International Journal of
Computer & Software Engineering

Research Article Open Access

A Multi-User Searchable Encryption Scheme with Constant-Size Keys

Yu Jui Chang[*], **Yung Chen Hsieh and Ja Ling Wu**[*]
Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan

Abstract

Cloud storage is widely adopted nowadays. Considering about the data leakage issue, people encrypted the data before uploading them to cloud server. However, due to the loss of data's original properties, it is hard to search the encrypted data directly. To solve this problem, searchable encryption scheme has been proposed to search the data stored on Cloud server in the ciphertext domain. To enhance the searchable encryption scheme's practicability, we propose a scheme, which has constant- size keys, to decrease the corresponding storage requirement. In this work, we also provide efficient mechanisms for the participation and revocation of a user. Therefore, it can be easily applied to storage systems of a University or a cooperate user. More importantly, it is our belief that our work provides a useful function for dealing with the mandating enforcement of General Data Protection Regulation (GDPR). Finally, a prototype based on the proposed scheme has been built to verify the feasibility of our work.

Publication History:

Received: December 17, 2017
Accepted: March 06, 2018
Published: March 08, 2018

Keywords:

Attribute-based encryption, Ciphertext policy, Constant-size keys, Searchable encrypti.

Introduction

Advances in networking technology and an increase in the need for computing resources have persuaded many organizations to outsource their storage and computing needs. This new economical computing and storage model is commonly referred to as Cloud computing. Cloud infrastructures can be roughly categorized as either private or public. In a Private Cloud, the infrastructure is managed and owned by the customer and located on premise. This means the access to customer data is under its control and is only granted to parties it trusts. On the other hand, in a Public Cloud, the infrastructure is owned and managed by a cloud service provider and is located off-premise. This implies that the customer data is outside its control and could potentially be granted to untrusted parties.

By moving their data/computing to the Cloud, customers can avoid the costs of building and maintaining a private Information Technology (IT) infrastructure, opting instead to pay a service provider as a function of its needs. For most customers, this provides several benefits including availability (i.e., being able to access data from anywhere and at any time) and reliability (i.e., not having to worry about backups) at a relatively low cost.

While the benefits of using a Public Cloud infrastructure are clear, it also introduces significant security and privacy risks. In fact, it is well-recognized that the biggest hurdle to the adoption of Cloud storage (and Cloud computing in general) is the concern over the confidentiality and integrity of data. For example, data leakage issue in the Cloud, caused by malicious adversaries or misbehaving Cloud operators, is a big threat to customers' important and/or sensitive data like business secrets or personal health records. The leakage of celebrities' photos in iCloud is a well-known example. The privacy of the involved celebrities has been seriously infringed in these events. Although, lots of consumers may be willing to trade their privacy for the convenience of software services (e.g., web-based email, social activities, calendars, pictures, etc.), it is our belief that, this is not the case for enterprises and government organizations. The General Data Protection Regulation (GDPR) [1] has been approved by the European Union and, once it comes into force in 25 May 2018, will give data subjects significant new rights over how their personal data is collected, processed, and transferred by data controllers and processors. It demands significant data protection safeguards to be implemented by organizations. This newly harmonized European-wide regulation mandates the protection of data about people living in the European Union, by every organization that controls or processes data on people in the EU, regardless of where that organization is located around the world. Clearly, the adoption of GDPR in EU gives the best evidence of our previous claim.

To protect the sensitive data, a common remedy for data leakage is: data owner encrypts all the data with their own key before uploading them to cloud server. In this way, people who are not authorized by the data owner cannot retrieve and decrypt the data stored on Cloud server. The Cloud storage equipped with this kind of protection is called the cryptographic Cloud storage.

After encrypting all the data stored on the Cloud server, searching the stored data brings forth a new challenge. Due to the loss of data's original properties, it is hard to search the encrypted data directly. Therefore, Song, Wagner, and Perrig proposed the first searchable encryption (SE) scheme in 2000 [2]. In the scheme, data owner encrypts the involved keywords and uploads them to Cloud server together with the corresponding encrypted data. For retrieving a keyword matched data, user sends the corresponding trapdoor function of the keyword to the Cloud server for performing search over the encrypted data. In this way, the search in the ciphertext domain becomes desirable and feasible.

A lot of literatures on the subject of SE have been published nowadays. However, some important issues about the practical usage of SE schemes are usually ignored in those literatures. Although combining cryptographic cloud storage and searchable encryption scheme can achieve the basic security requirements of a Cloud storage, some challenges, such as secure communication, storage requirement, and computational complexity, still render SE schemes impractical and inefficient.

[*]**Corresponding Author:** Prof. Ja Ling Wu, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan; E-mail: wjl@cmlab.csie.ntu.edu.tw

[*]**Corresponding Author:** Prof. Yu Jui Chang, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan; E-mail: will@cmlab.ntu.edu.tw

Citation: Chang YJ, Hsieh YC, Wu JL (2018) A Multi-User Searchable Encryption Scheme with Constant-Size Keys. Int J Comput Softw Eng 2: 129. doi: https://doi.org/10.15344/2456-4451/2018/129

A typical example of the impracticality caused by the limited storage capacity is in the applications with lightweight mobile devices. This challenge comes from the fact that in traditional SE schemes the required key size is usually proportional to the number of involved attributes. The limited memory capacity of lightweight devices, such as mobile phones and smart watches, has become a bottleneck for enabling some security related applications, especially when the number of involved attributes is getting larger.

In this work, a multi-user SE scheme with constant-size keys is proposed. In our scheme, the size of keys is independent of the number of the user's attributes. Since the key size is constant and small, our scheme is allowed to be applied to all applications even with limited key storage, such as in lightweight devices. Moreover, the proposed scheme provides efficient mechanisms for the participation and the revocation of a user. For example, it is easy to process the participation and leave of students or staffs of a school. Therefore, it can be easily applied to storage management systems of an organization with data security concerns. We also analyze and investigate the security level of our scheme and present the corresponding executional performance. Finally, a prototype system based on the proposed scheme has been built to verify the feasibility of our work.

Related Work

Searchable Encryption

As we have mentioned in Section 1, the concept of SE scheme was first introduced in [2]. The authors also presented the first practical symmetric searchable encryption (SSE) scheme in that paper. In that SSE scheme, each encrypted document is embedded with a keyed hash for each keyword and the server can search for keywords by recalculating and matching with the hash values. Later, Boneh, Di, Ostrovsky, and Persiano proposed the first asymmetric searchable encryption (ASE) scheme based on the Bilinear pairing in [3]. In the ASE scheme, every customer can use the public key to encrypt the data and keywords but only the user who has the private key can search and decrypt the encrypted data.

Single-user searchable encryption (SUSE) scheme allows only one user in the system. That is to say, the only legal user is the data owner, he himself. It is obvious that SUSE scheme is extremely impractical. In 2006, Curtmola, Garay, Kamara, and Ostrovsky proposed the first symmetric multi-user searchable encryption (MUSE) scheme [4], which made use of an SUSE for keyword search and a broadcast encryption (BE) for key management and user management. In that scheme, data owner is the broadcaster and the group users with keyword search permission are receivers. Data owner shares the key of SUSE to receivers to achieve the goal of multi-user search ability. However, sharing key will lead to the leakage of the key when an untrusted server colludes with revoked users. Later, Bao, Deng, Ding, and Yang proposed an MUSE scheme based on Bilinear pairing in [5]. Their scheme allows every customer in the system to generate the encrypted data and keywords and everyone also can search the data for any keyword. However, [5] did not take the critical requirement of access control into account.

Subsequently, a number of various SE schemes have been proposed for different purposes, including using conjunctive keyword search [13,14] or fuzzy keyword search [15,16] to extend the searching functionality, such as using the scheme against inclusion-relation attacks to improve the security [17,18], and using logarithmic-time search to improve the operational efficiency [19]. Moreover, Lv, Zhang, and Feng proposed an MUSE scheme by using the Ciphertext-Policy Attributed-Based Encryption (CP-ABE) scheme, which provides an efficient access control scheme for Cloud storage applications [12].

Encrypted Access Control for Cloud Storage

In 1984, Shamir introduced the notion of identity-based (ID-based) cryptography and presented a concrete signature scheme [6]. The major merit of an ID-based cryptosystem is that the public key can be an arbitrary string. Therefore, the public key can be set using user's identification information, such as telephone number or email address. Although [6] successfully constructed an ID-based signature, it did not work out how to construct an ID-based encryption (IBE) scheme. This problem had remained open until Boneh and Franklin introduced the notion of bilinear pairings and successfully constructed a concrete ID-based encryption scheme in [7]. Based on it, many IBE schemes with further enhancements have been proposed. These include signature schemes, key establishment schemes, functional (or attribute-based) encryption, and privacy-enhancing techniques such as anonymous credentials. Pairing-based cryptography has also been adopted commercially. The two largest companies in this field are Voltage Security and Trend Micro.

Attribute-based encryption (ABE) scheme is an extension of IBE scheme which allows user to encrypt and decrypt messages based on attributes and access structures. ABE scheme was first introduced by Sahai and Waters in [8]. There are mainly two kinds of ABE schemes: Key-Policy ABE (KP- ABE) [9] and Ciphertext-Policy ABE (CP-ABE) [10]. In a KP-ABE scheme, a ciphertext is associated with a set of attributes and a user's key can be associated with an access structure, which can be a logic composition of attributes, for example. The ciphertext can be decrypted with the user's key if and only if the attribute set of ciphertext satisfies the access structure associated with the user's key. At the end of [9], the authors also suggested the possibility of building a CP-ABE scheme, but did not give any realistic construction.

In 2007, Bethencourt, Sahai, and Waters proposed a CP-ABE scheme [10]. They also provided an implementation of their system, which included several optimization techniques. In a CP-ABE scheme, a ciphertext encrypts a message with an access structure while a user's key is associated with a set of attributes. A user can decrypt the ciphertext if and only if the attribute set fulfils the access structure. For example, suppose that a teacher delivers a course for computer science in NTU. After midterm test, he may want to encrypt the midterm result so that only personnel that have credentials or appropriate attributes can access it. For instance, the teacher may specify the following access structure for accessing the midterm result: "NTU" AND ("Teacher" OR ("Student" AND "Computer Science")). By this, the midterm result should only be seen by the Computer Science Department's students or the teacher himself. Furthermore, both of them are asked for being a member of NTU. In comparison with KP-ABE, CP-ABE is more appropriate for being applied to applications with complicated access control requirement since it enables data owner to choose the access structure for deciding who can access the message.

As a result, many CP-ABE schemes have been proposed for various purposes, such as extending the functionality or improving the security or efficiency of the system. However, most of existing CP-ABE schemes in the literature have linear-size decryption keys. Linear-size decryption keys mean that the size of decryption keys is linearly proportional to the size of the number of involved attributes. This drawback prevents lightweight devices being used as the storage of decryption keys for a CP-ABE scheme, in practice.

For conquering this shortage, Guo, Mu, Susilo, Wong, and Varadharajan has proposed a CP-ABE scheme with constant-size keys for lightweight devices in [11]. In that work, a novel CP-ABE scheme, where the size of decryption keys is independent of the number of involved attributes, was proposed. The size of the decryption key in [11] can be as small as 672 bits, which is suitable for storing CP- ABE keys on lightweight devices.

Preliminaries

For the ease of explanation, some notations and preliminary backgrounds are respectively defined and briefly addressed in this Section, first.

Bilinear Pairing

Let G_1 and G_2be two elliptic curve based groups, and G_T is an elliptic curve based multiplicative group. The three groups are of the same order p. Let g_1 be a generator of G_1 and g_2 be a generator of G_2, and e(g_1, g_2)be a generator of G_T. A bilinear pairing e: $G_1 \times G_2 \rightarrow G_T$ is an injective function with the following properties hold:

1. Bilinearity: For allg ε G_1, h ε G_2 and a,b ε Z_p, we have
$$e(g^a,h^b) = e(g,h)^{ab} \qquad (1)$$
2. Non-degeneracy: e(g,g)≠1
3. Computability: There is an efficient algorithm to computee(g,h) for all g ε G_1 and h ε G_2

CP-ABE with constant-Size keys for lightweight devices

As described in 0, a CP-ABE scheme uses a binary vector to represent an attribute set. For example, let *n*=4. The 4-bit string $\mathbb{A} = 1011$ stands for the attribute set, $\mathbb{A}$, consisting of the attributes {A_1, A_3, A_4}. We use $|\mathbb{A}|$ to denote the number of attributes in $\mathbb{A}$. A CP-ABE scheme is usually composed of four algorithms: Setup, KeyGen, Encrypt, and Decrypt, which can be described as follows.

Setup

Taking a security parameter λ and a universe attribute set {A_1,A_2,...,A_n} as input, the setup algorithm outputs a public key PK and a master secret key MSK. For a random number α ε Z_p, the keys are respectively set as:

$$PK = (h, e(g,h), \mathbb{BG}, H_1, H_2, H_3, H_4,$$
$$\forall i = 1,2,\ldots,n, v_i = g^{\alpha^i}, h_i = h^{\alpha^i}) \qquad (2)$$
$$MSK=(\alpha,g) \qquad (3)$$

In (2), $\mathbb{BG}$ stands for the bilinear pairing that is described above with generators g ε G_1 and h ε G_2. Moreover, H_i for i=1,2,3,4 are hash functions.

Key generation

Taking an attribute set $\mathbb{A}$, a public key PK and a master secret key MSK as input, the key generation algorithm outputs a key which is identified with respect to $\mathbb{A}$. Without loss of generality, let's focus on $\mathbb{A}$=a_1 a_2…a_n which is an attribute string. Pick a random number s εZ_p, compute

$$f(\alpha, \mathbb{A}) = \Pi_{i=1}^{n}\left(\alpha + H_1(i)\right)^{1-a_i} \qquad (4)$$

and generate the secret key for $\mathbb{A}$ as:

$$SK = \left(g^{\frac{s}{f(\alpha,\mathbb{A})}}, h^{\frac{s-1}{\alpha}}\right). \qquad (5)$$

From the definitions of the function $f(\alpha,\mathbb{A})$and the parameter $g^{\frac{s}{f(\alpha,\mathbb{A})}}$, it is easy to check that SK is independent of the number of involved attributes.

Encrypt

Taking an access structure $\mathbb{P}$(also represented in binary form), a public key PK and a message M as input, the encryption algorithm encrypts a message M under an access structure $\mathbb{P}$. Let$\mathbb{P}$=b_1 b_2…b_n be the policy string. Pick a random number σ. Compute

$$r = H_4(\mathbb{P}, \mathrm{M}, \sigma) \qquad (6)$$

and

$$\mathrm{f}(\mathrm{x},\mathbb{P}) = \Pi_{i=1}^{n}\left(\mathrm{x} + \mathrm{H}_1(\mathrm{i})\right)^{1-\mathrm{b_i}}. \qquad (7)$$

Finally, the ciphertext *CT* is outputted as:

$$CT = (\mathbb{P}, C_1 = (h^{f(\alpha,\mathbb{P})})^r = (h^{f_0 \Pi_{i=1}^{n-1} h_i^{f_i}})^r, \qquad (8)$$

$$\forall i = 1,2,\ldots,n-|\mathbb{P}|+1, C_{2,i} = v_i^r, \qquad (9)$$

$$C_3 = H_2(e(g,h)^r) \oplus \sigma, \qquad (10)$$

$$C_4 = H_3(\sigma) \oplus M \qquad (11)$$

According to the definitions of polynomial functions $f(\alpha,\mathbb{A})$given in (4), and $f(x,\mathbb{P})$, given in (7), we know that

$$\frac{f(x,\mathbb{P})}{f(x,\mathbb{A})} = \Pi_{i=1}^{n}(x + H_i(i))^{a_i - b_i} \qquad (12)$$

should be a polynomial, if$\mathbb{P} \subseteq \mathbb{A}$. This can be used to verify the validity of the attribute set.

Decrypt

Taking a ciphertext CT, a public key PK and a secret key SK as input, the decrypt algorithm outputs the message M or⊥(the empty set). The decryption algorithm first computes

$$F(x) = F(x,\mathbb{A},\mathbb{P}) = \frac{f(x,\mathbb{P})}{f(x,\mathbb{A})} = \Pi_{i=1}^{n}(x + H_1(i))^{c_i} \qquad (13)$$

We define F_iε Z_p to be the coefficient of x^i and set F_0≠0. Second, we compute (U,V,W) as

$$U = e\left(C_{2,1}, \Pi_{i=1}^{n-|\mathbb{P}|} h_{i-1}^{F_i}\right) = e(g,h)^{rF(\alpha) - rF_0} \qquad (14)$$

$$V = e\left(\Pi_{i=1}^{n-|\mathbb{P}|+1} C_{2,i}^{F_{i-1}}, h^{\frac{s-1}{\alpha}}\right) = e(g,h)^{rsF(\alpha) - rF(\alpha)} \qquad (15)$$

$$W = e\left(g^{\frac{s}{f(\alpha,\mathbb{A})}}, C_1 \right) = e(g,h)^{rsF(\alpha)} \tag{16}$$

Third, we get e(g,h)^r by the function

$$e(g,h)^r = \left(\frac{W}{U \cdot V} \right)^{\frac{1}{F_0}} \tag{17}$$

Then, we compute the random number σ by

$$\sigma = H_2\left(e(g,h)^r \right) \oplus C_3 \tag{18}$$

Finally, we extract the message M by the function

$$M = H_3(\sigma) \oplus C_4 \tag{19}$$

The Proposed Scheme

The proposed "Multi-User Searchable Encryption Scheme with Constant-Size Keys" is addressed in this Section.

Syetem Model

As shown in Figure 1, our system consists of four players: a trusted authority, a cloud server, a proxy server and a user/data owner.

Cloud server (CS)

A Cloud server abounds with huge storage space and rich computational resources, which is usually operated by the Cloud service provider, such as Amazon and Microsoft.

Proxy server (PS)

A proxy server is introduced to facilitate user's secure usage of Cloud service, which can be deployed inside each enterprise (e.g., under the supervision of Chief Data Protection Officer of an organization). It helps data owners to build the keyword index and helps users to generate the trapdoor. Moreover, PS carries out the operation of user revocation when DO received a revocation order from TA.

System Operation Procedures

System Setup

At the beginning, a trusted authority calls **Setup** to obtain a public key PK and a master secret key MSK. Then he chooses a random number β ε Z_p and sets K_{mk}= β as the search master key. After the generation of these three keys, the trusted authority sends the public key PK to a Cloud server and keeps the master secret key MSK and the search master key K_{mk} in secret.

New User Grant

When a user wants to join the system, the granting procedure is as follows:

1. The trusted authority needs to select a unique identity ID for the user. Then, he has to assign and verify an attribute set associated with the user. The verification of attributes which are claimed by the user cannot be done automatically. Normally, it will be granted by asking the user to provide some proofs of his identity, such as student ID card number or organization's identity card number.
2. After verifying of the attributes, the trusted authority calls Key Gen to generate a secret key SK for the user. For restricting the key size of the user, on the basis of the user ID, the trusted authority computes the control parameters A_{ID} and B_{ID}, which are ID and action-dependent (i.e., they are upload and/or search dependent), and also computes an attribute-set-dependent secret key SKA for the proxy server.
3. The trusted authority randomly chooses a number μ ε Z_p, and sets A_{ID}=μ.

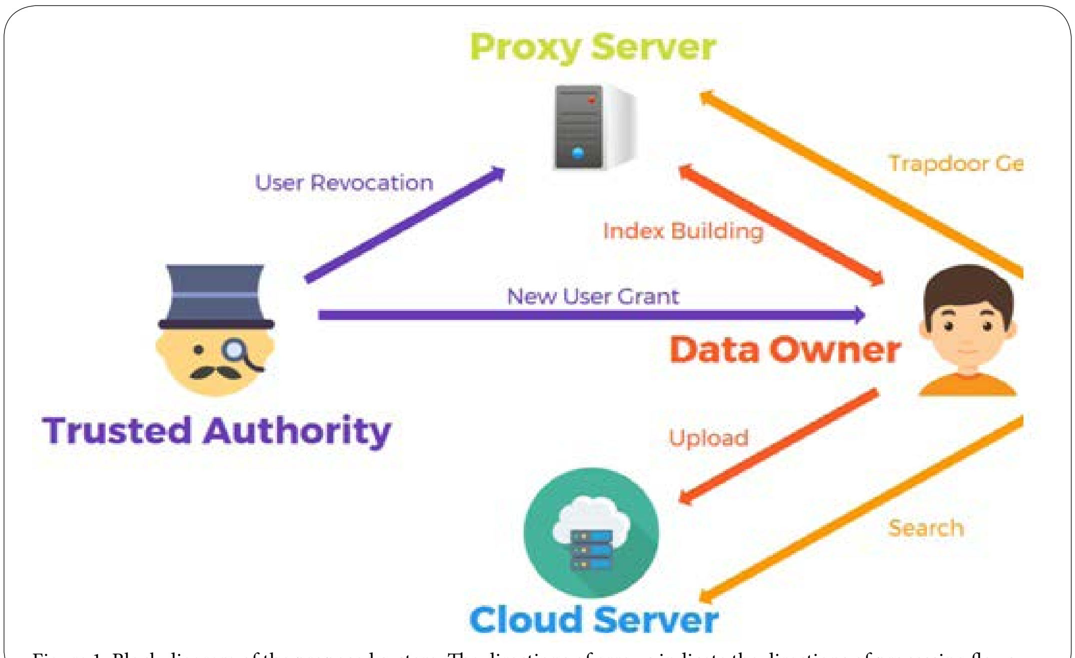

Figure 1: Block diagram of the proposed system. The directions of arrows indicate the directions of processing flows.

4. After getting A_{ID}, the trusted authority uses the search master key K_{mk} and A_{ID} to compute the other control parameter B_{ID} by
$$B_{ID} = g^{K_{mk}/A_{ID}} = g^{\beta/\mu} \quad (20)$$
5. Finally, the trusted authority transmits the tuple (ID,A_{ID},SK) to the user and (ID,B_{ID}) to the proxy server. We can see that the tuple (ID,A_{ID},SK), sent to the user, is independent of the number of involved attributes.

Keyword Index Building

Before uploading the encrypted file, a data owner has to build an encrypted keyword index with the help of the proxy server to let other users be able to search the files. The index building procedure is as follows.

1. We assume D={d_1,d_2,…,d_n} is the keyword set which is extractable from the stored files.
2. For alld_i ε D, data owner calculates
$$T_i = H_5(d_i)^{y_i} \quad (21)$$
3. where y_i ε Z_p is a random number and H_5 is a hash function that maps a keyword to a random element in G_1.
4. Data owner sends his ID and the tuple (T_1,T_2,…,T_n) to the proxy server.
5. After receiving the search request, proxy server finds the corresponding B_{ID} associated with the received ID.
6. For all T_i, proxy server computes
$$L_i = e\left(T_i, B_{ID}\right) \quad (22)$$
and then transmits all L_i back to the data owner.
7. Data owner computes
$$k_i = H_6\left(L_i^{A_{ID}/y_i}\right) \quad (23)$$
and then sets
$$I_{d_i} = (R_i, [R_i]_{k_i}) \quad (24)$$
8. Where R_i is a random number and$[R_i]_{k_i}$ stands for using a secure symmetric encryption algorithm, such as AES, to encrypt R_i with the secret key k_i. H_6 is a hash function that maps an element of G_T to another element of G_T.
9. Finally, the keyword index set is set to be $I_D = \left\{I_{d_1}, I_{d_2}, \ldots, I_{d_n}\right\}$.

Data Upload

After building the keyword index set, data owner randomly selects a symmetric key and encrypts the file by using a symmetric encryption algorithm such as AES. Then, data owner assigns an access structure and encrypts the key by running **Encrypt**. At last, he uploads the keyword index I_D, the encrypted files, and the ciphertext of the symmetric key to Cloud server.

Search in the Encrypted Domain

When a request for searching the encrypted files, stored on the Cloud server, is received he has to do the following three steps (in sequence):

Step 1: Trapdoor Generation

We assume d is the keyword that the user wants to search, he should generate a trapdoor with the help of the proxy server. This step is executed as follows.

1. For the keyword d, the user calculates
$$Q_d = H_5(d)^{A_{ID}} \quad (25)$$
2. then sends ID and Q_d to the proxy server.
3. After receiving the request, proxy server finds the corresponding B_{ID} according to ID.
4. Proxy server computes
$$k' = H_6\,(e(Q_d, B_{ID})) \quad (26)$$
5. and returns k' to the user.

Step 2: Attributes Verification

On receiving the request from a user, the cloud server has to verify the validity of attributes of the user. To verify attributes, the method we mentioned in Section 3 is applied. That is
$$\frac{f(x,\mathbb{P})}{f(x,\mathbb{A})} = \Pi_{i=1}^{n}(x + H_i(i))^{a_i - b_i} \quad (27)$$
should be a polynomial, if$\mathbb{P} \subseteq \mathbb{A}$. We know that if the user has the right to access the file, the access structure $\mathbb{P}$ of the file should be one of the subsets of the user's attribute set $\mathbb{A}$.

Step 3: Search a Keyword

To search a keyword, Cloud server gets $[R_i]_{k'}$ by using k' as the symmetric key and encrypts the random number R_i, which is generated by data owner while building the keyword index. Then, Cloud server checks whether $[R_i]_{k_i}$ is the same as $[R_i]_{k'}$. If they are the same, k_i and k' are referring to the same word, and the searching target is found. Otherwise, they are different and the searching job failed.

File Decryption

After receiving the result from Cloud server, the user running **Decrypt** to extract the symmetric key, SK, which is used to encrypt the file. Then, the user can decrypt the file with the symmetric key.

User Revocation

Because of graduation, retirement or something else, when a user wants to leave the system, his search privilege should be revoked. After completing the verification by the trusted authority, he instructs the proxy server deleting the corresponding tuple (ID,B_{ID}). Without B_{ID}, the user is no longer able to search files stored on the Cloud server.

Security Analysis And Performance Evaluation

Security Analysis

Data Confidentiality

Data confidentiality is an important property of protected data, usually resulting from legislative measures, which prevents it from unauthorized disclosure. In our scheme, data must be confidential against Cloud server, unauthorized users and proxy server. Two encryption algorithms are adopted in our scheme. One is the symmetric encryption algorithm, such as AES, and another is CP-ABE with Constant-Size Keys for Lightweight Devices 0. AES has been proved that it is secure and 0 has also proved its security in the paper. Therefore, it is our belief that the proposed scheme is secure.

Keywords Confidentiality

The keyword must be kept confidential against any entity except for the data owner or the requesting user. Before sending ID and the tuple $(T_1, T_2, \ldots, T_n)$ to proxy server while building keyword index, data owner calculates

$$T_i = H_5(d_i)^{y_i} \quad (28)$$

Moreover, before sending ID and Q_d to proxy server while generating trapdoor function, the user computes

$$Q_d = H_5(d)^{A_{ID}} \quad (29)$$

for the keyword d. Because of the using of one-way hash function, proxy server or any other attackers cannot get any information about the stored keywords.

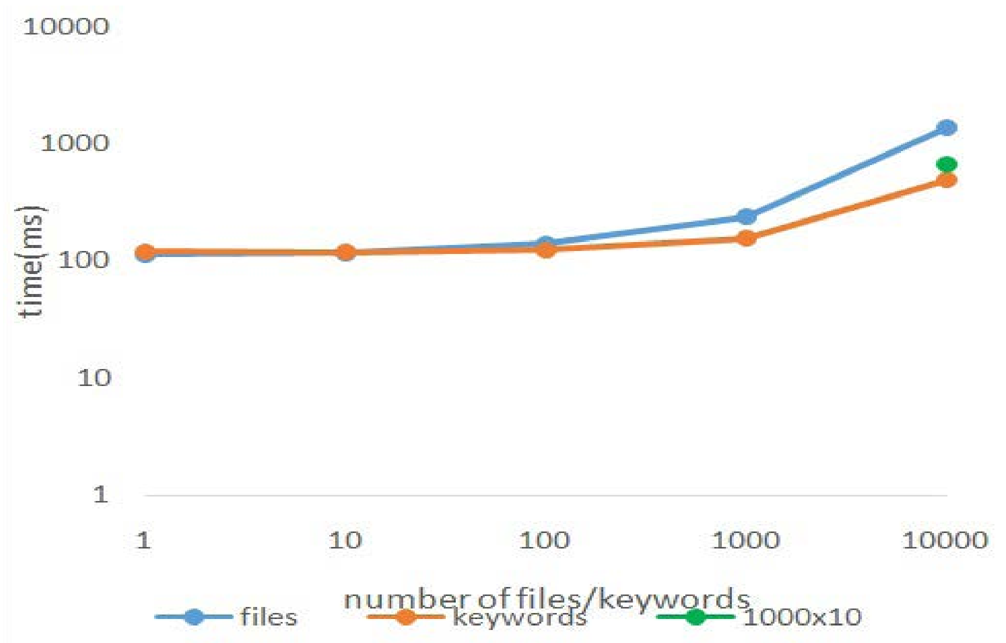

Figure 2: Timing costs for searching various number of keywords (from 1 to 10000)

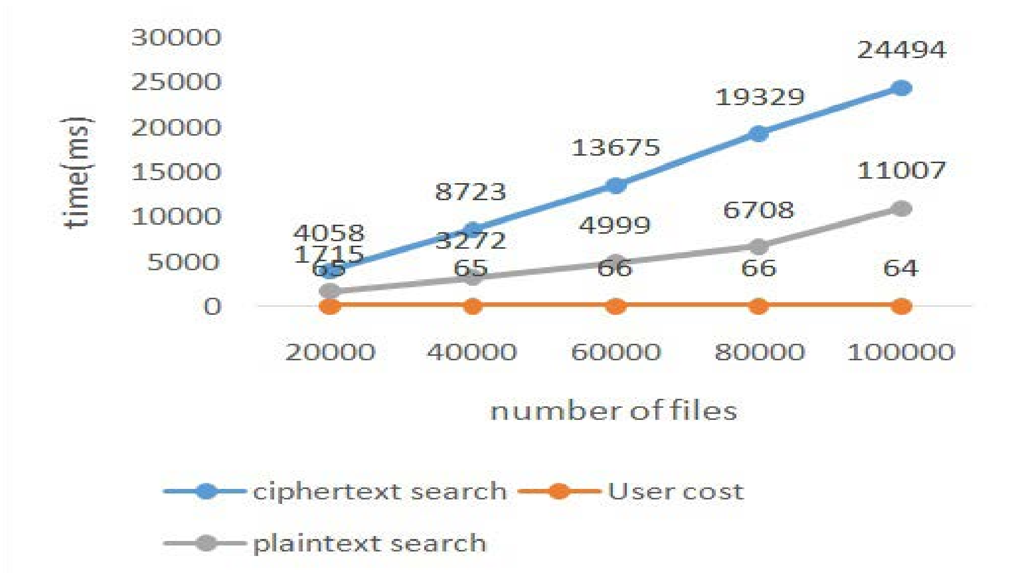

Figure 3: Timing costs for searching various number of files (from 20000 to 100000), in different domains.

Performance Evaluation

Implementation Details

In our implementation, jpbc library is used to implement cryptographic operations. We select the Type-A Bilinear pairing. Type-A Bilinear pairing is the fastest pairing among all types of Elliptical curves, which is constructed on the curve $Y^2=X^3+X$ over the field F_p for some prime p= 3 mod 4. Our system is realized by Java on a personal computer equipped with Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz CPU under Windows10 OS.

Experimental Results

Clearly, from the above figure, timing cost for search depends on the total number of searched keywords. In Figure 2, the line "files" stands for the results of the data set which consists of 10000 files with 1 keyword per file, while the line "keywords" represents the data set which consists of only 1 file with 10000 keywords. While searching, we first locate the position of each file from the table. Then, we search keywords corresponding to the file. Due to the way of searching, the speed of the line "keywords" is a little bit faster than the line "files". The higher cost for the upper case comes from the fact that it ran the search loop much more times than the bottom one. Take 10000 keywords as an example, the former just needs to find the position of the file one time but the latter needs to find it 10000 times. There is a point laying between the two lines, it stands for the data set which consists of 1000 files with 10 keywords. We can see that the speed of that point is in between of the two lines. In Figure 3, we see the timing cost for searching in the ciphertext domain is linearly proportional to the number of searched files while the timing costs for searching in the plaintext domain behaves almost the same. Notice that, the line "User cost" stands for the timing cost spent by the user. It is only a small and negligible portion of the total cost. In Fig. 3, the user is an authorized one and he is searching for all files. That is to say, he has searched 100000 files with 1 keyword by using 24494 milliseconds. It would be faster if the user doesn't have to or doesn't have the authority to search all 100000 files.

Conclusion And Fiture Work

Considering the practical problems of using SE schemes, we proposed and realized a multi-user searchable encryption scheme with constant-size keys. Lightweight devices usually have a limited-memory storage, which could be too small to store large sized keys and/or large number of keys. In our work, we built a provably secure SE scheme which possesses a constant-size key property, that is, its length is independent of the number of file's attributes. The applicability of our system to lightweight devices has been justified by both analytical and experimental results. Since we have not done anything to the files stored on Cloud server, such as sorting or building trees, there is still some room remained for further improving the speed of searching. Of course, it will cost much more time while uploading than searching, as shown in our experiments. If we did something like building trees to the files stored on the Cloud server, it may spend more time while uploading the encrypted data; fortunately, this extra cost is usually endurable to the Cloud server. Therefore, it is worthy of finding a method which can improve the search speed but won't increase too much time while uploading [20]. Of course, this is one of our future search directions.

Technical safeguards, involved with GDPR, include pseudonymization, encryption, and various capabilities for identifying and blocking data breaches, ensuring data security, and automatically identifying and classifying personal data, among others. It is important to note that a "data breach" according to the GDPR also includes "accidental or unlawful destruction, loss, alteration, unauthorized disclosure of, or access to, personal data transmitted, stored or otherwise processed", and so preventing unauthorized use or access must also be considered as a key element of GDPR compliance. Since, in our system, all the stored data are encrypted and searchable; moreover, the access right can be specified by an access structure defined by the user or the trusted authority (i.e., under the supervision of the highest administrator of the organization), it is our belief that the proposed scheme can be viewed as one of the preliminary tools to deal with the looming enforcement of GDPR.

Competing Interests

The authors declare that they have no competing interests.

References

1. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)
2. Song DX, Wagner D, Perrig A (2000) A Practical techniques for searches on encrypted data. Proc. of S&P'00.
3. Boneh D, Di CG, Ostrovsky R, Persiano G (2004) Public key encryption with keyword search. Proc. of EUROCRYPT.
4. Curtmola R, Garay J, Kamara S, Ostrovsky R (2006) Searchable symmetric encryption: improved definitions and efficient constructions. In: Proceedings of the 13th ACM conference on Computer and Communications Security, ACM Press.
5. Bao F, Deng RH, Ding X, Yang Y (2008) Private query on encrypted data in multi-user settings. Proc. of ISPEC.
6. Shamir A (1984) Identity-based cryptosystems and signature schemes. Blakely, Chaum GR, D. (eds.) CRYPTO. LNCS. Springer, Heidelberg 196: 47-53.
7. Boneh D, Franklin M (2001) Identity-Based Encryption from the Weil Pairing. Kilian, J. (ed.) CRYPTO. LNCS, Springer, Heidelberg 2139: 213-229.
8. Waters SB (2005) Fuzzy Identity-Based Encryption. Proc. of EUROCRYPT.
9. Goyal V, Pandey O, Sahai A, Waters B (2006) Attribute-based encryption for fine-grained access control of encrypted data. Proc. of CCS.
10. Bethencourt J, Sahai A, Waters B (2007) Ciphertext-policy attribute based encryption. Proc. IEEE Symp. Security Privacy.
11. Guo F, Susilo W, Wong DS, Varadharajan V (2014) CP-ABE with constant-size keys for lightweight devices. IEEE Trans. Inf. Forensics Security 9: 763-771.
12. Lv Z, Zhang M, Feng D (2014) Multi-User Searchable Encryption with Efficient Access Control for Cloud Storage. Proc. of Cloud Com.
13. Golle P, Staddon J, Waters B (2004) Secure conjunctive keyword search over encrypted data. Proc. of ACNS'04.
14. Hwang YH, Lee PJ (2007) Public key encryption with conjunctive keyword search and its extension to a multi-user system. Proc. of Pairing.
15. Li J, Wang Q, Wang C, Cao N, Ren K, et al. (2010) Fuzzy keyword search over encrypted data in cloud computing. Proc. of INFOCOM.
16. Sedghi S, Liesdonk PV, Nikova S, Hartel P (2010) Searching keywords with wildcards on encrypted data. Proc. of SCN.
17. Islam MS, Kuzu M, Kantarcioglu M (2012) Access Pattern disclosure on Searchable Encryption: Ramification, Attack and Mitigation. Proc. of NDSS.

18. Cai K, Hong C, Zhang M, Feng D, Lv Z, et al. (2013) A Secure Conjunctive Keywords Search over Encrypted Cloud Data Against Inclusion-Relation Attack. Proc. Of CloudCom.

19. Lu Y (2012) Privacy-preserving logarithmic-time search on encrypted data in cloud. Proc. of NDSS.

20. Zhan Q, Su J, Hu Y (2017) Research on encryption Strategy in large data environment based on proxy re-encryption. International Journal of Big Data Intelligence.