Journal of Network and Computer Applications 64 (2016) 12–22

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

An efficient and secure privacy-preserving approach for outsourced data of resource constrained mobile devices in cloud computing

Syam Kumar Pasupuleti [a,*], Subramanian Ramalingam [b], Rajkumar Buyya [c]

[a] Institute for Development and Research in Banking Technology (IDRBT), Hyderabad, India
[b] Department of Computer Science, Pondicherry University, Puducherry, India
[c] Cloud Computing and Distributed Systems (CLOUDS) Lab, Department of Computing and Information Systems, The University of Melbourne, Australia

ARTICLE INFO

Article history:
Received 1 January 2015
Received in revised form
16 October 2015
Accepted 8 November 2015
Available online 9 February 2016

Keywords:
Cloud computing
Privacy-preserving
Outsourced data
Probabilistic public-key encryption
Ranked keyword search
Mobile devices

ABSTRACT

Outsourcing of data into cloud has become an effective trend in modern day computing due to its ability to provide low-cost, pay-as-you-go IT services. Although cloud based services offer many advantages, privacy of the outsourced data is a big concern. To mitigate this concern, it is desirable to outsource sensitive data in an encrypted form but cost of encryption process would increase the heavy computational overhead on thin clients such as resource-constrained mobile devices. Recently, several keyword searchable encryption schemes have been described in the literature. However, these schemes are not effective for resource-constrained mobile devices, because the adopted encryption system should not only support keyword search over the encrypted data but also offer high performance. In this paper, we propose an efficient and secure privacy-preserving approach for outsourced data of resource-constrained mobile devices in the cloud computing. Our approach employs probabilistic public key encryption algorithm for encrypting the data and invoke ranked keyword search over the encrypted data to retrieve the files from the cloud. We aim to achieve an efficient system for data encryption without sacrificing the privacy of data. Further, our ranked keyword search greatly improves the system usability by enabling ranking based on relevance score for search result, sends top most relevant files instead of sending all files back, and ensures the file retrieval accuracy. As a result, data privacy ensures and computation, communication overheads in reduction. Thorough security and performance analysis, we prove that our approach is semantically secure and efficient.

1. Introduction

Cloud computing is emerging computing model where the data owners are outsourcing their data into the cloud storage. By outsourcing the data files into the cloud, it gives many benefits to the large enterprises as well as individual users because they can dynamically increase their storage space as and when required without buying any storage devices (Armbrust et al., 2009). They are: (1) the users can access the remotely stored data at anytime, from anywhere and gives permission to authorized users to share the data. (2) The users can be relieved from the burden of storage management at locally, (3) Avoidance of capital expenditure on hardware and software costs etc. To date, there are a number of cloud storage services: Amazon simple storage Space (S3), Rack space, Google, Microsoft, etc. (Jaeger et al. 2009).

Besides, all of these advantages of outsourced data in Cloud, there are also some significant issues. One of the major issues is the privacy of outsourced data in cloud (Jaeger and Schiffman, 2010) i.e., the sensitive information such as e-mail, health records, and government data may leak to unauthorized users (Slocum, 2009; Krebs, 2009) or even be hacked (Cloud Security Alliance, 2009). Since, the cloud is an open platform; it can be subjected to attacks from both malicious insiders and outsiders (Haclgiimfi et al., 2002). The Cloud service providers (CSPs) usually provide data security through mechanisms like firewalls and virtualization. However, these mechanisms do not protect users' privacy from the CSP itself due to remote cloud storage servers are untrusted.

A natural approach to preserve the privacy of sensitive data is to encrypt data before outsourcing it into the cloud and retrieves the data back through keyword based search over encrypted data. Although encryption provides protection from illegal accesses, it significantly increases the computation overhead on the data owners especially when they having resource-constrained mobile devices and large size of data files.

Further, the authorized users want to retrieve the certain files from cloud, need to communicate with the CSPs and allow him to

* Corresponding author.
E-mail addresses: psyamkumar@idrbt.ac.in (S.K. Pasupuleti), rsmanian.csc@pondiuni.edu.in (S. Ramalingam), rbuyya@unimelb.edu.au (R. Buyya).

http://dx.doi.org/10.1016/j.jnca.2015.11.023
1084-8045/

operate over the encrypted data. To meet effective data retrieval, it is preferred to get the most relevant files instead of getting all files i.e., The files should be ranked and only highest relevant files are send back to the users, which is highly desirable in the "pay-as-you-use" cloud model. However, it is challenging task that retrieves the data back in a secure and efficient manner without being able to extract useful information from the cloud.

Therefore, the efficient and secure mechanisms are needed to protect the privacy of sensitive data in a cloud environment. Moreover, the importance and necessity of privacy preserving of data search techniques are even more pronouncing in the cloud applications. For example, large companies that are operating on the public clouds like Google or Amazon may access the sensitive data, search patterns, hiding the query and retrieved data has great importance in ensuring the privacy of that using cloud services.

Recently, several keyword search based encryption schemes have been proposed to ensure the privacy of outsourced data (Song et al., 2000; Goh, 2003; Chang and Mitzenmacher, 2005; Curtmola et al., 2006; Li et al., 2010; Kuzu et al., 2012; Lu, 2012; Orencik and Savas, 2014; Wang et al., 2012; Cao et al., 2014; Boneh et al., 2004; Bellare et al., 2007; Attrapadung and Li Bert, 2010; Katz et al., 2008; Ogata and Kurosawa, 2004; Shi et al., 2007; Waters et al., 2004; Bao et al., 2008; Boldyreva et al., 2009; Liu et al., 2012; Yu et al., 2013). In all these schemes, the data owner first encrypts the data before outsourcing it and later retrieves them through keyword search or ranked keyword search. The schemes (Song et al., 2000; Goh, 2003; Chang and Mitzenmacher, 2005; Curtmola et al., 2006; Li et al., 2010; Kuzu et al., 2012; Lu, 2012; Orencik and Savas, 2014; Wang et al., 2012; Cao et al., 2014) and (Boneh et al., 2004; Bellare et al., 2007; Attrapadung and Li Bert, 2010; Katz et al., 2008; Ogata and Kurosawa, 2004; Shi et al., 2007; Waters et al., 2004; Bao et al., 2008; Boldyreva et al., 2009) proposed under symmetric-key cryptography and public-key cryptography respectively. However, such encryption schemes use too much CPU time and memory power of client during the encryption and decryption process. That is the thin client has only limited bandwidth, CPU power, and memory, therefore, the traditional encryption schemes cannot work well in cloud environment.

To avoid above problems, Liu et al. (2012) proposed a secure and privacy-preserving keyword search over the encrypted data for cloud storage applications using Elliptic Curve Cryptography (ECC) over Fp. However, this scheme supports only Boolean keyword search i.e., either a keyword exists in a file or not, without considering the difference of relevance with the queried keyword of these files in the result. To improve the efficiency without sacrificing privacy, Yu et al. (2013) proposed a Two Round Searchable Encryption (TRSE) scheme that supports ranked multi-keyword search over encrypted data for file retrieval. It employs the vector space model and homomorphic encryption as a result, the information leakage can be eliminated and data security is ensured. However, the computation and communication costs of this scheme are quite large, since every search term in a query requires several homomorphic encryption operations on the data owner side. Further, it uses two-round communication process to retrieve the files back which resulting the unnecessary communication overhead.

In this paper, we propose an efficient and secure privacy-preserving approach to avoid all above problems while preserving the privacy and integrity of outsourced data in the cloud. In our scheme, the data owner first builds the index for file collection, encrypts both index and data files, and stores them in the cloud. Later, to retrieve the stored files from the cloud server, the authorized user generate trapdoor for keywords and sends to the server. Upon receiving the trapdoor, the cloud server search for a list of matched file entries and their corresponding encrypted relevance scores. Then matched files should be sent back to the user in a ranked sequence based on the relevance scores. By decrypting it, the user gets the original files back. Further, our approach verifies the integrity of data in cloud. This approach utilizes the probabilistic public key encryption technique (Witten et al., 1999) and ranked keyword search (Cao et al., 2014; Yu et al., 2013). It greatly reduces the processing overhead of data owners while encrypting the files, index and it is most suitable for resource-constrained mobile devices (thin clients) in Cloud computing and ranked keyword search process reduce the communication overhead during the file retrieval. Through the security and performance analysis, we prove that our scheme is semantically secure and efficient.

The **key contributions** of our work can be summarized as follows:

1. We propose an efficient and secure privacy-preserving approach; it uses probabilistic public key encryption technique to reduce computational overhead on owners while encryption and decryption process without leaking any information about the plaintext.
2. Our approach uses ranked keyword search on encrypted data to retrieve the files back. It enables the cloud server to determine whether a given file contains certain keywords and associated relevance score without knowing of any information about both the keywords and the files. It greatly reduces the communication overhead during the file retrieval process. It also verifies the integrity of data stored in cloud
3. Through analysis on security demonstrates that propose scheme can be proved semantically secure under different attacks. Furthermore, the performance analysis and experiential results show that our scheme is efficient and it outperforms compared with existing schemes.

The rest of the paper is organized as follows: In Section 2, we briefly discuss about existing schemes. In Section 3, we explain system architecture used in our scheme. In Section 4, we describe our proposed approach. In Sections 5 and 6, we analyze security, and performance analysis respectively. In Section 7, we identify limitations and areas of improvements. Finally, we conclude our paper with future directions in Section 8.

2. Related work

The searchable encryption schemes have been widely investigated as a cryptographic primitive with a focus on security definition formalizations and efficiency improvements (Song et al., 2000; Goh, 2003; Chang and Mitzenmacher, 2005; Curtmola et al., 2006; Li et al., 2010; Kuzu et al., 2012; Lu, 2012; Orencik and Savas, 2014; Wang et al., 2012; Cao et al., 2014; Boneh et al., 2004; Bellare et al., 2007; Attrapadung and Li Bert, 2010; Katz et al., 2008; Ogata and Kurosawa, 2004; Shi et al., 2007; Waters et al., 2004; Bao et al., 2008; Boldyreva et al., 2009; Liu et al., 2012; Yu et al., 2013). These searchable encryption schemes can be divided into two types: symmetric key encryption (Song et al., 2000; Goh, 2003; Chang and Mitzenmacher, 2005; Curtmola et al., 2006; Li et al., 2010; Kuzu et al., 2012; Lu, 2012; Orencik and Savas, 2014; Wang et al., 2012; Cao et al., 2014) and public key encryption (Boneh et al., 2004; Bellare et al., 2007; Attrapadung and Li Bert, 2010; Katz et al., 2008; Ogata and Kurosawa, 2004; Shi et al., 2007; Waters et al., 2004; Bao et al., 2008; Boldyreva et al., 2009).

2.1. Searchable encryption schemes based on symmetric key encryption

The symmetric key encryption scheme allows a data owner to outsource its data symmetrically encrypted to an untrusted server and later to search for a specific file in server via trapdoor.

Song et al. Song et al. (2000) introduced the notion of symmetric searchable encryption scheme, where each word in the file encrypted under a special two-layered encryption. Thus, the search time is linear to file collection length. Later, Goh et al. Goh (2003) developed a Bloom filter based per-file index scheme to reduce a workload for each search request proportional to the number of files in the collection. Similarly, Chang et al. Chang and Mitzenmacher (2005) also described a per-file index scheme, which is slightly stronger, than (Goh, 2003). However, both definitions do not consider adaptive adversaries, which could generate the queries according to the outcomes of previous queries. Further, to enhance search efficiency, Curtmola et al. Curtmola et al. (2006) proposed a per-keyword-based approach, where a single encrypted hash table index built for the entire file collection, with each entry consisting of the trapdoor of a keyword and an encrypted set of related file identifiers.

Although traditional searchable symmetric encryption schemes allow a user to securely search over encrypted data through keywords and retrieve the files of their interest, these techniques support only exact keyword search. That is there is no tolerance of minor types and format inconsistencies. This significant drawback makes existing techniques unsuitable in Cloud computing as it greatly affects system usability, rendering user searching experiences are very frustrating and system efficiency is very low.

In ordered to solve these problems, Li et al. (2010) proposed wildcard based fuzzy keyword search scheme over the encrypted data. Although fuzzy scheme tolerates errors to some extent, it is only applicable to strings under edit distance. In addition, fuzzy sets may become too big if we have long words, which necessitate issuing large trapdoors. In order to avoid this, Kuzu et al. (2012) described an efficient similarity search over the encrypted data. Similarly, Lu (2012) described a privacy-preserving logarithm search over the encrypted data.

All these secure index schemes support only Boolean keyword search and none of them support the ranked search problem. If the user searches for a single or more keywords, there will be a possibility that many correct matches would come where some of them may not be useful for the user at all. Therefore, it is difficult to decide which documents are the most relevant.

To solve the above problems, Orencik and Savas (2014) proposed a practical privacy-preserving ranked keyword search scheme based on PIR that allows the multi-keyword queries with ranking capability. It increases the security of the keyword search while still satisfying efficient computation and communication requirements. However, this scheme is inefficient due to their blinding technique, which is not suitable for resource-constrained devices. A recent work defined by Wang et al. (2012) secure ranked keyword search over encrypted cloud data. The ranked keyword search greatly enhances system usability by enabling search result based on relevance ranking instead of sending undifferentiated results, and further ensures the file retrieval accuracy. Specifically, they explored the statistical measure approach, i.e., relevance score from information retrieval to build a secure searchable index, and develop a one-to-many order-preserving mapping technique to properly protect those sensitive score information. The resulting design is able to facilitate efficient server-side ranking without losing keyword privacy. Similarly, Cao et al., (2014) introduced a new method that allows multi-keyword ranked search over encrypted database. Moreover, it is not efficient due to matrix multiplication operations of square matrices where the numbers of rows are in the order of several thousands. However, cloud server has linearly traversed the whole index of all the documents for each search request.

However, all these schemes are working based on symmetric key encryption, where single key used to encrypt and decrypt the data. If the data owner needs to share the secrete-key, which is used in trapdoor generation to all authorized users. Sharing a secret key by several users forms a high security risk since it can easily leak to the unauthorized parties. Once the unauthorized parties learn the secret key, they can break the system and access the data.

2.2. Searchable encryption based on public-key encryption

To avoid key leakage problems, the public-key encryption is used in a similar scenario with two keys: one key is for encryption and another for decryption.

Boneh et al., (2004) developed a public-key searchable encryption scheme that can be extended to handle range, subset and conjunctive queries. It is also hiding the attributes for message that match a query. Similarly, Bellare et al., (2007) proposed a deterministic efficiently searchable public-key encryption scheme. Since searchable tags are deterministic, the server can organize them in a sorted way and match the minimum logarithmic time. Although it is efficient, this scheme only supports equality search and it is hard to deal with duplicate attribute values. The records with duplicate attribute values will end up with same cipher text, exposing plaintext frequency.

Attrapadung and Li Bert (2010) designed a privacy preserving keyword search protocol based on RSA blind signatures. It requires public-key operations per item in the database for every query and this operation performed on the user side. Later, Katz et al., (2008) proposed a public-key based encryption that supports inner products. Ogata and Kurosawa (2004) improved the efficiency of the inner product encryption by sacrificing attribute privacy.

Allowing range queries over encrypted data in the public key settings has studied in other related works (Shi et al., 2007), and as an attempt to enrich query predicates, conjunctive keyword search over encrypted data also have been proposed in Waters et al. (2004), Bao et al. (2008) and Boldyreva et al. (2009). Though these schemes provide provably strong security but they are generally not efficient for a single search request, Moreover, these schemes do not support the ordered result listing on the server side. Thus, they cannot be effectively utilized, since the user does not know which retrieved files would be the most relevant.

However, not all these schemes support for resource-constrained devices. This is due to their encryption and decryption process creates process overhead on system.

To avoid above problems, Liu et al. (2012), proposed a secure and privacy preserving keyword searching scheme for cloud storage services using ElGamal public-key encryption based on Elliptic Curve Cryptography (ECC) over Fp. It allows the CSP to participate in the decipherment, and return the encrypted files containing certain keywords without knowing any information. However, this scheme may disclose information to cloud service provider because it allows the CSP to participate in the encryption process. Furthermore, like previous schemes (Song et al., 2000; Goh, 2003; Chang and Mitzenmacher, 2005; Curtmola et al., 2006; Li et al., 2010; Kuzu et al., 2012; Lu, 2012), this scheme does not support the ranked search technique. To support ranked keyword search with less efficiency, Yu et al., (2013) proposed a Two-Round Searchable Encryption (TRSE) scheme that supports top-k multi-keyword retrieval based on ranking. the TRSE scheme used a vector space model and homomorphic encryption techniques, it enables users to involve in the ranking process while the majority of computing works done at the server side by operations only on cipher text. As a result, information leakage can be eliminated and security is ensured. However, the computation and communication costs of this method are quite large since every search term in a query requires several homomorphic encryption operations both on the server and on

the user side. Further, it uses two-round communication process to retrieve the data from the server.

3. Problem statement

3.1. System architecture

In our work, we consider a model of cloud data storage system, which consisting of three main entities as illustrated in Fig. 1: Data Owner, Cloud Service Provider (CSP) and Authorized Users.

***Data Owner* (DO):** is an entity that has large amount of data to be stored in the cloud, can be individual user having mobile constrained devices such as smart phones, PDA, TPM chip, etc.
***Cloud Service Provider* (CSP):** is an entity, provides data storage services and computational resources dynamically to the data owner and users
***Authorized Users* (AU):** The data owner allows the authorized users to use their files and share some keying material with the data owner. The authorized users would retrieve the data from the cloud in an encrypted form and by decrypting it they get the original data.

The typical interactions between these three entities of the system (see Fig. 1) are follows:

1) The data owner wants to outsource the set of files on the cloud server in encrypted form while still keeping the capability to search them through keyword for effective data utilization reasons.
2) When an authorized user wants to retrieve the file collection, send a search request to the CSP
3) Then, the CSP search the files and returns set of files and hash values of files to the user.
4) Finally, the authorized user verifies the integrity and decrypts the files and gets the corresponding plaintext

3.2. System model

In above system model, the data owner first outsources the encrypted data files into cloud servers via Cloud Service Provider (CSP). Once data moves to the cloud, he has no control over it. This lack of control on data raises privacy issue in the cloud, even if CSP provides some standard security mechanism to protect the data from attackers, still it is hacking. Therefore, we need an efficient and secure mechanism to protect the privacy of sensitive outsourced data in the cloud.

In our scheme, we consider the efficient and secure ranked keyword search over encrypted data as follows: the search result should return the files according to certain ranked relevance criteria to improve file retrieval accuracy for users without prior knowledge on the file collection. However, the cloud server should learn nothing about the index and data as they exhibit significant sensitive information against keyword privacy. To reduce bandwidth, the CSP sends only top-k most relevant files to the users for inserted keywords.

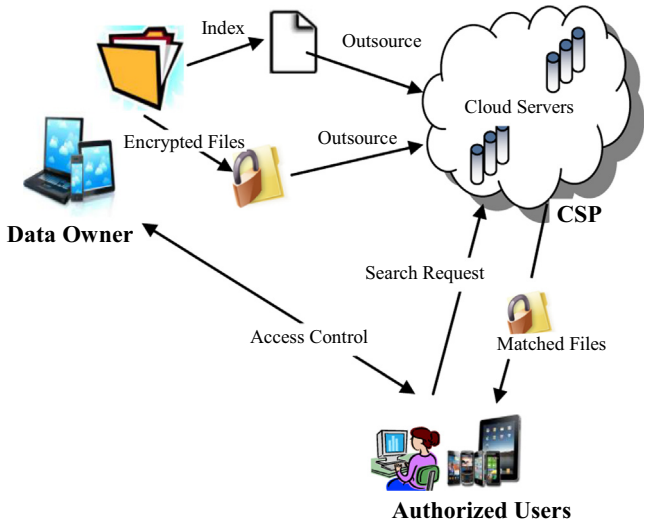

Fig. 1. Cloud data storage architecture.

3.3. Threat model

In threat model, we are considering mainly two types of threats, which are disturbing the outsourced data in the cloud: Internal Attacks and External Attacks

1. Internal Attacks: which are initiated by malicious insiders: Cloud users, malicious third party user (either cloud provider or customer organizations) are self-interested to accesses the data or disclose the data stored in the cloud. They also alter or modify the data.
2. External Attacks: which are initiated by unauthorized outsiders, we assume that external attackers can compromise all storage servers, so that they can intentionally access the owner's data

3.4. System goals

In order to address the privacy of sensitive data stored in the cloud, we propose an efficient and secure privacy-preserving approach with following goals:

1. Privacy Preserving**:** to ensure that there is no way for unauthorized parties and malicious insiders to access the sensitive data content from the cloud
2. Index Privacy**:** the search index or the query index does not leak any information about the corresponding keywords
3. Efficiency**:** the above goals should be achieved with less computation and communication overhead
4. Data Integrity: detect the modifications or deletions of data and maintain the consistency of data

3.5. Preliminaries and notations

- C – the total file collection denoted as a set of n data files $C = \{F_1, F_2, \ldots, F_n\}$.
- w –the keywords denoted as a set of m words $w_i = \{w_1, w_2, \ldots w_m\}$
- id(F_{ij}) – the file identifier F_{ij} that can help to uniquely identify the actual file.
- I – the index built from the file collection, including a set of posting lists $\{I(w_i)\}$
- Tw_i – the trapdoor for search request of keywords w_i.
- H – is cryptographic hash function like SHA−1 H: $\{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}_2^{\log(l)}$.

4. Efficient and Secure Privacy-Preserving Approach(ESPPA)

The existing TRSE scheme (Yu et al., 2013) has been proposed based on fully homomorphic encryption and ranked keyword search for privacy of outsourced data. However, homomorphic

encryption increases heavy computational burden on the data owner side of mobile resource-constrained mobile devices and ranked keyword search process would increase the high communication overhead due to the two-round communication between the cloud server and the authorized user for file retrieval.

To alleviate the computation and communication overhead and ensure the privacy of outsourced data of resource-constrained mobile devices in the cloud, we propose an Efficient and Secure Privacy-Preserving approach (ESPPA) using probabilistic public key encryption and ranked keyword search. In our scheme, the data owner creates an index for file collection then encrypts the both index and files. Later, the authorized user generates a query and sends to the server. When the cloud server receives a query, it searches for corresponding files, and sends top-k matched files to the authorized user. Then, the user decrypts the files and gets the original data. The ESPPP consists of three phases: (1) Setup phase, (2) Retrieval Phase and (3) Integrity verification

In setup phase, the data owner first generates public and private key pairs. Then builds the index from multiple keywords extracted from file collection, then calculate the relevance score and add to index post list. Then, to ensure the privacy of the index and file collection, the data owner encrypts the both. Finally, the data owner distributes the encrypted files and index to the cloud server.

Later, in retrieval phase, the data owner or authorized user generates trapdoor for set of keywords and send to the server. Then, the server search for the matched files and their corresponding relevance scores based on trapdoor. If keywords match with index, the server ranks the matched files based relevance score and send the files to the user in a ranked ordered manner. Then the data owner or user decrypts the file using private key

In integrity verification, the user verifies the integrity of both file collection and index i.e. stored data in cloud is safe or not?

4.1. Setup phase

In setup phase, the data owner pre-processes the file before outsourcing it into the cloud. It consists of three algorithms: (1) Key Generation, (2) Index Creation and (3) Privacy-Preserving.

4.1.1. Key generation

In this algorithm, the data owner generates key pairs as follows: the data owner chose the two large primes and calculates $N = pq$. Next computes r and s using extended Euclidian algorithm where $pr + qs = 1$. Then public key is $PK = \{N\}$ and private key is $PR = \{p, q, r, s\}$. The process of key generation is illustrated in Algorithm 1.

Algorithm 1. Key Generation

1 Select two large random primes p,q
2 Calculate $N = pq$
3 Compute r & s using Euclidian algorithm where $pr + qs = 1$
4 Public Key $PK = \{N\}$ and Private key $PR = \{p, q, r, s\}$

4.1.2. Index creation

After generating key pairs, the data owner creates an index for file collection. Without loss of generality, we use Wang's scheme (Wang et al., 2012) as the base of our index creation. The process of index creation is illustrated in Algorithm 2.

Algorithm 2. Index Creation

1 **Procedure: Index Creation**
2 Scan the file **Collection *C***
3 Extract keywords $w_i = \{w_1, w_2, \ldots w_m\}$ from the C
4 **for** each i from 1 to m
5 $I(w_i) = id(F_{ij})$
6 end **for**
7 for j from 1 to w_i
8 $S_{ij} = \sum_{t \in i} \frac{1}{j} \cdot (1 + \ln f_{d,t}) \cdot \ln(1 + \frac{N}{f_t})$
9 end **for**
10 end **for**
11 $I(w_i) = (id(F_{ij}) \| (S_{ij}))$
12 return I

The detail of index creation in Algorithm 2 as follows:

1) the data owner scan the file collection $C = \{F_1, F_2, \ldots, F_n\}$ and extract the its multiple keywords $w_i = \{w_1, w_2, \ldots, w_m\}$ for each file $F_i \in C$, the data owner build index is

$$I(w_i) = id(F_{ij}) / 1 \le i \le n \text{ and } 1 \le j \le m \quad (1)$$

2) Then, calculate the relevance score for set of keywords in file F_{ij} using ranking function denoted as S_{ij}

$$S_{ij} = \sum_{t \in Q} \frac{1}{j} \cdot (1 + \ln f_{d,t}) \cdot \ln\left(1 + \frac{N}{f_t}\right) \quad (2)$$

where i is the set of keywords,

$f_{d,t}$ denotes the term frequency in file F
f_t, denote the number of files that contain term frequency t of term d in F,
N denotes the number of files,
j is the length of the file

3) Add the relevance score to post index:

$$I(w_i) = (id(F_{ij} \| (S_{ij})) \quad (3)$$

We build the index by using the technologies from IR community like stemming that are employed to build searchable index I from file collection C (Witten et al., 1999).

4.1.3. Privacy-preserving

After creating an index, to ensure the privacy of index and files, the data owner encrypts both index and file collection. Due to the limited computing power on the data owner side, we encrypts index and files using probabilistic public key encryption technique (Menezes et al., 1996) instead of Homomorphic encryption (Yu et al., 2013). The procedure of encryption process is given in Algorithm 3 as follows:

1) Let the file $F = \{m_1, .., m_n\}$ with length n, where each m_i is a binary string of length h and index $I(w_i)$
2) Select the random seed t and generate

$$x = t^2 \bmod N \quad (4)$$

3) Generate the pseudorandom bits

$$x_i = x_{i-1}^2 \bmod N \quad (5)$$

$$p_i = x \bmod 2 \quad (6)$$

where p_i is least significant bits of x_i.

Algorithm 3. Encryption

1 **Procedure: Encryption**
2 Owner authentic public key N
3 Let file $f = \{m_i\} 1 \le i \le n$

4 Select r as a seed
5 $x = t^2 \bmod N$
6 for each i from 1 to n do
7 $x_i = x_{i-1}^2 \bmod N$
8 $p_i = x_i \bmod 2$ /pseudorandom sequence bits
9 Compute $I'(w_i) = p_i \oplus I(w_i)$
10 Compute $c_i = p_i \oplus m_i$
11 end for
12 Compute $x_{n+1} = x_n^2 \bmod N$

4) Then, the pseudorandom bit sequence p_i XORed with index and plaintext to get the ciphertext

$$I'(w_i) = p_i \oplus I(w_i) \tag{7}$$

and

$$c_i = p_i \oplus m_i \tag{8}$$

5) Finally, generate next random

$$x_{n+1} = x_n^2 \bmod N \tag{9}$$

After encrypting the data, the data owner sends encrypted file collection and index $C = \{c_1, c_2, \ldots c_n, I'(w_i)\}$ to the CSP. The resulting bit sequence x_{n+1} send to the authorized users or keep it locally.

4.2. Retrieval phase

In this phase, the authorized user retrieves the files from the CSP through ranked keyword search. This phase consists of three methods: (1) Trapdoor Generation, (2) Ranked Keyword Search and (3) Data Decryption

4.2.1. Trapdoor Generation

After storing the data in cloud, whenever the authorized user wants retrieve the file containing certain keywords, computes the trapdoor for keywords $w_i \in w$ and sends to the CSP as search request. The procedure of computing trapdoor is given in Algorithm 4.

Algorithm 4. TrapDoor Generation

1 Procedure: TrapDoor
2 user authenticate key r
3 Compute Trapdoor
4 $T_{w_i} = \sum_{i=1}^{m} H(w_i)^r$
5 Send trapdoor to cloud server

1) The User gets the trapdoor information from the owner
2) For inserted keywords, the user computes the trapdoor is

$$T_{w_i} = \sum_{i=1}^{m} H(w_i)^r \tag{10}$$

where H is a collision resistant hash function like SHA−1, in which case p is 160 bits and r be the random key.
3) Then, the user sends trapdoor (T_{w_i}, k) to the CSP where k is a optional value.

4.2.2. Ranked Keyword Search

In this method, the cloud server searches for the matching files after receiving a Trapdoor T_{w_i} from user as follows:

1) The cloud server first finds the matching entries of file via trapdoor T_{w_i}, if server gets matching file identifiers along with their associated relevance scores $(id(F_{ij}) \| (S_{ij}))$
2) Then, the server ranks the matched files according to relevance scores and sends top-k most relevant files $\boldsymbol{C_i = \{c_1, c_2, \ldots c_k\}}$ $1 \le i \le k$ to the user.

The procedure of ranked keyword search is illustrated in Algorithm 5.

Algorithm 5. Ranked Search Index

1 **Procedure: Ranked Keyword Search**
2 for each level i from 1 to n do
3 if$(I'(w_i) == T_{w_i})$
4 Rank (C_i)/ highest level that match with query
5 end if
6 end for
7 send $C = \{c_1, c_2, \ldots c_k,\}$ to the user

To search the files, we use a B^+ tree-based data structure to get the corresponding file list. We assume that there are n levels in the tree for some integer $n > 1$. For each file, each tree level stores an index for frequent keywords of that file in a cumulative way. The server starts comparing the trapdoor against the first level identity of each file. If matching file found as a result of the comparison in the first level then this process continue to the other levels in tree as shown in Algorithm 5. Therefore, the overall search time is almost as efficient as on unencrypted data. Our search phase focus is on top-k retrieval, the server can process the top- k retrieval almost as fast as in the plaintext domain.

4.2.3. Data Decryption

After receiving the matched files from CSP for corresponding search request, the authorized user decrypts them with the private key and obtains their plain text. The procedure of decryption process is given in Algorithm 6 as follows:

1) To decrypt the file, the user computes

$$d_1 = (p+1)/4)^{n+1} \bmod (p-1) \tag{11}$$

$$d_2 = (q+1)/4)^{n+1} \bmod (q-1) \tag{12}$$

$$a = x_{n+1}^{d_1} \bmod p \tag{13}$$

$$b = x_{n+1}^{d_2} \bmod q \tag{14}$$

$$x = brp + asq \bmod N \tag{15}$$

2) Then, the user uses x to construct x_i and p_i just as data owner did for encryption

$$x_i = x_{i-1}^2 \bmod N \tag{16}$$

3) Finally, the user get the plaintext m_i by XORing the p_i with ciphertext blocks c_i

$$m_i = p_i \oplus c_i \tag{17}$$

Algorithm 6. Data Decryption

1 **Procedure: Data Decryption:** To recover plaintext from C, user should do the following:
2 Compute $d_1 = (p+1)/4)^{n+1} \bmod (p-1)$
3 Compute $d_2 = (q+1)/4)^{n+1} \bmod (q-1)$
4 Compute $a = x_{n+1}^{d_1} \bmod p$

Compute $b = x_{n+1}^{d_2} \bmod q$
Compute $x = brp + asq \bmod N$
for i to n do
Compute $x_i = x_{i-1}^2 \bmod N$
Let p_i be the h least significant bits of x_i
end **for**
Compute $m_i = p_i \oplus c_i$

4.3. Integrity verification

Due to possible data corruption by internal attacks i.e., insider attacker adds in scrambled data into the encrypted files or even to the encrypted index, the search result may return false files to the user. We need a security mechanism to verify the integrity of the search results by the user is desirable.

Therefore, we design an integrity check mechanism over the returned search results to address data corruptions problems. In order to verify the integrity of data, we are using collision-resistant hash function H (e.g., SHA−1) as follows:

1. The data owner computes the hash value of each file and index before storing them in the cloud:

   $h_1 = \{H(I'(w_i) \| \phi(F_i)\}_{1 \le i \le m, 1 \le i \le n}$

   where H is a hash function, m is number of keywords and n is the total number of files.
2. Then, the data owner stores hash values locally or may share with authorized users.
3. Later, upon receiving a query request from the user, the cloud server computes hash value h_2 of file collection, index and sends hash value h_2 along with ranked search results:

   $h_2 = \{H(I'(w_i) \| \phi(F_i)\}_{1 \le i \le m, 1 \le i \le n}$

4. After receiving the search results from the cloud server, the user compares hash value of file collection and index before uploading to cloud and after receiving from the cloud is

   $h_1 = h_2$?

5. if $h_1 = h_2$ the user can be assured that returned files from the cloud server has integrity and maintaining the correct ranking order in index, other wise data are corrupted in cloud.

As a result, the proposed scheme can achieve the integrity of encrypted files and index stored in cloud. In this way, privacy-preserving approach satisfies the design goal of data privacy along with data integrity.

5. Security analysis

In this section, we analyze the security of our scheme against insider and outsider attacks and chosen cipher-text attacks.

Definition 1. (Semantic Security) Semantic security captures our intuition that given a cipher text, the adversary (insider/outsider) learns nothing about the corresponding encrypted plaintext, thus, we can say that it is semantically secure.

Definition 2. (**Data Privacy**) ESPP approach has data privacy, if there is polynomial time adversary that, the given retrieved encrypted data and the corresponding encrypted secret key, learns nothing about the data.

Definition 3. (Index Privacy) A ESPP approach has index privacy, if for all polynomial time adversaries that, given index I for set of keywords, does not leak any information about the corresponding keywords.

Definition 4. (Data Integrity) the ESPP approach has data integrity, if an insider attacker alters or modified the data, they can be detected by users.

Definition 5. A hash function H is collision resistant; if for any polynomial-time probabilistic algorithm A, i.e. $Pr[(x,y) = A(1^k, H) : x \neq y \ \wedge \ H(x) = H(y)]$

is negligible.

Theorem 1. *The ESPPP is semantically secure against insider/outsider attacks according to Definition 1.*

Proof. Here, we have to prove that adversary (insider/outsider) cannot access or learn nothing from the cipher text and index.

Consider our key generation and encryption process: the data owner selects two large primes p, and q, generates public and private key pairs $PK = \{N\}$ and $PR = \{p, q, r, s\}$ and then encrypts the file using public key $x_i = x_{i-1}^2 \bmod N$ $p_i = x_i \bmod 2$ and $c_i = p_i \oplus m_i$.

Observe that N is an integer; an adversary (insider/outsider) can see only the ciphertext c_i. Assuming that factoring N is difficult, and then h least significant bit of the principal square root x_n of x_{n+1} modulo N is simultaneously secure. Thus, the adversary (insider/outsider) can do nothing better than guessing the pseudorandom bits p_i, $1 \le i \le t$. More formally, if the integer factorization problem is hard, then the ESPPP is semantically secure against insider/outsider attacks.

Theorem 2. *The proposed ESPPP satisfies Data Privacy and Index Privacy according Definitions 2 and 3.*

Proof. Here, we prove that data privacy and index privacy against CCA. In this attack, the attacker may have temporary access to encryption files and tries to decrypt it. We do this by modifying our system based on the assumption that the RSA function is intractable. This implementation maintains the same cost of encryption, decryption, and same security against sensitive information attacks as integer factorization problem. Secure the information against the CCA as the deterministic RSA as follows:

Let N be the modules in the RSA module. i.e $N = pq$ where p and q are the primes of the same size. Let the file of the data owner contains such a composite number N whose factors p and q only knows to data owner.

Define p_i to be the bit vector whose value is significant bits of x_i, where $x_i = x_{i-1}^2 \bmod N$. To encrypt the m_i pick a random p_i, then data owner compute $c_i = p_i \oplus m_i$ and $x_{n+1} = x_n^2 \bmod N$. To decrypt m_i, the user computes $x_i = x_{i-1}^2 \bmod N$ and $m_i = p_i \oplus c_i$. The given $x_i = x_{i-1}^2 \bmod N$ is hard for attacker to compute random seed x to access the file? If so, thus one may build extreme efficient encryption scheme. Hence, it is secure against the Chosen-Cipher text Attack (CCA)

Hence, due to security strength of our encryption scheme against insider/outsider attacks and Chosen-Cipher text Attack (CCA) from Theorems 1 and 2, the data privacy and index privacy is well protected.

Theorem 3. *ESPP approach satisfies the data integrity according to Definition 4.*

Proof. We can now prove that ESPP approach has data integrity against insider attacks.

If data is corrupted by insider attacker in cloud environment, the user checks over this attack by matching hash value h_2 from cloud server with hash value h_1 from data owner. So that user can detect that the data modification or alteration to data occurred in

cloud i.e. if $h_1 = h_2$ says that encrypted data and index is not modified in cloud and data integrity is satisfied, other wise encrypted data/index is modified.

Hence, if an insider attacker adds in scrambled data to encrypted files or encrypted index must be detected. In this way, ESPP approach satisfies the integrity of data stored in cloud against insider attacks.

Theorem 4. *The hash function H is collision resistant, it is*

Computationally hard for attacker to find two different inputs $x \neq y$ *that have same hash value* $H(x)=H(y)$ *according to* Definition 5.

Proof. We prove that any attacker cannot produce same hash value on two distinct inputs.

Formally, a pair of inputs x, y are called collision for function H if $x \neq y$ but have same hash $H(x)=H(y)$. The collisions-resistance requirement states that any probabilistic polynomial-time algorithm A, that is on given inputs, succeeds in a finding a collision for the function H with negligible probability.

Therefore, if H is collision resistant, it is computationally infeasible to find two distinct inputs x and y that produce the same hash value, i.e., $H(x)=H(y)$.

Hence, an attacker cannot inject hash a value and thus, it does not make $h_1 = h_2$ due to security strength of hash function.

6. Performance evaluation

In this section, we present performance analysis of the ESPPA, in which communication and computation costs are analyzed separately. Especially, low computation costs on the data owner and authorized users side are crucial for rendering the ESPP approach is feasible for mobile applications where the data owner and users usually perform the all computations through resource-constrained mobile devices such as smart phones.

We have conducted the experimental evaluation of the proposed ESPPP scheme on real data set: Request For Comments (RFC) database (RFC, 2012). Our experiment environment includes the user and server. The user use the C programming language on a Linux machine with dual Intel Xeon CPU running at 2.0 GHz and algorithms use both open ssl and MATLAB libraries and the server use C programming on a Linux machine with Xeon E5620 CPU running at 2.4 GHz. The user acts as a data owner, authorized user, and the server acts as a CSP. The performance of our scheme evaluated regarding the efficiency in terms of computation and communication costs.

6.1. Computation cost

In this section, we evaluate the computation cost of the data owner, authorized user, and cloud server.

6.1.1. Data Owner

Here, we measure the computation cost of the data owner during the setup phase, which includes key generation, encryption and index build algorithms. We mainly concentrating on computation cost of data owner for encrypting the file collection and compare the experimental results with existing scheme.

Fig. 2 shows that ESPPA encryption process is quite efficient because it takes only 1 modular multiplication to encrypt h bits of plaintext. By comparing TRSE encryption technique (Yu et al., 2013), the ESPPA encryption takes less computation cost for longer files (GB).

The computation cost of key generation and index creation is negligible compare to encryption process.

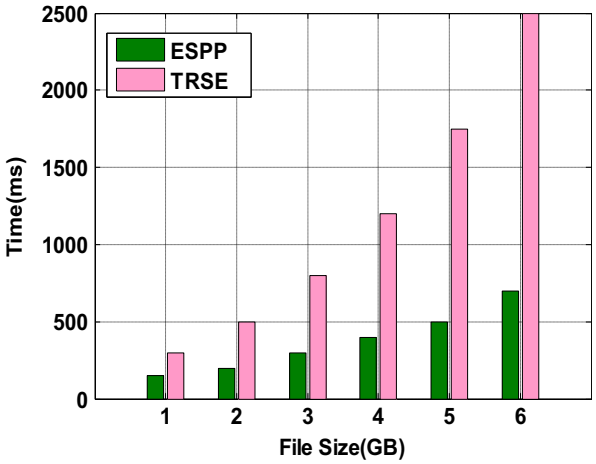

Fig. 2. Computation cost of Data Owner for Encrypting the File.

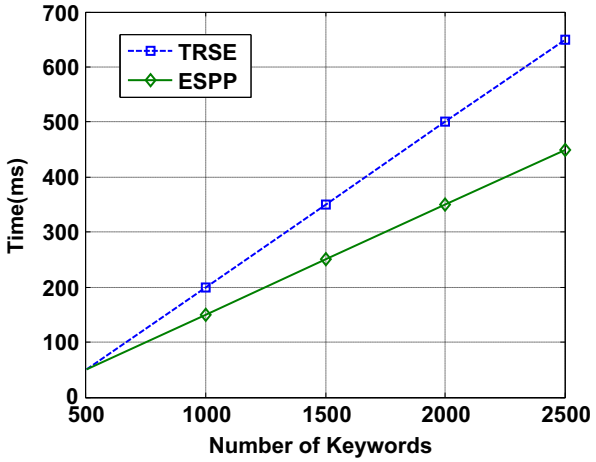

Fig. 3. The time to generate Trapdoor on different number of keywords.

6.1.2. Authorized users

Here, we analyze the computation cost of authorized user to generate trapdoor for multiple keywords and decrypt the retrieved files in retrieval phase.

6.1.2.1. Trapdoor generation. The Trapdoor generation T_{w_i} from multiple keywords requires only addition operations. Fig. 3 shows the time to generate a trapdoor of different lengths of keywords and Fig. 4 shows that time to generate trapdoor for queried keywords where number of keywords are 4000. In both cases, our scheme is efficient than TRSE (Yu et al., 2013), because they encrypt the trapdoor.

6.1.2.2. Data decryption. Upon receiving the files from the server, the user decrypts the files and gets the corresponding plaintext. The probabilistic public-key decryption is quite efficient because its requiring 1 exponentiation modulo $p-1$, 1 exponentiation modulo $q-1$, 1 exponentiation modulo p, 1 exponentiation modulo q, and n multiplications modulo N to decrypt h_n cipher text bits. By comparing homomorphic decryption for longer files, probabilistic public-key decryption takes less computation cost. Fig. 5 shows that the computational cost of user for decrypting large files in ESPPP is less than that in TRSE.

The main difference between proposed ESPPP scheme and TRSE scheme is that the former uses the Homomorphic encryption and decryption algorithms to encrypt and decrypt the files, which require more computation cost, whereas ESPPA uses the probabilistic public

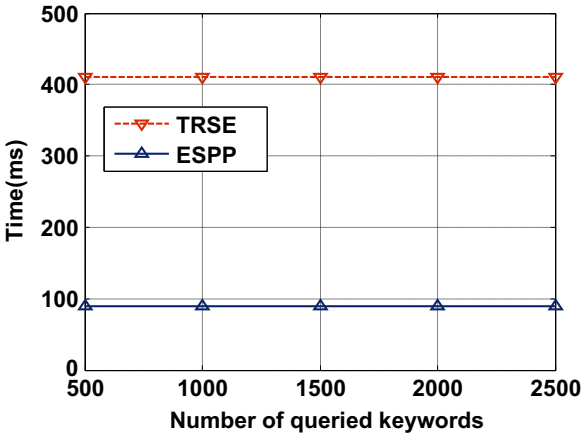

Fig. 4. The time to generate Trapdoor for different number of queried keywords.

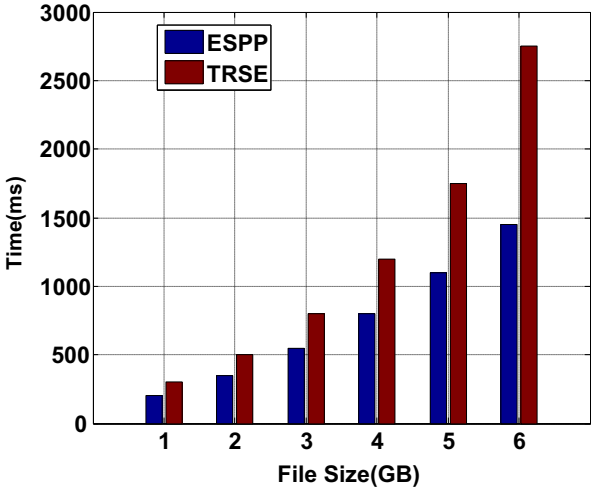

Fig. 5. Computation Cost of Authorized user for decrypting the files.

key encryption and decryption algorithms to encrypt and decrypt the files, which require less computation cost. We claim that ESPPA scheme reduces the computational overhead of a thin client, thus it is more adaptable to resource-constrained mobile devices in cloud environment than TRSE (Yu et al., 2013). To validate our claim, we compared the computation cost of our scheme with existing scheme during the encryption and decryption process as shown in Figs. 2 and 5 respectively.

6.1.3. Cloud server

Here, we measure the time of the server for search the matching files based on search request (trapdoor) generated by the user and for selecting top-k files from total file collection. Fig. 6 shows the time to search the files based on trapdoor; the search time includes fetching the file entry list in the index using B+ tree. The overall search time of ESPPA is almost as efficient as compared existing TRSE (Yu et al., 2013). Similarly, Fig. 7 shows the time of server to select the top-k files from the all matched files based on relevance score computed by data owner. From Fig. 7, we can see that top-k file retrieval time against the value of k increases for the same index of ESPPP performs better than TRSE (Yu et al., 2013) where relevance score calculated at server side. Therefore, computation cost of server for retrieval of top-k files from whole file collection is less compared to existing TRSE (Yu et al., 2013) scheme.

6.2. Communication cost

We analyze the communication cost of proposed ESPPA scheme between the authorized user and server during file retrieval

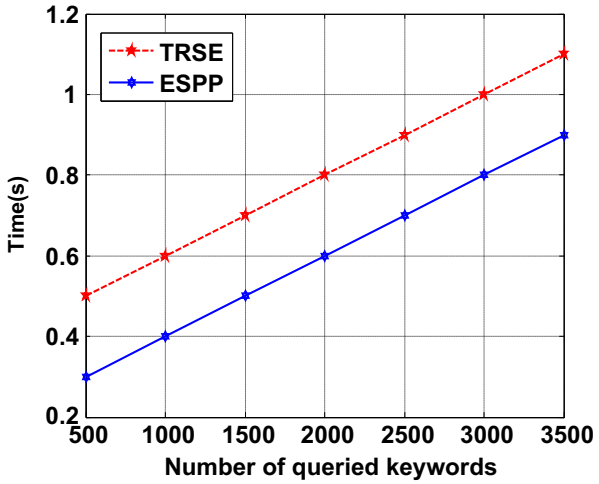

Fig. 6. The time of the server to search the files based on queried keywords.

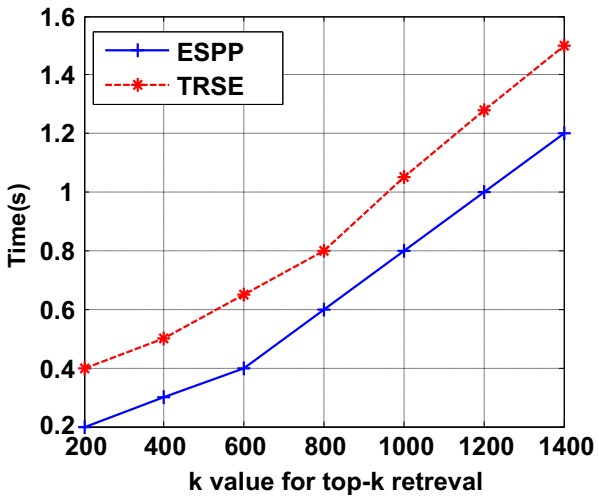

Fig. 7. The time of server for selecting Top-k files where total number of files = 2000.

process. In Fig. 8, we show communication cost of our scheme and compare results with TRSE scheme. From Fig. 8, we can observe that ESPPA vastly reduces the communication overhead burden compared to TRSE (Yu et al., 2013) scheme, because ESPPA use one round communication between the user and server to retrieve the matched files back, where as TRSE scheme uses two round communication process between the user and server.

6.3. Experimental results on real mobiles devices

Moreover, we implement the ESPP approach on real mobile device and Enron Email Dataset (Cohen, 2013). The user uses java language on Android smart phone has 1.2 GHz with 1 GB RAM.

Here, we analyze the computation cost of user for encrypting, decrypting the files and generating trapdoor respectively as shown in Figs. 9–11.

Figs. 9 and 10 show that computation cost of ESPP is less efficient than existing methods such as TRSE (Yu et al., 2013), MRSE (Cao et al., 2014) and RRSE (Wang et al., 2012). Because, the ESPP approach uses the probabilistic public key encryption and Decryption algorithms where as existing methods uses deterministic encryption and decryption algorithms.

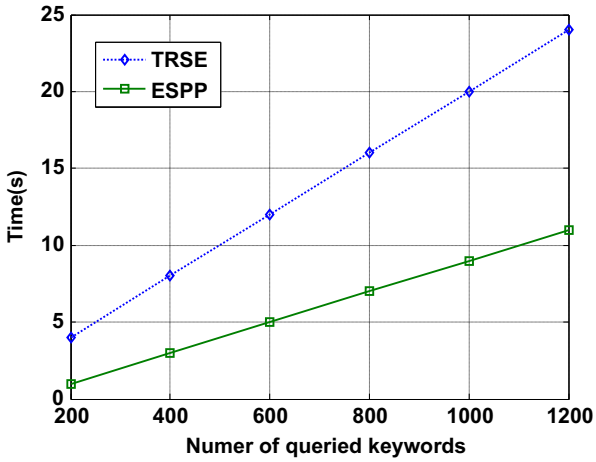

Fig. 8. Communication cost.

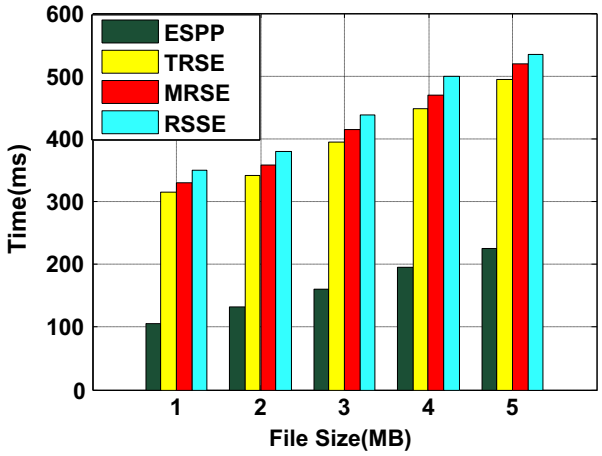

Fig. 10. Computation Cost of Authorized user for decrypting the files on real smart phone.

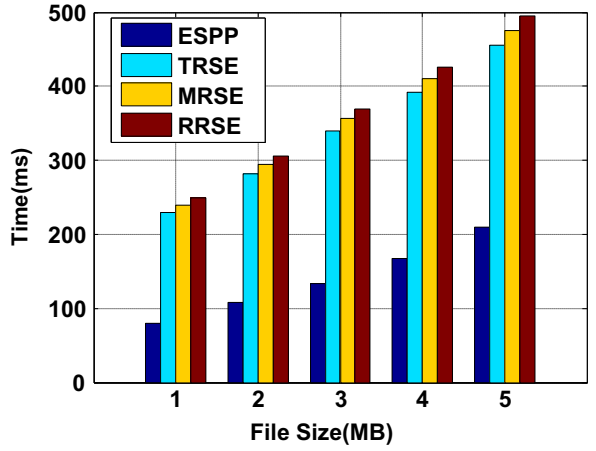

Fig. 9. Computation Cost of Data Owner for encrypting the files on real smart phone.

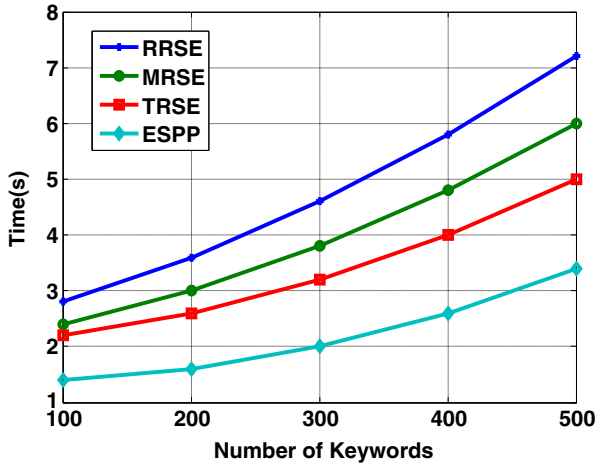

Fig. 11. The time to generate Trapdoor on different number of keywords on real smart phone.

Fig. 11 shows the time to generate a trapdoor of different lengths of keywords. As shown Fig. 11, the time to generate trapdoor is changeless when the number of queried keywords increases. The total time cost of trapdoor generation of ESPP is efficient than existing approaches TRSE (Yu et al., 2013), MRSE (Cao et al., 2014) and RRSE (Wang et al., 2012). Because existing use large key size for generating trapdoor.

7. Discussion on future enhancement

The above discussions have shown how to achieve an efficient ESPP approach. In this section, we give further discussions on how to make the ESPP approach more readily deployable in practice. Based on the current research, there are two interesting research issues to be addressed in ESPP approach.

7.1. Efficiency improvement

During the implementation, we observe that proposed approach achieved the efficiency at the Data Owner side and user side. However, our ranked keyword search mechanism at server side is inefficient when searching for large volumes of data, especially in Big Data environment, which will lead to significant search accuracy performance degradation. This will make it even more challenging to design ranked keyword search schemes that can provide efficient and reliable online information retrieval on large volume of encrypted data.

To solve this problem and, thus, improve efficiency, a tradeoff of the efficiency of ranked search mechanism is needed.

7.2. Dynamic data updates

In Cloud storage, the outsourced file collection may be updated in addition to being retrieved for practical importance. There are three main dynamic data operations such as insertion, modification and deletion for updating documents and corresponding index. Since dynamic data operations also affect the document frequency of corresponding keywords, we also need to update the dictionary and relevance score.

In a practical cloud computing system, data updates like adding, modifying or deleting files and index lead to a new challenge to the ESPP without introducing re-computation overhead on data owners.

Hence, an ESPP approach should support efficient dynamic data operations on documents and index without violating privacy.

8. Conclusions and future directions

In this paper, we addressed the problem of supporting efficient and secure privacy-preserving ranked keyword search over the

encrypted data for achieving effective data utilization of outsourced encrypted data of resource-constrained mobile devices in cloud. The user's data protected against privacy violations. We first presented a basic survey on existing schemes and shown that those are very inefficient to achieve privacy of outsourced data and not suitable for resource constrained mobile devices. Then, we proposed an Efficient and Secure Privacy-Preserving approach based on probabilistic public key encryption and ranked multi-keyword search. We first created an index for file collection and stored both index and file collection in the cloud in an encrypted form. Later, to retrieve data files, the authorized user creates a trapdoor and sends it to the server. Then, server starts search for corresponding files over the encrypted data via trapdoor. The server returns the matching files back to the user, if any file matches with keywords. We appropriately increase the efficiency of our scheme by using probabilistic public key encryption technique rather than other encryption technique for file encryption. Moreover, our scheme also verifies the integrity of data. Finally, we have proved that ESPPA satisfies the security and efficient requirements through the security and performance analysis.

ESPPA is a mechanism that allows a user to search by ranked keywords on encrypted data. It aims at preserving the privacy of the outsourced data of owner while providing a way that allows a user to search efficiently without the need of decrypting the cipher text. Thus, ESPPA has become more important in storage and retrieval of encrypted outsourced data of resource-constrained mobile devices in cloud computing

In our future work, we will enhance ESPP to support efficient dynamic data operations and ranked keyword search over the encrypted big data in cloud (as identified in Section 7).

References

Armbrust M, Fox A, Griffith R, Joseph AD, Katz RH, A. Konwinski A, Lee G, Patterson DA, Rabkin A, Stoica I, Zaharia M. Above the clouds: a Berkeley view of cloud computing, Technical Report UCB-EECS-2009-28. Berkeley: University of California; 2009. p. 1–23.

Attrapadung N, Li Bert B. Functional encryption for inner product: achieving constant size cipher text switch adaptive security or support for negation. In: Nguyen P, Pointcheval D, editors. Public Key Cryptography, 6056 LNCS. Springer Berlin/Heidelberg; 2010. p. 384–402.

Bao F, Deng R, Ding X, Yang Y. Private query on encrypted data in multi-user settings. In: Proceedings of 4th international conference on information security practice and experience. Sydney; 2008. p. 71–85.

Bellare M, Boldyreva A, Neill AO. Deterministic and efficient searchable encryption. In: Menezes A, editor. Advances in Cryptology-CRYPTO 2007, 4622 LNCS. Berlin/Heidelberg: Springer; 2007. p. 535–52.

Boldyreva A, Chenette N, Lee Y, O'Neill A. Order-preserving symmetric encryption. In: Proceedings of 28th annual international conference on theory and applications of cryptography techniques. Springer, Germany; 2009. p. 224–41.

Boneh D, Crescenzo GD, Ostrovsky R, Persiano G. Public key encryption with keyword search. In Proceedings of international conference on theory and applications of cryptographic techniques: advances in cryptology. Switzerland; 2004. p. 506–22.

Cao N, Wang C, Li M, Ren K, Lou W. Privacy-preserving multi-keyword ranked search over encrypted cloud data. IEEE Trans Parallel Distrib Syst 2014;25 (1):222–33.

Chang Y-C, Mitzenmacher M. Privacy preserving keyword searches on remote encrypted data. In: Proceeding of Third International Conference on Applied Cryptography and Network Security. New York; 2005. p. 442–55.

Cloud Security Alliance, Security Guidance for Critical Areas of Focus in Cloud Computing; 2009. ⟨http://www.cloudsecurityalliance.org⟩.

Cohen WW. Enron email data set; 2013 ⟨http://www.cs.cmu.edu/~enron/⟩.

Curtmola R, Garay JA, Kamara S, Ostrovsky R. Searchable symmetric encryption: improved definitions and efficient constructions. In: Proceedings of 13th ACM conference on computer and communication security. Alexandaria; 2006. p. 79–88.

Goh E-J. Secure indexes, Technical Report 2003/216, Cryptology, ePrint Archive; 2003 ⟨http://eprint.iacr.org⟩.

Haclgiimfi H, Iyer B, Li C, Mehrotra S. Executing SQL over encrypted data in database-service-provider model, Technical Report TR-DB−02−02. Irvine: Database Research Group at University of California; 2002.

Jaeger PT, Lin J, Grimes JM. Cloud computing and information policy: computing in a policy cloud? J Inform Technol Polit 2009;5(3):269–83.

Jaeger T, Schiffman J. Outlook: Cloudy with a chance of security challenges and improvements. IEEE Secur Priv 2010;8(1):77–80.

Katz J, Sahai A, Waters B. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Proceedings of 27th annual international conference on the theory and applications of cryptographic techniques. Berlin, Heidelberg; 2008. p. 146–62.

Krebs B. Payment processor breach may be largest ever; 2009. ⟨http://voices.washingtonpost.com/securityfix/2009/01/payment_processor_breach_may_b.html⟩.

Kuzu M, Saiful Islam M, Kantarcioglu M. Efficient similarity search over encrypted data. In: Proceedings of IEEE international conference on data engineering. Washington; 2012. p. 1156–67.

Li J, Wang Q, Wang C, Cao N, Ren K, Lou W. Fuzzy keyword search over encrypted data in cloud computing. In: Proceedings of of IEEE 29th international conference on computer communications. San Diego; 2010. p. 441–5.

Liu Q, Wang G, Wu J. Secure and efficient privacy preserving keyword searching for cloud services. J Netw Comput Appl, 35. Elsevier; 927–33.

Lu Y. Privacy-preserving logarithmic- time search on encrypted data in cloud. In: Proceedings of 19th NDSS. San Diego, California, USA; 2012.

Menezes Alfred J, van Oorschot Paul C, Vanstone Scott A. A hand book of applied cryptography. FL: CRC Press; 1996.

Ogata W, Kurosawa K. Oblivious keyword search. J Complex 2004;20:356–71.

Orencik C, Savas E. An efficient privacy-preserving multi-keyword search over encrypted cloud data with ranking. Parallel and Distributed Databases, 32. Springer; 119–60.

RFC: Request for comments database; 2012 ⟨http://www.ietf.org/rfc.html⟩.

Shi E, Bethencourt J, Chan H, Song D, Perrig A. Multi-dimensional range query over encrypted data. In: Proceedings of IEEE symposium on security and privacy. California; 2007. p. 350–64.

Slocum Z. Your Google docs: soon in search results?; 2009. ⟨http://news.cnet.com/8301-17939_109-10357137-2.html⟩.

Song D, Wagner D, Perrig A. Practical techniques for searches on encrypted data. In: Proceedings of the IEEE symposium on security and privacy. California; 2000. p. 44–55.

Wang C, Cao N, Ren K, Lou W. Enabling secure and efficient ranked keyword search over outsourced cloud data. IEEE Trans Parallel Distrib Syst 2012;23 (8):1467–79.

Waters B, Balfanz D, Durfee G, Smetters D. Building an encrypted and searchable audit log. In: Proceedings of annual network and distributed security symposium. California; 2004.

Witten IH, Moffat A, Bell TC. Managing gig a bytes: compressing and indexing documents and images. Second ed. CA, USA: Morgan Kaufmann Series; 1999.

Yu J, Lu P, Zhu Y, Xue G, Li M. Toward secure multi-keyword top-k retrieval over encrypted cloud data. IEEE Trans Depend Secur Comput 2013;10(4):239–50.