精密かつ柔軟なデータ共有における複数ユーザキーワード検索の提案

趙　方明†‡　　西出　隆志†　　櫻井　幸一†

† 九州大学
819-0395 福岡市西区元岡 744 ウエスト 2 号館 712
‡(株) 東芝　研究開発センター
212-8582 川崎市幸区小向東芝町 1
fangming.zhao@toshiba.co.jp {*nishide*, *sakurai*}@inf.kyushu-u.ac.jp

あらまし ISPEC2011 で、属性暗号システムに基づいた暗号化クラウドストレージにおける精密かつ柔軟なアクセス制御方法を提案した。しかし、このような暗号化クラウドストレージに対してキーワード検索の機能は考慮されていなかった。本論文では、属性暗号に基づいた複数ユーザの暗号化クラウドの検索方法を提案する。特に、初めてのユーザアクセス権限を考慮した暗号化キーワード検索のコンセプトを提案する。提案方式では、クラウドサーバはユーザの復号できるファイル群に絞り込み、そして検索を行う。検索効率と情報漏洩の観点から我々の方式の既存方式に対する優位性を示す。

Multi-User Keyword Search Scheme for Secure Data Sharing with Fine-Grained Access Control

Fangming Zhao†‡　　Takashi Nishide†　　Kouichi Sakurai†

†Kyushu University, Japan
Kyushu University, 744 Motooka, Nishi-ku, Fukuoka, 819-0395,JAPAN
‡Toshiba Corporate Research & Development Center, Japan.
1,Komukai-Toshiba-cho, Saiwai-ku, Kawasaki, 212-8582, JAPAN

Abstract In ISPEC2011, we proposed a data sharing scheme with fine-grained and flexible access control for cloud storage based on attribute-based cryptosystems. However, we did not give the method to perform keyword search over this kind of cryptographic cloud storage. In this paper, we consider the keyword search of a multi-user cryptographic cloud storage using attribute-based encryption. Our scheme proposes a novel keyword search concept for the first time: fine-grained access control aware keyword search. By narrowing the search scope to the user's decryptable files' group, our keyword search scheme can both decrease information leakage from the query process and be more efficient than other existing schemes.

1 Introduction

1.1 Background and Motivation

For reasons of management cost and convenience, users often store their data not on their own machine, but on remote servers, i.e., cloud storage. To address users' concerns of data confidentiality on the cloud storage, a common approach is using cryptography. Encryption at the server's side is not appropriate when the server is not fully trusted. Data owners shall encrypt all data before sending to the cloud servers and later then the encrypted data can be retrieved and decrypted by users who have decryption key. This kind of cloud storage is often called *cryptographic cloud storage* [4]. Even if this will ease user's concerns of data leakage, it also introduces a new problem: because the encryption of data is not meaningful to the cloud servers, many useful data processing operations performed by cloud servers become infeasible. One of the most important data operations for efficient data retrieval and sharing in the cloud is the keyword search.

Many protocols have been proposed to partially solve the above problems. However, most of exist-

ing schemes are limited to the single-user setting where the owner who generates the encrypted data on the cloud is also the single user to perform encrypted keyword searches on it. They can not satisfy the characteristics of cryptographic cloud storage: sharing encrypted data with multiple users who have the appropriate access rights. In the *multi-user cryptographic cloud storage* setting, the data owner shall share his encrypted data to multiple users and also allow users who have the access permission to perform encrypted keyword searches over the owner's shared data on the cloud server side.

In our previous work [1], an attribute-based cryptosystems based multi-user cryptographic cloud storage was proposed for secure data sharing. Users with different attributes can access the cryptographic cloud storage for secure data sharing, the fine-grained and flexible access control is realized by an access tree [5]. However, no specified keyword search method is given in that work. In this paper, for efficient use of cryptographic cloud storage, we are focusing on the keyword search toward the cryptographic cloud storage under the multi-user setting.

1.2 Related Works

Several schemes have been developed to encrypt data on the client-side and enable server-side searches on encrypted data. Song et al. [7] proposed the first practical scheme for searching on encrypted data. The scheme enables clients to perform searches on encrypted text without disclosing any information about the plaintext and the keyword to the untrusted server. The untrusted server cannot learn the plaintext given only the ciphertext, it cannot search without the user's authorisation, and it learns nothing more than the encrypted search results. The basic idea is to generate a keyed hash for the keywords and store this information inside the ciphertext, then the server can search the keywords by recalculating and matching the hash value. In [8], Goh presents a scheme for keyword search on encrypted data using Bloom Filters. Golle et al. [9] first considers keyword conjunctions which is based on pairings on elliptic curves. The first public key schemes for keyword search over encrypted data are presented in [10]. The authors consider a setting in which the sender of an email encrypts keywords under the public key of the recipient in such a way that the recipient is able to give capabilities for any particular keyword to their mail gateway for routing purposes. Their scheme allows multiple users to encrypt data using the public key, but only the user who has the private key can search and decrypt the data. Bao et al. [3] considers the multi-user query over encrypted data. Their scheme allows each user to possess a distinct secret key for generating the encrypted keyword.

1.3 Challenging Issues

For existing keyword search works [7, 8, 9, 10], if we want to apply their protocols to the multi-user cryptographic cloud storage setting, a naive approach is sharing the secret key to all valid users. However, sharing keys is generally not a good idea since it increases the risk of key exposure. The keys must be changed if a user is no longer qualified to access the data. Moreover, changing keys may result in decrypting all the data with the old key and re-encrypting it using the new keys. For the cloud storage with large number of users and files, this is not practical.

All existing schemes have not considered the user's access right while designing the keyword search process on the cloud server's side. (i): In the multi-user's cryptographic cloud storage environment, since not all users can read(decrypt) all data, each user shall not be able to search through data that is not decryptable to that user for the cloud data confidentiality. For example, a curious user of the cloud storage who possesses a common secret key for searching, can perform an investigation over all encrypted data to know whether some data saved on the cloud contains a specified keyword w, by generating an encrypted query based on that keyword. Especially in the cryptographic cloud storage using the latest cryptographic technique: attribute-based encryption(ABE) [5], likes our previous work [1], such result of the curious user's investigation also leaks more confidential information: encrypted data using CP-ABE shows what kind of user attributes are required to decrypt that encrypted data. That is, the relationship between encrypted data and its access requirement can be known to the curious user from his keyword search. (ii): We consider the search efficiency problem. Assuming n and m $(m < n)$ are numbers of total files and decryptable files on the cryptograhic cloud storage for a specific user. Let r be the average number of keywords associated in

a file. Traditionally, the cloud storage's computation cost for executing a keyword search shall be $O(n \times r)$. So the cloud server wastes $O((n-m) \times r)$ computation cost in each query process. Allow the cloud server to narrow the search scope from n to m(to the user's decryptable data group) is a new challenge for the multi-user cryptographic cloud storage.

1.4 Our contributions

Considering the characteristics of multi-user cryptographic cloud storage, main contributions of our work can be summarized as follows.

- We propose a novel keyword search concept: access control aware keyword search. By narrowing the search scope to the user's decryptable data group, our scheme can:
  (1). decrease information leakage from the keyword search process executed between users and cloud servers.
  (2). be more efficient than existing works since our method does not search unrelated files which can not be decrypted by that user.
- Under the multi-user cloud setting, by providing fine-grained and flexible access control to the data on the cloud, not only the data owner, but authorized users can also update the encrypted data and encrypted keyword list.
- Since each user has a distinct key for keyword search, the key management becomes more simple. For example, key update and user revocation can be easily achieved without complicated process of decrypt and re-encrypt.

2 System Models and Definitions

2.1 System Models

We consider a multi-user cryptographic cloud storage which is described in [1]. In this system, a group of authorized users(E.g. readers and writers in [1]) can share encrypted data and perform keyword search on the encrypted data without decrypting them.

Cloud Server: The main responsibility of the cloud storage server is to store and retrieve encrypted data according to authorized users' requests. Moreover, an new external functionality is provided by the cloud server: before executing the keyword search for each user, the cloud server first needs to sort out those files which can be decrypted by that user, then the cloud server searches the keyword only from his decryptable file group;

Trusted authority(TA): Being similar to the assumption in [1], TA is a trusted third party in our system. Firstly, it is responsible for managing all attributes and their related keys used in ABE and ABS. Secondly, about users' keyword search, it is also responsible for user enrollment and revocation, i.e. managing keys for user' query generation.

Users: Being similar to the assumption in [1], we consider a multi users setting. Not only the owner can perform the keyword search, other users, e.g. readers(who has the decrypt right, can read data) and writers(who has both the read right and the update right), can also perform the keyword search corresponding to their access right. Only writers can update the encrypted file and its associated encrypted keyword list.

Data: As described in the previous work [1], all data is encrypted with CP-ABE on the user's side before sending to the cloud storage. Here, the data file can be documents, videos, images, etc. Each file can be associated with a list of keywords which is also encrypted by the user, using a symmetric key(e.g. AES[2]).

2.2 Definitions

In this subsection, we define our scheme: decryptable keyword search for cryptographic cloud storage. Our proposed scheme consists of a tuple of algorithms (Setup, BuildIndex, Write, Query, Search) such that:

- Setup(1^k): The initialization algorithm *Setup* is run by the TA which takes as input the security parameter k and outputs the unique master secret key K_{msk} and the key pair $\langle K_{U_{id}}, CK_{U_{id}} \rangle$ for each valid user whose user ID is U_{id}. The TA respectively distributes the $K_{U_{id}}$ and the $\langle U_{id}, CK_{U_{id}} \rangle$ to the user and the cloud server.
- BuildIndex($w, CK_{U_{id}}$): The *BuildIndex* algorithm is run by the data owner and the cloud server interactively. This algorithm outputs an index I(w) for all keywords $w = \{w_1, w_2, ...\}$.
- Write(I(w), CT, $T_{decrypt}$)): This *Write* algorithm is run by the data owner. After the

owner generate encrypted data CT, $T_{decrypt}$ and the I(w), he writes(or, uploads) them to the cloud server. CT is the ciphertext of the original data. We consider CP-ABE[5] for the data encryption as described in [1].

- Query(U_{id}, Q(w), Sig(Q(w))): The *Query* algorithm is run by the user to generate a trapdoor for the keyword w and query to the cloud server. The user first compute the trapdoor Q(w) by his keys' material, then generate a signature Sig($Q(w)$) of the trapdoor. We consider the ABS[6] for the digital signature. Finally, the user will send the query data: $\langle U_{id}, Q(w), Sig(Q(w))\rangle$ to the cloud server for a keyword w.
- Search(CK_{Uid}, CT, I(w), Q(w), Sig($Q(w)$), T_{sign}): The *Search* algorithm is run by the cloud server. For each user, cloud server will only search for the keyword Q(w) on the data's group which can be decrypted by that user.
- Revoke(U_{id}): The user search revocation algorithm is run by the TA and the cloud server. Given user ID U_{id}, they revoke the user by updating its user's key list $L = L \setminus \langle U_{id}, CK_{U_{id}}\rangle$, then the user is no longer able to search the cloud storage.

3 Technical preliminaries

We build on the work by Bethencourt et al.[5] and Maji et al.[6] respectively (We do not describe their computation process here, please refer to their works for details). We also review some notions about efficiently computable bilinear maps.

3.1 Ciphertext-Policy Attribute-Based Encryption

CP-ABE[5] is one of the latest public key cryptography primitives for secure data sharing. More precisely, a user's private key will be associated with an arbitrary number of attributes expressed as strings. When a party encrypts a message, they first specify an associated access structure over attributes. A user will only be able to decrypt a ciphertext if that user's attributes satisfy the ciphertext's access structure. At a mathematical level, access structures in our system are described by a monotone access structure (or access tree) $T_{decrypt}$ [5], where nodes of the access structure are composed of threshold gates and the leaves describe attributes. Usually, AND gates can be constructed as n-of-n threshold gates and OR gates as 1-of-n threshold gates. If a set of attributes U satisfies the access structure $T_{decrypt}$, we denote it as $T_{decrypt}(U) = 1$.

Setup is probabilistic and run by the TA: on input the security parameter and a universe of attributes, the master key MK and public key PK are generated.

Encryption$(PK, m, T_{decrypt})$ is probabilistic and run by a user who wants to encrypt a plaintext message m for a user with a set of attributes in the access structure $T_{decrypt}$, this algorithm generates a ciphertext CT.

Decryption(CT, SK) is deterministic and run by a user with a set of attributes U. On input CT and SK, this algorithm outputs the underlying plaintext m, if CT is a valid encryption of m and U is contained in the access structure $T_{decrypt}$ specified in the computation of CT. Otherwise an error will be returned.

3.2 Attribute-Based Signature

Like the CP-ABE, there are two entities in ABS: a central trust authority(TA) and users. The authority is in charge of the issue of attribute private key to users requesting them. Denote the universe of attributes as U, as the access structure in the CP-ABE, there is a a monotone boolean claim-predicate(access structure) T_{verify} over U whose inputs are associated with attributes of U. We say that an attribute set U satisfies a predicate T_{verify} if $T_{verify}(U)$ =1. The algorithms are defined as follows.

Setup The authority obtains a key pair (PK, MK) and outputs public parameters PK and keeps a private master key MK.

Key-Generation(MK, U) To assign a set of attributes U to a user, the authority computes a signing key SK_U and gives it to the user.

Sign$(PK, SK_U, m, T_{verify})$ To sign a message m with a claim-predicate T_{verify}, and a set of attributes U such that $T_{verify}(U) = 1$, the user computes a signature σ by $(PK, SK_U, m, T_{verify})$.

Verify$(PK, m, T_{sign}, \sigma)$ To verify a signature σ on a message m with a claim-predicate T_{sign}, a user runs $Verify(PK, m, T_{sign}, \sigma)$, which outputs a boolean value, accept or reject.

4 Concrete Constructions

We describe details of our fine-grained access control aware multi-user keyword search scheme in this section. As mentioned, access control needs to be enforced before the cloud server searches a keyword and a user is not allowed to search through data which is not decryptable for him. Our scheme is mainly based on the attribute-based cryptosystems as described in [1] and an query protocol Bao et.al. [3].

Setup(1^k): The initialization algorithm Setup(1^k) is run by the TA which takes as input the security parameter 1^k and outputs the TA's unique master secret key K_{msk} and the key pair $\langle K_{U_{id}}, CK_{U_{id}} \rangle$ for each valid user whose user ID is U_{id}, where $CK_{U_{id}} = g^{K_{msk}/K_{U_{id}}}$ is a complementary key for a user. The TA respectively distributes the $K_{U_{id}}$ and the $\langle U_{id}, CK_{U_{id}} \rangle$ to the user and the cloud server.

BuildIndex($w, CK_{U_{id}}$): The BuildIndex algorithm is run by the data owner and the cloud server interactively. This algorithm outputs an index I(w) for the keywords set $w = \{w_1, w_2, ...\}$. Data owner first uploads the $\langle U_{id}, h(w)^r \rangle$ to the cloud server. $h()$: $\{0,1\}^* \rightarrow \mathbb{G}_0$ is the hash function and $r \in \mathbb{Z}_p$ is a random number. After receiving the request, the cloud server calculates the $Cap_w = \hat{e}(h(w)^r, CK_{U_{id}})$ for each w then send it back to the data owner. The data owner can build the index for w as I(w) $= [R, HMAC_k(R)]$, where the key for the $HMAC$ calculation is $k = h(Cap_w^{K_{U_{id}}/r})$, $R \in \mathbb{Z}_p$ is also a random number.

Write(I(w), CT, $T_{decrypt}$): After the owner generated the encrypted data CT and the I(w), he writes(or, uploads) them to the cloud server. CT is the ciphertext of CP-ABE[5] as described in [1]. $CT = Enc(PK_{enc}, M, T_{decrypt})$, PK_{enc} is the public key for encryption, M is the data's plaintext, $T_{decrypt}$ is the access tree for the CP-ABE generated by the owner. Finally, $\langle CT, I(w), T_{decrypt} \rangle$ is write to the cloud server.

Query(U_{id}, Q(w), Sig(Q(w))): For a specific keyword w, the user first generates a trapdoor Q(w) $= h(w)^{K_{U_{id}}}$. Then he generate an attribute-based signature(ABS) for that trapdoor: $Sig(Q(w)) = \{PK, SK, Q(w), T_{sign}\}$. Note that the T_{sign} is made by all of the user's attribute: $T_{sign} = \{Att_1 \wedge Att_2 \wedge Att_3 ...\}$. This signature shows that the user certainly possesses a set of attributes from the authority. The cloud server can verify the user's identification (attribute) by public keys from the TA. In this step, the user sends $\langle U_{id}, Q(w), Sig(Q(w)) \rangle$ to the cloud server.

Search($CK_{U_{id}}$, CT, I(w), Q(w), Sig($Q(w)$), T_{sign}): After receiving the query from a user, the server first checks the complementary key $CK_{U_{id}}$ by the user ID U_{id}. If the U_{id} is valid, the server shall confirm the user's decryptable file group by: (i). Verify user's attribute set $\{Att_1 \wedge Att_2 \wedge Att_3 ...\}$ as described in T_{sign} by the ABS-verification process,: Verify$(PK, Q(w), T_{sign}, Sig(Q(w))) = True$ or $False$. The verification key PK are published by the TA. If the verification result is true, the user's attributes are confirmed. (ii). Using $T_{decrypt}$ of each CT stored on the server, the cloud server can confirm the search scope S as the following program:

```
{
S = Null;
for(i = 0; i < n; i++)
//i is the file index number; n is the total file number.
    {
        if((T_sign ∩ T_decrypt[i])! = null)
        S = S ∪ i;
    }
return S; }
```

Then the cloud server performs the the keyword searche only in the scope S. It first computes $k' = \hat{e}(Q(w), CK_{U_{id}})$, then check each index of the data CT in the scope S as: $HMAC_k(R) \stackrel{?}{=} HMAC_{k'}(R)$. Finally, the server sends the search result to the user.

Revoke(U_{id}): Since TA and the cloud server manage all users' pair $\langle U_{id}, CK_{U_{id}} \rangle$. For a comprised user, TA just instructs the cloud server to delete the entry from the user list L: $L = L \setminus \langle U_{id}, CK_{U_{id}} \rangle$, then the user is no longer able to search the cloud storage.

5 Discussions

In this work, we discuss the security of the cloud server and the user in our access control aware keyword search scheme.

For the cloud server, since both the data and the index of keyword are encrypted before upload to the server, the cloud server does not know the plaintext of them. The server also can not get any information about the secret keys for decryption. In the keyword search process, the infor-

mation allowed to leak to the cloud server is the access(decrypt) right of each user, i.e. from the attribute-based signature, the cloud server can deduce which encrypted file can be decrypted by that user. This kind of information leakage will not be a threat of our scheme.

For the user, since we mainly consider the comprised user in this system, we do not allow any information leakage from the query process. By narrowing the search scope of each user, only the information of decryptable files is returned to the user. So in the multi-user setting, our protocol can avoid the information leakage of the user's undecryptable files, such as which file contains a specific keyword, or, the access control information of other files on the cloud.

6 Conclusions

In this paper, we consider the keyword search of a multi-user cryptographic cloud storage(such as [1]) using attribute-based encryption. Our scheme proposes a novel keyword search concept for the first time: fine-grained access control aware keyword search. By narrowing the search scope to the user's decryptable files' group, our keyword search scheme can both decrease information leakage from the query process and be more efficient than other existing schemes.

Acknowledgments

This work is partially supported by Grant-in-Aid for Young Scientists (B) (23700021), Japan Society for the Promotion of Science (JSPS).

参考文献

[1] F.Zhao, T.Nishide, and K.Sakurai. Realizing fine-grained and flexible access control to outsourced data with attribute-based cryptosystems. The 7th Information Security Practice and Experience Conference(ISPEC2011). LNCS 6672, pp. 83-97, 2011.

[2] FIPS PUB 197, Advanced Encryption Standard, Federal Information Processing Standards Publication 197, Department of Commerce, National Institute of Standards and Technology (NIST), Information Technology Laboratory (ITL), 2001.

[3] F. Bao, R. H. Deng, X. Ding, and Y. Yang. Private query on encrypted data in multi-user settings. In Proceedings of the 4th International Conference on Information Security Practice and Experience Conference(ISPEC2008), pages 71-85. 2008.

[4] Seny Kamara and Kristin Lauter Cryptographic cloud storage In Proceedings of Financial Cryptography: Workshop on Real-Life Cryptographic Protocols and Standardization 2010

[5] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In IEEE Symposium on Security and Privacy, 2007.

[6] Hemanta Maji, Manoj Prabhakaran and Mike Rosulek Attribute-Based Signatures. In Proceedings of the Cryptographer's Track at RSA Conference 2011 (CT-RSA 2011). An full version on Cryptology ePrint Archive: `http://eprint.iacr.org/2010/595`

[7] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In IEEE Symposium on Security and Privacy, , 2000.

[8] E.-J. Goh. Secure indexes. Cryptology ePrint Archive, Report 2003/216, 2003. http://eprint.iacr.org/2003/216/.

[9] P. Golle, J. Staddon, and B. Waters. Secure conjunctive keyword search over encrypted data. In Applied Cryptography and Network Security Conference (ACNS '04), 2004.

[10] D. Boneh, G. di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In EUROCRYPT 2004, volume 3027 of LNCS, Springer.