F⁵P⁵: Keyword Search over Encrypted Data with Five Functions and Five Privacy Assurances

Huimin Shuai^{1,2} and Wen Tao Zhu¹

¹ State Key Laboratory of Information Security
Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
² Graduate University of Chinese Academy of Sciences, Beijing, China
hmshuai@is.ac.cn, wtzhu@ieee.org

Abstract. Breaches of data security and privacy have prompted concerns about data outsourcing. Encryption is an ideal solution, yet searching over encrypted data is a challenging task. Traditional data retrieval no longer hold, so searchable encryption (SE) techniques rise in response. Most of the related efforts are narrowly focused and problem-specific. None of them has *simultaneously* achieved the five keyword search functions (F⁵): *fuzzy*, *multi-keyword*, and *ranked search* as well as *keyword addition/removal* and *instantaneous search revocation*, together with the five privacy preserving requirements (P⁵): *keyword privacy*, *index privacy*, *token privacy*, *search pattern privacy*, and *access pattern privacy*. In this paper, a full-featured approach called F⁵P⁵ is presented, which achieves F⁵ and P⁵ simultaneously. Analysis and experiments show that F⁵P⁵ is secure and privacy preserving, and provides high-precision search results and is efficiency in terms of computation and storage.

Keywords: Data outsourcing, security, privacy, searchable encryption.

1 Introduction

Data outsourcing promises an economic paradigm of data sharing such as Dropbox. Despite of all the hype, this paradigm deprives data owners of direct control over outsourced data and prompts concerns about potential data breaches on the service provider (SP) side. Searchable encryption (SE) is an ideal solution to protecting sensitive data, yet searching over encrypted data without touching the plaintext is a challenging task, which requires that the index for each encrypted file be elaborately encrypted in a searchable way. Many efforts have been made. However, none of them has *simultaneously* achieved the five keyword search functions (F⁵): *fuzzy*, *multi-keyword*, and *ranked search* as well as *keyword addition/removal* and *instantaneous search revocation*, together with the five privacy preserving requirements (P⁵): *keyword privacy*,

* This work was supported by the National Natural Science Foundation of China under Grants 60970138 and 61272479, and also by the Strategic Priority Research Program of Chinese Academy of Sciences under Grant XDA06010702.

Table 1. Overview of related work regarding F⁵ and P⁵

Scheme	Fuzzy search	Multi-keyword search	Ranked search	Keyword +/-	Search revo-cation	Keyword privacy	Index privacy	Token privacy	Search pattern privacy	Access pattern privacy
[1]	×	×	✓	×	×	✓	✓	✓	×	×
[2]	×	✓	✓	×	×	✓	✓	✓	✓	✓
[3]	✓	×	×	✓	×	✓	✓	✓	×	×
[4]	✓	✓	×	✓	×	✓	✓	✓	×	×
F ⁵ P ⁵	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

index privacy, *token privacy*, *search pattern privacy*, and *access pattern privacy*. To simultaneously achieve F⁵ and P⁵ (F⁵+P⁵), big surgeries to the underlying algorithms of the previous proposals are required. An overview of the widely referenced related work is illustrated in Table 1.

The rest of the paper is organized as follows. Section 2 is related work. Preliminaries are given in Section 3. Section 4 presents the construction of F⁵P⁵. Security and privacy analysis is presented in Section 5. Section 6 is performance evaluation. Section 7 concludes this paper.

2 Related Work

Multi-Keyword Search. Multi-keyword search enables users to search by specifying keywords of interests, and [2] achieved this by leveraging coordinate matching [5] which uses the number of queried keywords that appear in a file (i.e., the number of keywords that are hit) as the measurement of similarity between a file and a query. In [2], the similarity score is the inner product of the searchable index (*SI*) and searchable token (*ST*), both of which are binary vectors recording the existence of the predefined keyword in a file or in a query. Nevertheless, the similarity scores are coarse in [2], as binary vectors only indicate existence or not. Suppose files F and F' contain keyword w ; yet, the number of times that w appears in F is larger than that in F' . In [2], both F and F' will be returned to users with the same similarity score; users need to decrypt the files and conduct certain manual postprocess to obtain the more related one by themselves. Our F⁵P⁵ modifies coordinate matching by changing binary bits in *SI* into weight information of corresponding keywords, which is typically the number of times that the keyword appears in a file. The SP, by doing the modified coordinate matching, will get fine-grained similarity scores.

Ranked Search. The SP returns top- k files as search results according to certain scoring mechanism, without ever touching any sensitive information, which has been achieved by [1] and [2]. However, the order preserving symmetric encryption in [1] only supports single and exact keyword search where search pattern privacy and access pattern privacy cannot be protected. In [2], two $(n+1) \times (n+1)$ matrices (M_1, M_2) corresponding to n predefined keywords are used to encrypt *SI*; keyword addition/removal will result in corresponding changes to the size of (M_1, M_2) , so the data owner has to re-generate all *SI*.

Fuzzy Search. Fuzzy search aims to tolerate various typos and representation inconsistencies in different searching inputs, yet even two words that are close to each other would no longer be so after the cryptographic transformation. This has been coped with in [3], [4] via building a fuzzy keyword set (Fuz). However, they did not achieve search pattern privacy and access pattern privacy due to the underlying algorithms. Besides, storage cost of Fuz is another problem. Edit distance $ed(w_1, w_2)$ is introduced which is the number of operations¹ required to transform word w_1 to w_2 [3], [4]. The storage cost of Fuz is $\mathcal{O}(t^{ed})$ for a keyword with length t and edit distance ed in [3]. In [4], both data owners and users are required to construct bloom filters according to every possible distance $ed' \leq ed$. So, the storage cost is even larger than [3], considering both the data owner and the user sides. We construct Fuz by leveraging an information retrieval technique [8] and the storage cost of Fuz has been reduced to $\mathcal{O}(t^2)$ per keyword with length t without requiring users to specify appropriate edit distance.

Keyword Addition/Removal. Suppose W is the predefined keyword set and a new file, characterized by some new keywords $W' (W' \not\subseteq W)$, needs to be outsourced. W' must be added into W for further search. Also, sometimes some keywords need to be withdrawn from W . Previously, [3] and [4] have tackled this, yet they are inherently incapable of fulfilling $F^5 + P^5$. In [1] and [2], keyword addition/removal will result in re-encryption of all SI .

Search Revocation. It is preferable that the data owner may invoke a user's search ability immediately, which is known as instantaneous search revocation. Previously, [6] achieved this, but a look-up table is introduced, which is inflexible and has not been widely adopted. So far, none of the methods without look-up table has achieved search revocation due to the limit of the underlying algorithm.

3 Preliminaries

Fig. 1. Architecture of F^5P^5

Table 2. Fuzzy keyword set for “cate”

Permuterms	Prefixes			
/cate	/c	/ca	/cat	/cate
e/cat	e/	e/c	e/ca	e/cat
te/ca	te/	te/c	te/ca	
ate/c	ate/	ate/c		

System Model. We follow a typical architecture for data outsourcing as shown in Fig. 1. Three entities are involved: data owner O , data user U , and the SP. O has a collection of files F to be outsourced. O extracts predefined keywords W from F . O builds SI for each file in F and encrypts the file with a symmetric cipher $E(\cdot)$. SI is attached to $E(F)$ and $(SI, E(F))$ are outsourced to the SP.

¹ The operations include changing, deleting, and inserting characters into a word.

Each user U will be granted a searchable key (SK) by O . U specifies keywords of interest and generates ST with SK . Then, U sends ST to the SP. The SP evaluates the ST over each SI and returns the ranked top- k files without decrypting $E(F)$. We assume the authorization between O and U has been appropriately established by separate cryptographic techniques. The search ability of U can be revoked by O , which is called search revocation. Any user U with a valid SK has the same ability to search. Issues such as selection of $E(\cdot)$, symmetric key agreement, and access control, are not within the scope of this paper. We only focus on searchable encryption, which is consistent with previous work [1], [2].

Threat Model. Any user with a valid SK can search over the *entire* dataset, which makes collusion of users meaningless, as the collusion will not provide a greater search ability. The SP is “honest-but-curious” who could be the potential adversary, which is consistent with previous work [1], [2], [6]. The SP will honestly follow the protocol, but it is curious to guess or infer underlying information based on what it knows. To meet high safety standard as [2], in addition to SI , ST , encrypted files and the search results, we assume the SP has some additional background information about the encrypted files.

Fuzzy Keyword Set. We construct the fuzzy keyword set Fuz by leveraging the method in [8]. First, for keyword $w = c_1c_2\dots c_t$ with t characters, add “/” to the end: $w = c_1c_2\dots c_t/$. Then, by rotating t times w we get t permuterms: $(/c_1c_2\dots c_t, c_t/c_1c_2\dots c_{t-1}, \dots, c_2\dots c_{t-1}/c_1)$. To illustrate, the permuterms for “cate” is: $(/cate, e/cat, te/ca, ate/c)$. Each permuterm is expanded to get all its prefixes. The prefixes of “/cate” are $(/c, /ca, /cat, /cate)$. Note that a valid prefix must contain “/” and at least one character. Namely, “/” of “/cate” and “e” of “e/cat” are invalid. Table 2 summarizes Fuz for “cate”. Each keyword with length t will generate t permuterms and each permuterm has up to t prefixes. So the storage cost of Fuz is up to $\mathcal{O}(t^2)$, which yields a better performance than that in [3].

4 Construction of F⁵P⁵

Setup. First, data owner O extracts predefined keywords $W = (w_i, 1 \leq i \leq d)$ from the files $F = (F_j, 1 \leq j \leq l)$ and builds Fuz_i for w_i , $Fuz = (Fuz_i, 1 \leq i \leq d)$. Second, O finds a large prime p with $\|p\|$ bits² and picks $g \in_R [2, p-1]$, $g^{-1} \cdot g \equiv 1 \pmod p$. Third, O creates a truncated normal distribution $TN(\mu, \sigma^2)$ within the interval $[\mu - 3\sigma, \mu + 3\sigma]$, where $\mu, \sigma \in \mathbb{Z}^+$, $\mu - 3\sigma > 0$, and $\|\mu\| < \frac{\|p\|}{4}$. Then, O randomly picks $2d$ secret integers ($r_q \in \mathbb{Z}^+, 1 \leq q \leq 2d$) and computes:

$$R = (R_q = g^{r_q} \pmod p, 1 \leq q \leq 2d), R^{-1} = (R_q^{-1} = g^{-r_q} \pmod p, 1 \leq q \leq 2d).$$

Index Generation. For $F_j \in F$, O calculates the number of times that $w_i \in W$ appears in F_j : $N_{j,i}$, $N_j = (N_{j,i}, 1 \leq i \leq d)$. Then, a d -dimension vector θ_j is generated for F_j , where each entry $\theta_{j,i}$ is constructed as: $\theta_{j,i} = aN_{j,i}$, if $N_{j,i} \neq 0$; else $\theta_{j,i} = a'$. Here, $a = x_1^{a_1} x_2^{a_2} \dots x_e^{a_e}$, $\|a\| < \frac{\|p\|}{4}$ is a large integer for enlarging

² We use $\|\cdot\|$ to denote the number of bits in $\cdot$.

$N_{j,i}$, $a' = x_1^{a'_1} x_2^{a'_2} \dots x_e^{a'_e}$ is a small integer as dummy weight. Subsequently, based on θ_j , a $2d$ -dimension vector α will be generated. Each entry α_i is set as:

$$\alpha_i = \theta_{j,i} + \theta_{j,i+1}, \quad \alpha_{i+d} = \theta_{j,i} - \theta_{j,i+1}, \quad 1 \leq i \leq d.$$

O randomly picks d integers ($m_i, 1 \leq i \leq d$) according to $TN(\mu, \sigma^2)$ and rounds m_i to its closest integer. The searchable index SI_j for F_j is generated as:

$$SI_j = (m_1 \alpha_1 R_1, \dots, m_d \alpha_d R_d, m_1 \alpha_{d+1} R_{d+1}, \dots, m_d \alpha_{2d} R_{2d}) \bmod p.$$

Then, O randomly picks $v \in \mathbb{Z}^+$ and publishes $(p, 2v\mu^2)$. Finally, O encrypts F_j as $E(F_j)$ and outsources $((SI_j, E(F_j)), 1 \leq j \leq l)$ to the SP. The secret parameters: $(g, W, R, a, a', v, N_j, 1 \leq j \leq l)$ are secretly held by O .

Searchable Key Generation. O sends each authorized user the same searchable key: $SK = (Fuz, \mu, \sigma^2, R^{-1})$. SK is only sent once.

Token Generation. User U searches by specifying a set of keywords and generating a searchable token ST_u with SK . Note that the keywords of interests may not exactly match the predefined keywords in W . To support fuzzy search, certain pretreatment should be conducted. Suppose “cate” is a fuzzy keyword. U will go through Fuz to get the most possible keywords in W for “cate”.

1. Deletion. Delete some characters of “cate”. We illustrate by deleting one character: (ate, cte, cae, cat) . By adding “/” before each word, we get $H = (/ate, /cte, /cae, /cat)$. For each $h \in H$, if there exists $w \in Fuz_i$ satisfying $h = w$, i is added into Pos .
2. Insertion. Insert some characters into “cate”: $(*cate, c * ate, ca * te, cat * e, cate*)$. Three transformations should be conducted.
 - Prefix transformation. Wildcards like $c_1 c_2 \dots c_t *$ are translated into $/c_1 c_2 \dots c_t *$.
 - Suffix transformation. Wildcards like $*c_1 c_2 \dots c_t$ are translated into $c_1 c_2 \dots c_t / *$.
 - Pre/Suffix transformation. $c_1 \dots c_i * c_j \dots c_t$ are translated into $c_j \dots c_t / c_1 \dots c_i *$.
 Finally, we get $H = (cate/*, ate/c*, te/ca*, e/cat*, /cate*)$. For each $h \in H$, if there exists $w \in Fuz_i$ such that w is a prefix of h , i is added into Pos .
3. Substitution. Change one character to another: $(cat*, ca*e, c*te, *ate)$. After the same transformations, we get: $H = (/cat*, e/ca*, te/c*, ate/*)$. For each $h \in H$, if there exists $w \in Fuz_i$ such that w is a prefix of h , add i to Pos .

After the same pretreatment for all the fuzzy keywords, we get $(w_i, i \in Pos)$. With Pos , a d -dimension vector η will be generated, where each entry η_i is constructed as: $\eta_i = b$, if $i \in Pos$; else, $\eta_i = b'$. $b = x_1^{b_1} x_2^{b_2} \dots x_t^{b_t}, ||b|| < \frac{||p||}{4}$ is a random large integer chosen by U for enlarging the actual query entries. $b' = x_1^{b'_1} x_2^{b'_2} \dots x_t^{b'_t}$ is a small random integer called dummy query. A $2d$ -dimension vector β is generated where each entry is constructed as:

$$\beta_i = \eta_i + \eta_{i+1}, \quad \beta_{i+d} = \eta_i - \eta_{i+1}, \quad 1 \leq i \leq d.$$

U randomly picks d integers ($n_i, 1 \leq i \leq d$) according to $TN(\mu, \sigma^2)$ and rounds n_i to its closest integer. The searchable token ST_u is constructed as:

$$ST_u = (n_1 \beta_1 R_1^{-1}, \dots, n_d \beta_d R_d^{-1}, n_1 \beta_{d+1} R_{d+1}^{-1}, \dots, n_d \beta_{2d} R_{2d}^{-1}) \bmod p.$$

Query. U sends ST_u to the SP. The SP evaluates ST_u over each SI_j as:

$$\begin{aligned} SI_j \cdot ST_u \bmod p &= [\sum_{i=1}^d m_i n_i (\theta_{j,i} + \theta_{j,i+1}) (\eta_i + \eta_{i+1}) \\ &\quad + \sum_{i=1}^d m_i n_i (\theta_{j,i} - \theta_{j,i+1}) (\eta_i - \eta_{i+1})] \bmod p \\ &= 2 \sum_{i=1}^d (m_{i-1} n_{i-1} + m_i n_i) \theta_{j,i} \eta_i \bmod p, \end{aligned}$$

where we set $m_0 = m_d, n_0 = n_d$. Since p is a large prime, the results can be simplified as $SS_j = 2 \sum_{i=1}^d (m_{i-1} n_{i-1} + m_i n_i) \theta_{j,i} \eta_i$. The SP gets the final similarity scores by computing:

$$SS_j / 2v\mu^2 = (2/v) \sum_{i=1}^d \theta_{j,i} \eta_i (m_{i-1} n_{i-1} + m_i n_i) / 2\mu^2.$$

Files will be ordered according to the similarity scores in a descending way and top- k files will be returned. Here, $(m_{i-1} n_{i-1} + m_i n_i) / 2\mu^2$ are interference elements whose expectation is: $E(m_{i-1} n_{i-1} + m_i n_i) / 2\mu^2 = (E(m_{i-1})E(n_{i-1}) + E(m_i)E(n_i)) / 2\mu^2 = 1$. The slight fluctuation of similarity scores due to the interference elements is a tradeoff between access pattern privacy and search precision. Experiments later shows that this is a practice-friendly tradeoff.

Keyword Addition/Removal. Suppose F_{l+1} is the new file needs to be outsourced, which introduces a new keyword w_{d+1} . First, O builds Fuz_{d+1} for w_{d+1} . Second, O randomly picks r_{2d+1}, r_{2d+2} and generates R_{2d+1}, R_{2d+2} as in **Setup**. Third, O re-picks m'_d and randomly chooses m_{d+1} as in **Index Generation**. Last, for each F_j , O sets $\theta_{j,d+1} = a'$ and re-encrypts $SI_{j,d}, SI_{j,2d}$ as:

$$SI_{j,d} = (\theta_{j,d} + \theta_{j,d+1}) m'_d R_d \bmod p, SI_{j,2d} = (\theta_{j,d} - \theta_{j,d+1}) m'_d R_{2d} \bmod p.$$

Two new entries $SI_{j,d+1}, SI_{j,2d+2}$ will be inserted into SI_j :

$$SI_{j,d+1} = (\theta_{j,d+1} + \theta_{j,1}) m_{d+1} R_{2d+1} \bmod p,$$

$$SI_{j,2d+2} = (\theta_{j,d+1} - \theta_{j,1}) m_{d+1} R_{2d+2} \bmod p.$$

Next, O generates SI_{l+1} for F_{l+1} as it did in **Index Generation**. Finally,

$$((SI_{j,d}, SI_{j,2d}, SI_{j,d+1}, SI_{j,2d+2}, j \in [1, l]), (SI_{l+1}, E(F_{l+1})))$$

will be sent to the SP and $(Fuz_{d+1}, R_{2d+1}^{-1}, R_{2d+2}^{-1}, d+1)$ will be sent to users. Sometimes, certain predefined keywords w_z needs to be removed due to deletion of some files. In such cases, O re-picks m'_{z-1} and updates $SI_{j,z-1}, SI_{j,z+d-1}$:

$$SI_{j,z-1} = (\theta_{j,z-1} + \theta_{j,z+1}) m'_{z-1} R_{z-1} \bmod p,$$

$$SI_{j,z+d-1} = (\theta_{j,z-1} - \theta_{j,z+1}) m'_{z-1} R_{z+d-1} \bmod p.$$

Update message $(z, SI_{j,z-1}, SI_{j,z+d-1}, 1 \leq j \leq l)$ will be send to the SP. The SP deletes $SI_{j,z}, SI_{j,z+d}$ and updates $SI_{j,z-1}, SI_{j,z+d-1}$. U updates SK as $SK = (Fuz \setminus Fuz_z, \mu, \sigma^2, R^{-1} \setminus R_z^{-1})$.

Search Revocation. F⁵P⁵ achieves search revocation by updating pivotal information. First, O re-picks $(r'_q \in \mathbb{Z}^+, 1 \leq q \leq 2d)$ as it did in **Setup** and generates two update keys respectively for the SP and all unrevoked users:

$$UK_{sp} = (g^{r'_q - r_q} \bmod p, 1 \leq q \leq 2d), UK_u = (g^{r_q - r'_q} \bmod p, 1 \leq q \leq 2d).$$

The SP updates SI_j with UK_{sp} : $SI_{j,q} = SI_{j,q} \cdot g^{r'_q - r_q} \bmod p$. Unrevoked user U updates SK with UK_u : $R_q^{-1} = R_q^{-1} \cdot g^{r_q - r'_q} \bmod p$.

5 Security and Privacy

5.1 Security

We prove the security of F^5P^5 in two scenarios. First, the SP guesses with SI , ST , and background knowledge. Additionally, the SP guesses with $ST \triangle SI$, which is a binary relation: $ST \triangle SI = (m_q n_q \alpha_q \beta_q, 1 \leq q \leq 2d)$, $m_i = m_{i+d}$, $1 \leq i \leq d$.

Theorem 1. Alone with the SI , ST , and the background knowledge, F^5P^5 is secure as long as the entries of SI or ST are indistinguishable to the SP.

Proof. Since $m_q, \alpha_q, n_q, \beta_q \in [2, p-1]$, $1 \leq q \leq 2d$, SI and ST can be uniformly denoted as $S = (g^{S_1}, \dots, g^{S_{2d}}) \bmod p$. The SP guesses by picking $g' \in_R [2, p-1]$ and S'_q such that $(g')^{S'_q} = g^{S_q}$. Since the choice of g and S_q are out of the SP's sight, it cannot verify whether $g' = g$. Namely, the entries in SI and ST are indistinguishable to SP. Thus, F^5P^5 is secure according to Theorem 1.

Theorem 2. In addition to what the SP knows in Theorem 1, F^5P^5 is secure if the SP decomposes $ST \triangle SI$ with a negligible advantage.

Proof. According to the construction of SI and ST , a' is a divisor of α_q : $a' | \alpha_q$. Also, $b' | \beta_q$. If the SP decomposes $m_q n_q \alpha_q \beta_q$, it decomposes $m_q n_q a' b'$. We set:

$$m_q = x_1^{m_{q1}} x_2^{m_{q2}} \dots x_y^{m_{qy}}, n_q = x_1^{n_{q1}} x_2^{n_{q2}} \dots x_y^{n_{qy}}, a' = x_1^{a'_1} x_2^{a'_2} \dots x_y^{a'_y}, b' = x_1^{b'_1} x_2^{b'_2} \dots x_y^{b'_y}.$$

The probability that the SP decomposes $m_q n_q a' b'$ is:

$$\varepsilon = 1 / [(m_{q1} + n_{q1} + a'_1 + b'_1 + 1) \cdot (m_{q2} + n_{q2} + a'_2 + b'_2 + 1) \dots (m_{qy} + n_{qy} + a'_y + b'_y + 1)].$$

The number of divisors (y) can be large enough as long as p is large enough, so ε is negligible and F^5P^5 is secure according to Theorem 2.

5.2 Privacy

Keyword Privacy. Which keywords are being queried must not be divulged. In F^5P^5 , the SP cannot speculate which keywords are being queried from ST without SK . Keyword privacy has been achieved.

Index Privacy. The SP should not deduce any sensitive information from SI and its associated encrypted file. In F^5P^5 , SI are indistinguishably encrypted without deterministic relationship due to the random m_i and r_i , so the SP cannot deduce any underlying information even if it knows background information.

Token Privacy. The SP should not have the ability to generate a valid ST based on previously ones received from users. In F^5P^5 , the entries within a single ST are indistinguishably generated thanks to n_i, r_i . Among different ST , there is no

Fig. 2. *Fuz*: Storage cost Fig. 3. *Fuz*: Time cost Fig. 4. Time cost of generating *SI* Fig. 5. Time cost of generating *ST*

deterministic relationship thanks to the random choice of b, b' . Namely, the SP cannot make up a valid *ST* from previous ones to search by itself.

Search Pattern Privacy. Search pattern is the information that can be derived when the SP knows two arbitrary searches are performed for the same keywords. In F⁵P⁵, the random b, b' makes *ST* indistinguishable even if two *ST* involve the same keywords, so search pattern privacy is preserved.

Access Pattern Privacy. Access pattern refers to search result sequence. Private information retrieval (PIR) [7] could solve this but it is inefficient. In F⁵P⁵, result sequence fluctuates slightly due to the interference elements, even if the same keywords have been queried. Namely, an “as-strong-as-possible” access pattern privacy is provided, which is consistent with previous work [2].

6 Performance Analysis

We implement F⁵P⁵ on a Windows 7 Server (Intel Core(TM) Processor 3.00GHz) with a real-world dataset the same with [2]: the Enron Email Dataset [9].

Fuzzy Keyword Set Generation. Time and storage costs of building *Fuz* depend directly on the cardinality of W : $|W|$. For the first time, we reduce

Fig. 6. Time cost of query with different number of keywords in the query

Fig. 7. Time cost of query with different number of files in the dataset

Fig. 8. Precision

Fig. 9. Privacy

the storage cost to $\mathcal{O}(t^2)$ per keyword with length t . Fig. 2 is the storage cost according to different $|W|$, where 25KB is required when $|W| = 140$. In Fig. 3, only 0.68 milliseconds are required when $|W| = 140$. Note that building *Fuz* is a one-time operation before data outsourcing.

Index Generation. Generation of *SI* involves calculating weight information and encryption. Fig. 4 is the evaluation of generating *SI* where $|W| = 140$. 25.1 seconds are required to build searchable indexes for 1200 files, which is an improvement compared with previous work.

Token Generation. Generation of *ST* involves pretreatment of keywords and encryption. Fig. 5 evaluates the generation of *ST*. In Fig. 5, the time cost is almost linear with the number of queried keywords, where 21 milliseconds are required for 30 keywords (15 fuzzy keywords). Pretreatment accounts most of the time and encryption only consumes a small part.

Query. Fig. 6 is evaluation of time cost according to different number of keywords ($|F| = 1200$, $|W| = 140$). The time cost keeps almost constant (approximate 29 milliseconds) no matter how many keywords in the query, which it is a desirable feature and has not been achieved in most related researches except

[2]. Fig. 7 is evaluation according to different dataset sizes. 21 milliseconds are required to query among 600 files, which is almost negligible.

Precision and Privacy. The adoption of interference elements involves a trade-off between search precision and access pattern privacy. We follow the same definition as [2]: precision, $Pre_k = k'/k$, where k' is the number of files that exist in the real top- k files; privacy, $Pri_k = (\sum pri_j)/k$, where $pri_j = |Rank_j - Rank'_j|$ is the rank perturbation of F_j . $Rank_j$ is the rank of F_j in the retrieved top- k files and $Rank'_j$ is its rank in real top- k files. M_{SI} (M_{ST}) is the magnitude difference between a and a' (b and b'). We set $M_{SI} = M_{ST}$ for convenience. Fig. 8 is evaluation of precision and Fig. 9 is evaluation of privacy ($|F| = 1200$, $|W| = 140$), where large choice of $M_{SI}(M_{ST})$ will result in high precision and low privacy. So, $M_{SI}(M_{ST})$ is a balance parameter between precision and privacy. Precision in F⁵P⁵ is around 95% with approximate 5 privacy ($M_{SI} = M_{ST} = 3$), compared with 90% precision and 0.08 privacy in [2]. Namely, F⁵P⁵ provides higher precision and better privacy assurance. Also, the slight impact of σ indicates that the interference elements have done a good job of making SI and ST undistinguishable without reducing precision.

7 Conclusion

We have proposed a full-fledged searchable encryption scheme called F⁵P⁵ which simultaneously achieves the five keyword search functions (F⁵) and the five privacy preserving requirements (P⁵). Thorough analysis shows that F⁵P⁵ is secure and privacy preserving. Experiments based on a real-world dataset show that F⁵P⁵ performs well in terms of both computation and storage with high precision of search results. As for our future work, we will explore how to assign different search abilities to different users.

References

1. Wang, C., Cao, N., Li, J., Ren, K., Lou, W.: Secure Ranked Keyword Search over Encrypted Cloud Data. In: IEEE 30th International Conference on Distributed Computing Systems, pp. 253–262 (June 2010)
2. Cao, N., Wang, C., Li, M., Ren, K., Lou, W.: Privacy-preserving Multi-keyword Ranked Search over Encrypted Cloud Data. In: 30th IEEE Conference on Computer Communications, pp. 829–837 (March 2011)
3. Li, J., Wang, Q., Wang, C., Cao, N., Ren, K., Lou, W.: Fuzzy Keyword Search over Encrypted Data in Cloud Computing. In: 29th IEEE Conference on Computer Communications, pp. 1–16 (March 2010)
4. Chuah, M., Hu, W.: Privacy-aware Bedtree Based Solution for Fuzzy Multi-keyword Search over Encrypted Data. In: 31st International Conference on Distributed Computing Systems Workshops, pp. 273–281 (June 2011)
5. Witten, I.H., Moffat, A., Bell, T.C.: Managing gigabytes: Compressing and Indexing Documents and Images. Morgan Kaufmann Publishing (May 1999)

6. Curtmola, R., Garay, J.A., Kamara, S., Ostrovsky, R.: Searchable Symmetric Encryption: Improved Definitions and Efficient constructions. In: 13th ACM Conference on Computer and Communications Security, pp. 79–88 (November 2006)
7. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Cryptography from Anonymity. In: 47th IEEE Symposium on Foundations of Computer Science, pp. 239–248 (October 2006)
8. Salton, G.: Automatic Text Processing, the Transformation, Analysis and Retrieval of Information by Computer. Addison-Wesley, MA (1989)
9. Cohen, W.W.: Enron email dataset, <http://www.cs.cmu.edu/~enron/>