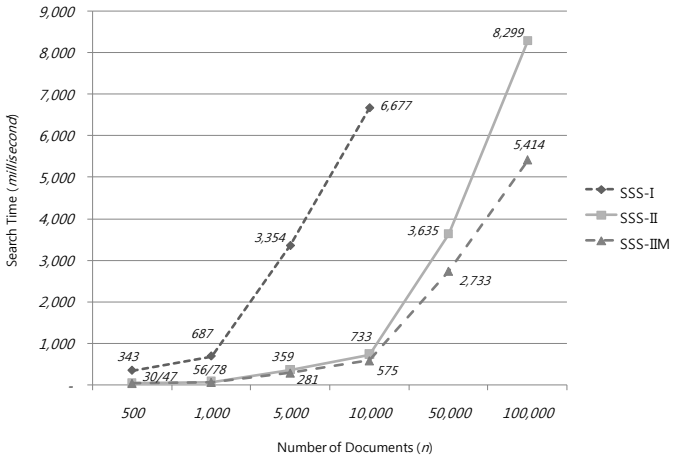

Fig. 3. Performance of the Search Process

Consequently, SSS-II scheme is more efficient then SSS-I since the exponentiation calculation of SSS-I requires heavy computational overhead. Instead, SSS-I can achieve the 'Perfect Similarity Search Privacy' while SSS-II cannot provide 'Query Privacy'. We expect that if the SSS-I and SSS-II schemes are internally supported in the DBMS, the performance of the search process can be greatly improved.

VI. CONCLUSION

In an information-oriented society, the management of a database which is a storage of sensitive data is one of the most important factors. To protect users' private information, the encryption is commonly utilized. However, the encryption method introduces some negative effects such as inefficiency and malfunctioning. Thus, it has been believed that similarity search with errors over encrypted data is impossible.

In this paper, we open a new page in this area. We design two similarity searchable schemes over encrypted document, implement the prototype, and analyze the security and efficiency. SSS-I can meet the 'Perfect Similarity Search Privacy' which we define as the best security and SSS-II is more efficient by accepting a little weaker security guarantee than SSS-I. It is not desirable that the encryption for privacy hinders the performance of database's other general operations. As future work, we would like to investigate how to further incorporate our searching algorithms into database management systems.

REFERENCES

[1] M. Abdalla, M. Bellare, D.Catalano, E.Kiltz, T.Kohno, T.Lange, J.Malone-Lee, G.Neven, P. Paillier, and H.a Shi, Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions, Crypto05, LNCS 3621, pp.205-222, 2005

[2] M. J. Atallah1, K.B. Frikken1, M.T. Goodrich, and R. Tamassia, Secure Biometric Authentication for Weak Computational Devices, FC 2005, LNCS 3570, pp. 357.371, 2005

[3] R. Baeza-Yates, Text retrieval: Theory and practice, In 12th IFIP World Computer Congress, Elsevier Science, vol. I, pp. 465-476, 1992

[4] D.Boneh, G.D.Crescenzo, R.Ostrovsky, and G.Persiano, Public-key encryption with keyword search, Eurocrypt04, LNCS, Springer-Verlag, 2004

[5] J. Bethencourt, H. Chan, A. Perrig, E. Shi, and D. Song, Anonymous multi-attribute encryption with range query and conditional decryption, Technical report, CMU-CS- 06-135, 2006.
[6] K.Bennett, C.Grothoff, T.Horozov, I.Patrascu, Efficient sharing of encrypted data, ACISP02, 2002
[7] L.Ballard, M.Green, B.de Medeiros, F.Monrose, Correlation-Resistant Storage via Keyword-Searchable Encryption, SPAR Technical Report TR-SP-BGMM-050705
[8] L.Ballad, S.Kamara, and F.Monrose, Achieving efficient conjunctive keyword searches over encrypted data, ICICS 2005, LNCS 3783, pp.414-426, 2005
[9] J.W.Byun, D.H.Lee, and J.I. Lim, Efficient conjunctive keyword search on encrypted data storage system, EuroPKI 2006, LNCS 4043 pp.184-196, 2006
[10] R. Baeza-Yates and G.Navarro, Faster approximate string matching, Algorithmica 23, pp. 127-158, 1999
[11] D. Boneh, B. Waters, Conjunctive, Subset, and Range Queries on Encrypted Data, In the Proceedings of TCC 07, 2007.
[12] B.Chor, N.Gilboa, M.Naor, Private Information Retrieval by Keywords, Technical Report TR CS0917, 1997
[13] Y.C.Chang and M.Mitzenmacher, Privacy preserving keyword searches on remote encrypted data, Cryptology ePrint Archive, 2004
[14] G. Cornode and S. Muthukrishnan, The String Edit Distance Matching Problem With Moves, ACM Transactions on Algorithms, Vol. 3, No. 1, Article 2, 2007
[15] Sun S.Chung and Gultekin Ozsoyoglu, Processing Aggregation Queries over Encrypted Databases, ICDE 2006, LNCS, 2006.
[16] J. Domingo-Ferrer, A new privacy homomorphism and applications, Information Processing Letters, 1996.
[17] J. Domingo-Ferrer, A Provably Secure Additive and Multiplicative Privacy Homomorphism, ISC, LNCS 2433, pp. 471-483, 2002.
[18] Eu-Jin Goh, Secure Indexes, Cryptology ePrint Archive, 2004
[19] Tingjian Ge, Stan Zdonik, Fast, Secure Encryption for Indexing in a Column-Oriented DBMS, Proceedings of the 23nd International Conference on Data Engineering pp.676-685 , ICDE 2007.
[20] P.Golle, J.Staddon, B.Waters, Secure Conjunctive Keyword Search Over Encrypted Data, ACNS04, 2004
[21] H. Hacigumus, B. Iyer, and S. Mehrotra, Query Optimization in Encrypted Database Systems, DASFAA 2005, LNCS 3453, pp. 43.55, 2005.
[22] Hakan Hacigumus, Balakrishna R. Iyer, Li Chen, Sharad Mehrotra, Executing SQL over Encrypted Data in the Database-Service-Provider Model, In the Proceedings of ACM SIGMOD, June, 2002.
[23] H.Hacigumus, B.Iyer, and S.MehrotraEfficient, Execution of Aggregation Queries over Encrypted Relational Databases, DASFAA2004, LNCS 2793, pp.125-136, 2004
[24] Einar Mykletun and Gene Tsudik, Aggregation Queries in the Database-As-a-Service Model, In the proceedings of DBSEC 2006.
[25] G.Navarro, A guided tour to approximate string matching. ACM Computing Surveys 33(1), pp. 31-88, 2001.
[26] G.Navarro and R. Baeza-Yates, Matchsimile: A Flexible Approximate Matching Tool for Searching Proper Names, Journal of the American society for Information Science and Technology, 2003
[27] Ogata and K. Kurosawa, Oblivious Keyword Search, Journal of Complexity, Volume 20, pp.356 - 371, 2004
[28] G.Ozsoyoglu, D.Singer, S.Chung , Anti-Tamper Databases: Querying Encrypted Databases, In the Proceedings of IFIP 11.3 Conference 2003 on Database Security, August 3 - 6.
[29] Hyun-A. Park, J.W.Byun, and D.H.Lee, Secure Index Search for Groups, TrustBus 2005, LNCS3592 pp.128-140, 2005
[30] D.Park, K.Kim, and P.Lee, Public Key Encryption with Conjunctive Field Keyword Search, WISA 2004, LNCS 3325, pp.73-86, 2004
[31] H.S.Rhee, J.W.Byun, and D.H.Lee, Oblivious Conjunctive Keyword Search, WISA2005, LNCS3886, pp.318-327, 2006
[32] Ontario, Office of the Information and Privacy Commissioner (IPC) and Netherlands Registratiekamer (1995), Privacy-Enhancing Technologies: The Path to Anonymity, Information and Privacy Commissioner and Registratiekamer, at: http://www.ipc.on.ca/english/pubpres/papers/anon-e.htm
[33] R.Sion and B.Carbunar, Conjunctive keyword search on encrypted data with completeness and computational privacy, Cryptology ePrint Archive, 2005
[34] D.Song, D.Wagner, and A.Perrig, Practical techniques for searches on encrypted data, IEEE Symposium on Security and Privacy, pp.44-55, 2000
[35] A. Takasu, An Approximate Multi-word Matching Algorithm for Robust Document Retrieval, CIKM06, ACM 1-59593-433-2/06/0011, 2006
[36] B.Waters, D.Balfanz, G.Durfee, and D.Smetters, Building an encrypted and searchable audit log, NDSS Symposium, pp.205-214, 2004
[37] $http://www.cut-the-knot.org/do_you_know/Strings.shtml$, $Distance$