Copyright © 2018 Tech Science Press CMC, vol.56, no.2, pp.339-352, 2018

A Proxy Re-Encryption with Keyword Search Scheme in Cloud Computing

Yongli Tang[1], Huanhuan Lian[1], Zemao Zhao[2] and Xixi Yan[1, *]

Abstract: With the widespread use of cloud computing technology, more and more users and enterprises decide to store their data in a cloud server by outsourcing. However, these huge amounts of data may contain personal privacy, business secrets and other sensitive information of the users and enterprises. Thus, at present, how to protect, retrieve, and legally use the sensitive information while preventing illegal accesses are security challenges of data storage in the cloud environment. A new proxy re-encryption with keyword search scheme is proposed in this paper in order to solve the problem of the low retrieval efficiency of the encrypted data in the cloud server. In this scheme, the user data are divided into files, file indexes and the keyword corresponding to the files, which are respectively encrypted to store. The improved scheme does not need to re-encrypt partial file cipher-text as in traditional schemes, but re-encrypt the cipher-text of keywords corresponding to the files. Therefore the scheme can improve the computational efficiency as well as resist chosen keyword attack. And the scheme is proven to be indistinguishable under Hash Diffie-Hellman assumption. Furthermore, the scheme does not need to use any secure channels, making it more effective in the cloud environment.

Keywords: Cloud computing, keyword search, proxy re-encryption, provable security.

1 Introduction

The applications of cloud service on the Internet are greatly promoted by the rapid development of cloud computing technology [Gupta (2015); Li, Chen, Huang et al. (2014)]. Recently, a large number of users choose data outsourcing services due to its convenience and interactivity. Compared with the traditional storage mode, the data outsourcing services do not require the local storage media, which not only save the cost of local storage, but also improve the efficiency of access and increase the security of the data storage. As more and more data which include personal privacy of users and business secrets of enterprises are outsourced to cloud servers, the amounts of data increase exponentially. However, for the legitimate users which request to access encrypted data, how to retrieve the cipher-text data in cloud servers and improve retrieval efficiency are the main challenges in cloud storage.

There are effective ways that users can encrypt their data locally and send to a cloud

[1] College of Computer Science and Technology, Henan Polytechnic University, Jiaozuo, 454000, China.

[2] School of Engineering, Lishui University, Lishui, 323000, China.

[*] Corresponding Author: Xixi Yan. Email: yanxx@hpu.edu.cn.

server to protect the sensitive information. But how to retrieve their data in the cloud server is a difficult problem. There are two ways to solve the problem. The simplest way is that users download all their encrypted data from the cloud server to the local media. Then the users decrypt the cipher-text and complete the corresponding keyword search on the plaintext data. The disadvantages of this way not only waste a lots of network resources and access costs, users will also waste a lot of computational overhead on the decryption of the encrypted data. Another more security idea is that users send their encryption key and search keywords to the cloud server. The cloud server can decrypt cipher-text according to the key. Thus the users complete keyword search in the cloud server. However, this idea makes the data to be stored in the form of plaintext for the cloud server. It is a great threat to the security of the sensitive information. Therefore, it has theoretical value and practical significance to research the keyword search of cipher-text data in the cloud environment.

1.1 Related work

In order to retrieve the encrypted data efficiently, the first construction of searchable encryption scheme based on symmetric algorithm which supported keyword search work in the cipher-text data was proposed by Song et al. [Song, Wagner and Perrig (2000)] in 2000. In recent years, searchable encryption has been researched and developed by the researchers [Terrorism (2013); Tan, Chin, Poh et al. (2015); Wen, Lu, Lei et al. (2014)]. The public key encryption with keyword search based on a bilinear pairing [Dan, Crescenzo, Ostrovsky et al. (2004)] was proposed, which provided a theoretical guide for the subsequent implementation of the cipher-text retrieval in the public key cryptosystem. Next, Wei et al. [Wei, Yang and Chen (2013)] proposed RSA's multiplicative homomorphism to ensure the storage security of privacy in a database. The searchable encryption schemes establish a foundation for the retrieval of cipher-text from encrypted databases. Furthermore, the proxy re-encryption with keyword search schemes are researched on the searchable encryption schemes [Blaze, Bleumer and Strauss (1998); Wang, Huang, Yang et al. (2012)].

Blaze et al. [Blaze, Bleumer and Strauss (1998)] first proposed the concept of proxy re-encryption (PRE) in the European cryptography conference in 1998, PRE is a cryptographic primitive, where a (potentially untrusted) proxy do not know why you mean here? There are problems with this sentence and it needs to be fixed without being able to see anything about the encrypted messages. After the PRE is proposed, it has attracted widely attention and has been studied by the domestic and foreigner researchers. Also it has been popularly applied in the cloud environment. Different technologies are used in the PRE schemes for application in the cloud storage environment [Shi, Liu, Han et al. (2014); Liu, Guo, Fan et al. (2018)]. In order to securely search encrypted messages and delegate the decryption right, Shao et al. [Shao, Cao, Liang et al. (2010)] firstly proposed the concept of proxy re-encryption with keyword search scheme (PRES) which combined the public key encryption with keyword search [Dan, Crescenzo, Ostrovsky et al. (2004)] and proxy re-encryption, and proved the security of the scheme in the random oracle model. However, in 2011, Chen et al. [Chen and Li (2011)] proposed that the security of the Shao's scheme was built at the cost of reduced computational efficiency.

In 2014, Lee et al. [Lee and Lee (2014)] improved the PRES scheme in, which effectively improved the search speed and storage capacity of the scheme. But there was no security proof in the scheme. Chen et al. [Chen, Li, Guo at al. (2014)] proposed a fine-grained access control scheme based on PRES, which limit the user's authority and resist collusion attack. Guo et al. [Guo and Lu (2014)] proposed a scheme which combined ideas of literature [Shao, Cao, Liang et al. (2010)] and [Rhee, Park, Susilo et al. (2010)]. Also they extracted the definitions of PRES and security model. In the re-encryption phase, the scheme only needs to re-encrypt partial file cipher-text. But the calculation efficiency of the scheme can be further optimized.

1.2 Our contribution

In this paper, a new proxy re-encryption with keyword search scheme, based on bilinear pairings, is proposed to solve the above problem. Compared with the traditional schemes, the re-encryption phase in our scheme does not need to re-encrypt partial file cipher-text, but only re-encrypt the cipher-text of keyword corresponding to the file. Besides, the security of our proposed scheme is analyzed and we could draw a conclusion, that is, in Hash Diffie-Hellman assumption, our scheme is proven to be indistinguishable under adaptive chosen keywords attack. Furthermore, the proposed scheme does not need to use any secure channels and strongly unforgeable one-time signatures, making it more effective in the cloud environment. The proposed scheme in this paper is safer compared with other similar schemes.

1.3 Paper organization

The rest of the paper is organized as follows. Section 2 gives some preliminaries of the bilinear map and Hash Diffie-Hellman assumption. In Section 3, we define our PRES scheme and give its security model. The flow and the construction of the scheme are shown in Section 3.2. Then we prove the security of the scheme and compare its efficiency with similar schemes. Finally, we conclude this paper in Section 4.

2 Preliminaries

2.1 Bilinear maps

Let G_1 and G_2 be two cyclic groups with the same large prime order q. A bilinear map *e*: $G_1 \times G_1 \to G_2$ should satisfy the following characteristics:

1) Bilinear: for any $a,b \in Z_p$, $g \in G_1$, we have $e\left(g^a, g^b\right) = e\left(g,g\right)^{ab}$.

2) Non-degenerate: for any $g \in G_1$, a bilinear map satisfies $e\left(g,g\right) \neq 1$.

3) Computable: for any $g \in G_1$, there is a polynomial time algorithm to compute $e\left(g,g\right)$.

2.2 Headings

The trapdoor security of our scheme is based on the Hash Diffie-Hellman (HDH) assumption [Abdalla, Bellare and Rogaway (2001)]. We define the HDH assumption in *G* as follows:

Let *hLen* be a security parameter and H: $\{0,1\}^* \to \{0,1\}^{hLen}$ be a safe and anti-collision hash function. Given $(g, g^a, g^b, H(g^c)) \in G^3 \times \{0,1\}^{hLen}$ and H: $\{0,1\}^* \to \{0,1\}^{hLen}$ as inputs, if $ab = c$, output "*yes*", and "*no*" otherwise. If

$$\left| \begin{array}{l} Pr\left[\Phi(g, g^a, g^b, H(g^{a \cdot b})) = \text{“}yes\text{”} \ : g \leftarrow G, a, b \leftarrow Z_p\right] \\ -Pr\left[\Phi(g, g^a, g^b, \eta) = \text{“}yes\text{”} \ : g \leftarrow G, \eta \leftarrow \{0,1\}^{hLen}, a, b \leftarrow Z_p\right] \end{array} \right| \geq \varepsilon \quad (1)$$

We say that the HDH assumption holds in *G* if no *t*-time algorithm has an advantage at least ε in solving the HDH problem in *G*.

3 A proxy re-encryption with keyword search scheme

First the definitions of PRES system model and security model of trapdoor indistinguishability are given. Then the process of PRES scheme is shown in Fig. 1. Then we describe the algorithm to construct the proposed scheme and prove its security. Finally, its efficiency is compared with congener schemes.

3.1 The basic definition

A PRES scheme is based on the searchable encryption scheme. It mainly includes three entities which are the cloud server S, the data owner A and the data receiver B.

Definition1. A PRES scheme consists of the following polynomial time algorithms.

1) $GlobSetup(1^k)$. Input the security parameter 1^k, this algorithm outputs the global system parameters *params*.

2) $KeyGen(params)$. Input the global parameters, this algorithm generates public and private key pairs of the data owner *A*, the data receiver *B* and the cloud server *S*, then outputs the corresponding key pairs (pk_a, sk_a), (pk_b, sk_b) and (pk_s, sk_s).

3) $Enc(M, sk_a, pk_s, \omega_i, x_i)$. Input the global parameters, pk_s, sk_a and a random number x_i. Separately encrypt file plaintext *M*, file index and the keyword corresponding to the file ω. This algorithm outputs file cipher-text *C*, the encrypted file index *FID*, and the keyword cipher-text $\tilde{\omega}$.

4) $ReKeyGen(a,b)$. Input the global parameters, security parameters *a*, *b*, $a, b \in G_1$, this algorithm outputs the re-encryption key $Rk_{A \to B}$.

5) $ReEnc(\tilde{\omega}, Rk_{A \to B})$. Input the re-encryption key $Rk_{A \to B}$ and the keyword cipher-text $\tilde{\omega}$, this algorithm outputs the re-encryption keyword cipher-text $\tilde{\omega}^{Rk_{A \to B}}$.

6) $Trapdoor(sk_b, \omega)$. Input sk_b and the searching keyword ω, this algorithm outputs the trapdoor T_ω which is associated with the keyword ω. Besides, the trapdoor will not leak any information about the keyword or file plaintext.

7) $Test(T_\omega, \Gamma)$. Input the trapdoor T_ω and trapdoor checking algorithm Γ, the cloud server *S* tests whether the stored re-encrypted keywords contain the querying keyword. If it holds, this algorithm outputs "yes" and continues to decrypt, otherwise, returns "$\perp$".

8) $Dec(C, sk_b)$. Input sk_b and file cipher-text C, this algorithm outputs file plaintext M.

The definition of trapdoor security is given in Shi et al. [Shi, Liu, Han et al. (2014)]. The trapdoor security asks that an outside attacker (excluding the server and the receiver) cannot distinguish the trapdoors between the two challenge keywords, ω_0 and ω_1, of its choice, under the situation that it is allowed to obtain trapdoors for any non-challenge keywords, i.e. $\omega \neq \omega_0, \omega_1$.The trapdoor security is indistinguishable under adaptive chosen keywords attack. To describe the security of the trapdoor, we define the following game between an attacker and the challenger. In addition, we define the data receiver *B* and server *S*in the game.

Game:

1) Setup: Run the global parameter generation algorithm: $GlobalSetup(\lambda)$, and the two key generation algorithms: $KeyGen_B(params)$ and $KeyGen_S(params)$. Parameters: pk_b and pk_s are sent to A while sk_b and sk_s are kept secret from A.

2) Phase 1 (Trapdoor queries): A selects the keyword ω for trapdoor queries and C can adaptively answer $T_\omega = dTrapdoor(sk_b, pk_s, \omega)$.

3) Challenge: A selects two challenged keywords ω_0 and ω_1 which cannot be the keywords in the phase 1, and sends them to C. C randomly selects $\delta \in \{0,1\}$ and computes the trapdoor $T_\omega = dTrapdoor(sk_b, pk_s, \omega_\delta)$. Then T_ω is sent to A.

4) Phase 2 (Trapdoor queries): A makes trapdoor queries of ω_δ, but $\omega_\delta \neq \omega_0, \omega_1$.

5) Guess: A outputs the guess $\delta' \in \{0,1\}$. If $\delta = \delta'$, A wins the game.

In a PRES scheme, the advantage of A in breaking trapdoor indistinguishability is defined as:

$$Adv_{dPRKS,A}^{dTrapdoor-ind-cka}(\lambda) = \left| Pr[\delta = \delta'] - 1/2 \right| \quad (2)$$

Definition2. We say that a PRSE scheme satisfies trapdoor indistinguishable under adaptive chosen keywords attack if for any polynomial time attackers A, $Adv_{dPRKS,A}^{dTrapdoor-ind-cka}(\lambda)$ is negligible.

3.2 Our proposal

3.2.1 The scheme model

In this paper, the proxy re-encryption scheme with keyword search mainly includes the data owner, data receiver, cloud server (main server and content server) and other entities, as well as re-encryption generation, re-encryption and search process. The model is shown in Fig. 1.

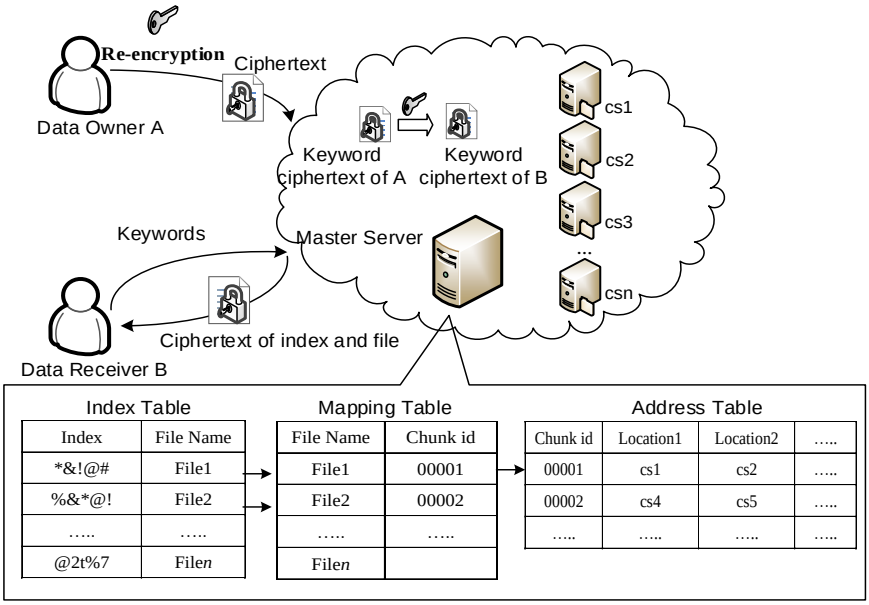

Figure 1: The model of PRES

The data of the owner are divided into the files, the file index and the corresponding keywords of the file, respectively, in which the cipher-text of the file index is mapped to form the address table and stored in the main server. The file cipher-text is divided into blocks, according to the corresponding address table stored in the content server; in the re-encryption stage, the main server re-encrypt the keyword cipher-text that stored in the main server. The data receiver uses the private key and searches keywords to generate trapdoor, the master server checks the generated trapdoor. If the trapdoor is valid, the master server reads the cipher-text file according to the address table and sends the cipher-text file to the data receiver. The data receiver decrypts the cipher-text.

3.2.2 The flow of the scheme

There are some algorithms as followed:

1) The data owner *A*, the data receiver *B* and the cloud server *S* generate public and private key pairs according to the $KeyGen(params)$ algorithm. The public keys are overt and the private keys are kept secret by their owners.

2) Enc Phase: *A* divides the data into three parts, these are, file, file index and the keyword corresponding to the file, and then encrypts them for cloud storage, respectively. The file is encrypted by the symmetric key k_i in the encryption process; file index is encrypted by the symmetric key k_i and random number x_i while *A* computes $y_i = Enc_{pk_b}(x_i)$ and then sends y_i to *B*; the keyword corresponding to the file is encrypted by the encryption key of *A*. After the encryption phase, *A* will send the cipher-text of the three parts to *S* for storage.

3) RKGen Phase: The private key of *B* is carried out using the specified hash function and the encrypted hash message is sent to *A*. *A* calculates the re-encryption key by the $ReKeyGen(a,b)$ algorithm and sends the re-encryption key to *S*.

4) ReEnc Phase: *S* uses encryption key to re-encrypt the keyword cipher-text. *S* utilizes a re-encryption key that allows it to translate the keyword cipher-text under the encryption key of *A* into the keyword cipher-text under the encryption key of *B*, without being able to see anything about the encrypted messages.

5) TGen Phase: *B* inputs the searching keyword that produces the trapdoor according to $Trapdoor(sk_b,\omega)$ algorithm and sends the trapdoor to *S*.

6) Test Phase: *S* tests the trapdoor which is given by *B*in accordance with $Test(T_\omega,\Gamma)$ algorithm. If the input keyword is an effective keyword query, *S* will send the index cipher-text and the cipher-text file to *B*. *B* continues to decrypt; otherwise, *S* will cease operation.

7) Dec Phase: According to the index cipher-text, *B* computes $x_i = Dec_{sk_b}(y_i)$ using its private key. The symmetric key of the cipher-text file is calculated, and the plaintext is obtained according to the decryption of the cipher-text file.

The work process of proxy re-encryption with keyword search is shown in Fig. 2.

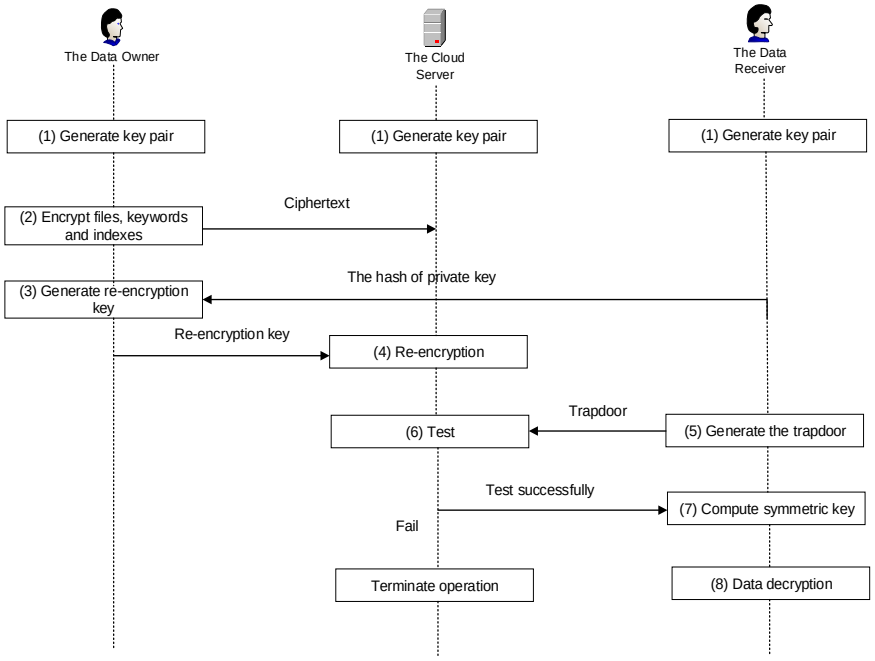

Figure 2: The work process of proxy re-encryption with keyword search

3.2.3 Construction for PRES

Given two cyclic groups G_1 and G_2 of prime order *p* and a bilinear map *e*: $G_1 \times G_1 \to G_2$, where *g* is a generator of G_1. Then it chooses a hash function $H_1: \{0,1\}^* \to G_1$, and a hash function $H_2: \{0,1\}^{\le l} \to G_1$, where H_1, H_2 are random oracles. Let 1^k be a security parameter. Finally, let k_i be a symmetric key that is randomly generated by the symmetric encryption algorithm. Our PRES scheme is constructed as the follows:

1) $GlobSetup(1^k)$. Input a security parameter 1^k $k \in N$. Algorithm outputs system global parameters $params = (p, G_1, G_2, e, g, H_1, H_2, l, k_i)$.

2) *KeyGen* (*params*). Input *params* and randomly select $x \in Z_p^*$, sets $sk = x$, $pk = g^x$. The data owner *A*, the data receiver *B* and the cloud server *S* generate public and private key pairs, respectively.

A: $sk_a = a$, $pk_a = g^a$

B: $sk_b = b$, $pk_b = g^b$

S: $sk_s = s$, $pk_s = g^s$

3) $Enc(M, sk_a, pk_s, \omega_i, x_i)$. Input the file plaintext *M*, the public key of cloud server *S*, the private key of the data owner *A* and the keyword corresponding to the file ω_i. The data owner *A* selects random number x_i and carries out the following steps:

①According to the symmetric key k_i, *A* encrypts the file plaintext *M*. Output the file cipher-text *C*:

$$C = Enc_{k_i}(M) \tag{3}$$

②Input the symmetric key k_i, pk_s and random number x_i, output the file index cipher-text *FID*:

$$FID = e(H_2(pk_s), x_i) \cdot k_i \tag{4}$$

$x_i \in \{0,1\}^*$. The random number x_i is encrypted using the public key of the data receiver *B*, that is, $y_i = Enc_{pk_b}(x_i)$. Then, A sends y_i to *B*.

③Input pk_s, sk_a and the keyword corresponding to the file ω_i, output the keyword cipher-text $\tilde{\omega}_i$.

$$\tilde{\omega}_i = e(pk_s, H_1(\omega_i)^r)^{H_2(sk_a)} \tag{5}$$

Where *r* is selected by *A*, $r \in Z_p^*$. The data owner *A* sends the file cipher-text *C*, file index cipher-text *FID* and the keyword cipher-text $\tilde{\omega}_i$ to the cloud server *S*.

4) $ReKeyGen\left(H_2\left(sk_a\right),H_2\left(sk_b\right)\right)$. The data receiver B sends $H_2\left(sk_b\right)$ to the data owner *A*. Input $H_2\left(sk_a\right)$ and $H_2\left(sk_b\right)$, the data owner A computes the re-encryption key:

$$Rk_{A\to B}=H_2\left(sk_b\right)/H_2\left(sk_a\right) \tag{6}$$

Output the re-encryption key $Rk_{A\to B}$, the data owner A sends $Rk_{A\to B}$ to the cloud server S.

5) $ReEnc\left(\tilde{\omega}_i,Rk_{A\to B}\right)$. Input the re-encryption key $Rk_{A\to B}$ and keyword cipher-text $\tilde{\omega}_i$. The cloud server S computes the re-encrypted keyword cipher-text $\tilde{\omega}_i^{Rk_{A\to B}}$:

$$\tilde{\omega}_i^{Rk_{A\to B}}=\tilde{\omega}_i^{H_2(sk_b)/H_2(sk_a)}=e\left(pk_s,H_1\left(\omega_i\right)^r\right)^{H_2(sk_b)} \tag{7}$$

Output the re-encrypted keyword cipher-text $\tilde{\omega}_i^{Rk_{A\to B}}$ for the cloud server *S*. In this phase, the cloud server cannot obtain any information of the plaintext.

6) $Trapdoor\left(\omega_i,sk_b\right)$. Input the private key of the data receiver *B* and the searching keyword ω_i. Output the trapdoor T_ω which is associated with the keyword ω_i.

$$T_{\omega_i}=\left[T_1,T_2\right]=\left[g^{r'},H_1\left(\omega_i\right)^{1/sk_b}\cdot H_1\left(pk_s^{r'}\right)\right] \tag{8}$$

7) *Test*. Input the trapdoor $T_{\omega_i}=\left[T_1,T_2\right]$, the private key of the cloud server *S* and trapdoor checking algorithm $\Gamma=T_2/H_1\left(T_1^s\right)$.The cloud server *S* performs the following tests using formula (9). It checks whether the stored re-encrypted keywords contain the querying keyword. If it is true, the output is "yes" and continues to decrypt, otherwise, returns "$\perp$" and terminate operation.

$$\tilde{\omega}_i^{Rk_{A\to B}}=e\left(pk_b^r,\left(\Gamma\right)^s\right)^{H_2(sk_b)} \tag{9}$$

8) $Dec\left(C,sk_b\right)$. After the test, the cloud server *S* sends the index cipher-text *FID* and the file cipher-text *C* to the data receiver *B*. The data receiver *B* inputs his private key sk_b and y_i which is obtained from Step 3, and computes the random number x_i:

$$x_i=Dec_{sk_b}\left(y_i\right) \tag{10}$$

The data receiver B computes the symmetric key k_i using x_i and *FID*:

$$k_i=FID/e\left(H_2\left(pk_s\right),x_i\right) \tag{11}$$

The data receiver *B* inputs the symmetric key k_i to decrypt the file cipher-text *C*. Output the file plaintext *M*:

$$M=Dec_{k_i}\left(C\right) \tag{12}$$

Correctness:

We assume the querying keyword ω_i which is given by the data receiver *B* to be a valid keyword. Compute the trapdoor checking algorithm Γ using $T_1 = g^{r'}$ and $T_2 = H_1(\omega_i)^{1/sk_b} \cdot H_1(pk_s^{r'})$:

$$\Gamma = T_2 / H_1(T_1^s) = H_1(\omega_i)^{1/b} \cdot H_1(pk_s^{r'}) / H_1\left((g^{r'})^s\right) = H_1(\omega_i)^{1/b} \tag{13}$$

Then the above results are brought into the formula (9):

$$e\left(pk_b^r, (\Gamma)^s\right)^{H_2(sk_b)} = e\left(g^s, H_1(\omega_i)^r\right)^{H_2(sk_b)} = e\left(pk_s, H_1(\omega_i)^r\right)^{H_2(sk_b)} \tag{14}$$

Observe that the formula (9) is established. The proof is completed.

3.3 Security proof

Theorem 1. In the game, our scheme can satisfy the trapdoor indistinguishability (dTrapdoor-IND-CPA) against a chosen keyword attack, under the circumstances that Hash Diffie-Hellman (HDH) is intractable.

Proof. Assume that there exists a malicious attacker *A* broke the trapdoor indistinguishability of the scheme via the advantage of ε. Let *A* be an external attacker and makes q_T trapdoor queries ($q_T \geq 0$). We construct an algorithm Φ which could solve the Hash Diffie-Hellman (HDH) problem in G, because of the advantage of ε' ($\varepsilon' = \varepsilon$). Φ inputs a random HDH challenge $(g, g^\alpha, g^\beta, \eta) \in G^4$ and $H: \{0,1\}^* \to G$, where *H* is a hash function and η is either $H(g^{\alpha\beta})$ or a random element of *G*. Besides, *B* randomly chooses hash functions H_1 in the following operations.

1) Establishment of the system. Algorithm Φ selects the data receiver's secret key $sk_b = \beta$ at random and calculates the data receiver's public key as $pk_b = g^\beta$. Moreover, it chooses a stochastic value $l \in Z_p^*$ and sets the server's public key as $pk_s = (g^\alpha)^l = A^l$. Here, there exists an unknown value, that is, $sk_s = \alpha l$ is uncharted.

2) Stage 1 of trapdoor queries: When *A* conducts a query about the *dTrapdoor*, corresponding to the keyword, *dTrapdoor* algorithm responds as follows. Φ randomly chooses $r' \in Z_p^*$ and computes $T_{\omega_j} = [T_1^*, T_2^*]$ and $T_1^* = g^{r'}$, $T_2^* = H_1(\omega_j)^{1/\beta} \cdot H\left((g^\alpha)^{l \cdot r'}\right)$ where $l \in Z_p^*$ was selected when the system was established. Then, Φ responds to *A* with the trapdoor T_{ω_j} of ω_j.

3) Stage of challenge: *A* outputs two keywords which wish to be challenged on ω_0. Φ generates the challenge trapdoor, $T_{\omega_\delta^*} = [T_1^*, T_2^*]$, as follows. Φ randomly selects a bit $\delta \in \{0,1\}$ and sets $T_1^* = (g^\beta)^{1/l}$, $T_2^* = H_1(\omega_\delta)^{1/\beta} \cdot \eta$, where $l \in Z_p^*$ is the security parameter

that is selected in the Setup phase and $l \in Z_p^*$ is a component of the HDH challenge. Φ responds with the challenge trapdoor, $T_{\omega_\delta^*} = \left[T_1^*, T_2^*\right]$.

If $\eta = H\left(g^{\alpha\beta}\right)$ and $T_{\omega_\delta^*}$ is a valid challenge trapdoor of ω_δ accompanied by randomness δ. Let r'^* equals β/l and computes $T_1^* = g^{r^*} = \left(g^\beta\right)^{1/l}$, $T_2^* = H_1\left(\omega_\delta\right)^{1/\beta} \cdot \eta = H_1\left(\omega_\delta\right)^{1/\beta} \cdot H\left(g^{\alpha l \cdot \beta/l}\right) = H_1\left(\omega_\delta\right)^{1/\beta} \cdot H\left(pk_s^{r'^*}\right)$.

4) Stage 2 of trapdoor queries: *A* can issue trapdoor queries for the keyword ω_δ. The restriction is that $\omega_\delta \neq \omega_0, \omega_1$. Algorithm Φ responds to these queries as Stage 1.

5) Guess: Finally, *A* outputs its guess $\delta' \in \{0,1\}$. If $\delta = \delta'$, then *A* outputs 1, meaning $\eta = H\left(g^{\alpha\beta}\right)$; otherwise, it outputs 0, meaning $\eta \neq H\left(g^{\alpha\beta}\right)$.

Probability Analysis: Assume that there exists an external attacker *A* in the game, who can break the game with a non-negligible advantage of ε. The probability of the algorithms Φ successfully solve the difficult problems will be given:

We show that when the input tuple $H\left(g^{\alpha\beta}\right)$ is sampled from HDH, in which case, *A* must successfully guess ω_δ with the probability of $\left|Pr\left[\delta = \delta'\right] - 1/2\right| \geq \varepsilon$. On the other hand, when the input tuple is sampled from HDH (where η is a random component selected by *G*), then $\eta \frac{n!}{r!(n-r)!}$ and $T_2^* = H_1\left(\omega_\delta\right)^{1/\beta} \cdot \eta$ are uniform and independent in which case A could guess ω_δ with the probability of $Pr\left|\delta = \delta'\right| = 1/2$. Therefore, with *g*, g^a, g^b and *G* being uniform, we have

$$\left|Pr\left[\Phi\left(g, g^\alpha, g^\beta, H\left(g^{\alpha\beta}\right)\right) = 0\right] - Pr\left[\Phi\left(g, g^\alpha, g^\beta, \eta\right) = 0\right]\right| \geq \left|\left(\frac{1}{2} \pm \varepsilon\right) - \frac{1}{2}\right| \geq \varepsilon \tag{15}$$

as required.

Obviously, the probability cannot be ignored, thus the proof is completed. Therefore, if the HDH of G is intractable, our scheme can satisfy the trapdoor indistinguishability.

In the security proof part, we do not take into account the security issue about the encryption of file plaintext *M*, because the file plaintext *M* uses a standard semantics secure symmetric encryption algorithm (such as AES symmetric encryption algorithm) in the encryption phase, which can ensure the security of data storage. The standard semantic security of public key encryption algorithm (such as RSA public key encryption algorithm) is used in the encryption of the file index, therefore the security of the file index cipher-text *FID* does not need to be proven in the security of the game. Because of the complexity of the algorithm based on public key cryptosystem, we can use non-secure channels to communicate between the users. Above all, we only consider the security of the PRES scheme, if the scheme satisfies trapdoor indistinguishability, it is proved that the scheme can resist chosen keyword attack.

3.4 Efficiency comparison

The efficiency comparison between our scheme, Shao et al.'s [Shao, Cao, Liang et al. (2010)] scheme, Fang et al.'s [Fang, Susilo, Ge et al. (2012)] scheme and an efficient proxy re-encryption with keyword search scheme proposed by Lee et al. [Lee and Lee (2014)] are given in Tab. 1. The encryption phase, the re-encryption phase, the trapdoor generation phase and the decryption phase of the schemes are analyzed in detail in terms of the communication cost and computation cost, etc. We denote T_p as the computational cost of a bilinear pairings and T_e as an exponential cost. The efficiency comparison results are shown in Tab. 1.

Table 1: Comparison of schemes efficiency

Scheme	Our scheme	[Shao, Cao, Liang et al. (2010)]	[Fang, Susilo, Ge et al. (2012)]	[Lee and Lee (2014)]
Enc	$2T_p + T_e$	$2T_p + 5T_e$	$8T_e$	$2T_p + 2T_e$
REnc	$T_p + T_e$	$4T_p + T_e$	$T_p + 2T_e$	$T_p + T_e$
TGen	T_e	T_e	T_e	T_e
Dec	T_p	$5T_p + T_e$	$5T_e$	$T_p + T_e$

We understand that because of the properties of the bilinear pairings operations, when the number of embedding is bigger than 1, its computational cost is much greater than an exponentiation over a bilinear group. In the scheme, the computation of the hash function is negligible compared to the exponentiation and bilinear operations. In addition, Shao et al. [Shao, Cao, Liang et al. (2010] and Fang et al. [Fang, Susilo, Ge et al. (2012)] also use strongly unforgeable one-time signatures and verification operations. The specific efficiency is analyzed as follows:

1) Enc Phase: Strongly unforgeable one-time signatures are constructed in the encryption phase in Shao et al. [Shao, Cao, Liang et al. (2010] and Fang et al. [Fang, Susilo, Ge et al. (2012)]. Furthermore, the scheme in Fang et al. [Fang, Susilo, Ge et al. (2012)] does not use bilinear pairings, but there are eight exponentiation operations, while our scheme has only one exponentiation operation. In summary, the scheme in Lee et al. [Lee and Lee (2014)] and our scheme are more efficient, better than the schemes in Shao et al. [Shao, Cao, Liang et al. (2010] and Fang et al. [Fang, Susilo, Ge et al. (2012)].

2) REnc Phase: Our scheme has the same efficiency as the scheme in Lee et al. [Lee and Lee (2014)], which is slightly superior to the scheme in Fang et al. [Fang, Susilo, Ge et al. (2012)]. Compared with the scheme in Shao et al. [Shao, Cao, Liang et al. (2010], the other three schemes have obvious advantages in the bilinear pairings operations.

3) TGen Phase: In the trapdoor generation phase, each scheme is only related to the exponentiation operation. The four schemes have the same computation cost in this phase.

4) Dec Phase: The scheme in Shao et al. [Shao, Cao, Liang et al. (2010] uses multiple bilinear pairings operations. The scheme in Fang et al. [Fang, Susilo, Ge et al. (2012)]

uses verification algorithms and five exponentiation operations. While the scheme in Wang et al. [Wang, Huang, Yang et al. (2012)] uses one more exponentiation operation than our scheme. Therefore, our scheme is superior to the other three schemes in terms of computational efficiency.

4 Conclusions

In this paper, we propose a new proxy re-encryption with keyword search scheme which can be proved secure. The scheme can be effectively applied in the cloud environment. The trapdoor of our scheme is proven to be indistinguishable under Hash Diffie-Hellman assumption, that is, our scheme satisfies trapdoor indistinguishable under adaptive chosen keywords attack. In the re-encryption phase, compared with other traditional schemes, our scheme does not need to re-encrypt partial file cipher-text, and only need to re-encrypt the cipher-text of the keyword corresponding to the file. Compared congener schemes, our scheme can improve the computation efficiency, proving the feasibility and effectiveness of the scheme. The next step is to extend its application in multi-user multi-file in cloud environment, and to study the complete attribute-based re-encryption scheme with keyword search, so that the practicality of the scheme can be further improved.

Acknowledgements: This work is supported by "13th Five-Year" National Crypto Development Fund (No. MMJJ20170122), Zhejiang Provincial Natural Science Foundation of China (No. Y15F020053), the Project of Education Department of Henan Province (No. 18A413001, No. 16A520013), Natural Science Foundation of Henan Polytechnic University (No. T2018-1).

References

Abdalla, M.; Bellare, M.; Rogaway, P. (2001): DHIES: An encryption scheme based on the Die-Hellman problem. *CT-RSA.*

Blaze, M.; Bleumer, G.; Strauss, M. (1998): Divertible protocols and atomic proxy cryptography. *Lecture Notes in Computer Science*, vol. 1403, pp. 127-144.

Chen, X.; Li, Y. (2011): Efficient proxy re-encryption with private keyword searching in untrusted storage. *International Journal of Computer Network & Information Security*, vol. 3, no. 2, pp. 50-56.

Chen, Z.; Li, S.; Guo, Y.; Wang, Y.; Chu, Y. (2014): A limited proxy re-encryption with keyword search for data access control in cloud computing. *International Conference on Network and System Security*, vol. 8792, pp. 82-95.

Dan, B.; Crescenzo, G. D.; Ostrovsky, R.; Persiano, G. (2004): Public key encryption with keyword search. *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 506-522.

Fang, L.; Susilo, W.; Ge, C.; Wang, J. (2012): Chosen-ciphertext secure anonymous conditional proxy re-encryption with keyword search. *Theoretical Computer Science*, vol. 462, no. 1, pp. 39-58.

Guo, L.; Lu, B. (2014): Efficient proxy re-encryption with keyword search scheme. *Journal of Computer Research and Development*, vol. 51, no. 6, pp. 1221-1228.

Gupta, U. (2015): Survey on security issues in file management in cloud computing environment. *International Journal of Computer Applications*, vol. 120, no. 5, pp. 22-24.

Lee, S. H.; Lee, I. Y. (2014): A study of practical proxy re-encryption with a keyword search scheme considering cloud storage structure. *Scientific World Journal*, no. 2, pp. 1661-1667.

Li, J.; Chen, X.; Huang, Q.; Wong, D. S. (2014): Digital provenance: enabling secure data forensics in cloud computing. *Future Generation Computer Systems*, vol. 37, no 7, pp. 259-266.

Liu, Y.; Guo, W.; Fan, C. I.; Chang, L.; Cheng, C. (2018): A practical privacy-preserving data aggregation (3PDA) scheme for smart grid. *IEEE Transactions on Industrial Informatics*, no. 99, pp. 1.

Rhee, H, S.; Park, J. H.; Susilo, W.; Dong, H. L. (2010): Trapdoor security in a searchable public-key encryption scheme with a designated tester. *Journal of Systems & Software*, vol. 83, no. 5, pp. 763-771.

Shao. J.; Cao, Z.; Liang, X.; Lin, H. (2010): Proxy re-encryption with keyword search. *Information Sciences*, vol. 180, no. 13, pp. 2576-2587.

Shi, Y.; Liu, J.; Han, Z.; Zheng, Q.; Zhang, R. et al. (2014): Attribute-based proxy re-encryption with keyword search. *Plos One*, vol. 9, no. 12.

Song, D. X.; Wagner, D.; Perrig, A. (2000): Practical techniques for searches on encrypted data. *Proceeding 2000 IEEE Symposium on Security and Privacy*, pp. 44

Tan, S. Y.; Chin, J. J.; Poh, G. S.; Kam, Y. H. S. (2015): A client-server prototype of a symmetric key searchable encryption scheme using open-source applications. *International Conference on IT Convergence and Security*, pp. 1-5.

Terrorism, W. T. I. (2013): SSE: A secure searchable encryption scheme for urban sensing and querying. *International Journal of Distributed Sensor Networks*, vol. 2013, no. 6, pp. 1-8.

Wang, X. A.; Huang, X.; Yang, X.; Liu, L.; Wu, X. (2012): Further observation on proxy re-encryption with keyword search. *Journal of Systems & Software*, vol. 85, no. 3, pp. 643-654.

Wei, Z.; Yang, Y.; Chen, Z. (2013): Ciphertext retrieval in database based on RSA's multiplicative homomorphism. *Journal of Harbin Engineering University*, vol. 34, no. 5, pp. 641-645.

Wen, M.; Lu, R.; Lei, J.; Li, H.; Liang, X. et al. (2014): SESA: An efficient searchable encryption scheme for auction in emerging smart grid marketing. *Security & Communication Networks*, vol. 7, no. 1, pp. 234-244.