Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Efficient revocation in ciphertext-policy attribute-based encryption based cryptographic cloud storage*#

Yong CHENG†, Zhi-ying WANG, Jun MA, Jiang-jiang WU, Song-zhu MEI, Jiang-chun REN

(*School of Computer, National University of Defense Technology, Changsha 410073, China*)

†E-mail: ycheng@nudt.edu.cn

Received Aug. 12, 2012; Revision accepted Nov. 6, 2012; Crosschecked Jan. 11, 2013

Abstract: It is secure for customers to store and share their sensitive data in the cryptographic cloud storage. However, the revocation operation is a sure performance killer in the cryptographic access control system. To optimize the revocation procedure, we present a new efficient revocation scheme which is efficient, secure, and unassisted. In this scheme, the original data are first divided into a number of slices, and then published to the cloud storage. When a revocation occurs, the data owner needs only to retrieve one slice, and re-encrypt and re-publish it. Thus, the revocation process is accelerated by affecting only one slice instead of the whole data. We have applied the efficient revocation scheme to the ciphertext-policy attribute-based encryption (CP-ABE) based cryptographic cloud storage. The security analysis shows that our scheme is computationally secure. The theoretically evaluated and experimentally measured performance results show that the efficient revocation scheme can reduce the data owner's workload if the revocation occurs frequently.

Key words: Revocation optimization, Cryptographic access control, Cloud storage
doi:10.1631/jzus.C1200240 **Document code:** A **CLC number:** TP309.2

1 Introduction

Nowadays, there is an emerging trend that increasingly more customers are beginning to use the public cloud storage for online data storing and sharing. However, these data applications in cloud storage are obstructed by some security issues, such as data confidentiality and information leakage. Once the users publish their private data to the public cloud storage, they lose the direct control of their data and have to trust the cloud storage service provider (CSP) reluctantly. Unfortunately, the CSP is usually 'untrusted' and the users' data may be compromised. To protect their sensitive data, customers need to encrypt the data before sending to the cloud storage, and then enforce the self-reliant authorization by building a cryptographic access control system.

Kamara and Lauter (2010) proposed such a secure cloud storage architecture called cryptographic cloud storage. The main idea of this architecture is that the data owner (DO) encrypts the sensitive data before publishing them to the cloud storage, and then distributes the secret keys to all authorized data users (DUs). In the cryptographic cloud storage, the data confidentiality protection and access control enforcement are achieved through the following ways: (1) DO encrypts a file F with a randomly chosen symmetric key k and sends the ciphertext $E_k(F)$ to the cloud storage; (2) If a DU wants to access F, he/she first sends the request to DO, then DO replies the k and access credential via a secure channel; (3) DU retrieves $E_k(F)$ from the cloud storage by the credential and then uses k to decrypt it. This scheme can protect the data confidentiality and

* Project (Nos. 61070037, 61070201, and 61103016) supported by the National Natural Science Foundation of China

A preliminary version was presented at the 2nd International Conference on Cloud and Green Computing, Nov. 1–3, 2012, Xiangtan, China

strengthen the access control in a way in which neither unauthorized users nor the untrusted CSP could obtain the plaintext.

However, the cryptographic cloud storage still has some shortcomings in its performance. Firstly, it is inefficient for DO to distribute the symmetric keys one by one, especially when there are a large number of files shared online. Secondly, the access policy revocation is expensive, because DO has to retrieve the data, and re-encrypt and re-publish it. The first problem can be resolved by using a promising public key algorithm named ciphertext-policy attribute-based encryption (CP-ABE) (Bethencourt *et al.*, 2007). In a CP-ABE based access control system, each DU is assigned with a set of attributes S, and the symmetric key k is encrypted with a certain monotonic access structure T, and then the ciphertext $E'_T(k)$ is published along with $E_k(F)$. A DU will be able to decrypt $E'_T(k)$ and finally obtain F if and only if its attributes set S satisfies T. Thus, DO can distribute k by specifying the access policy instead of sending it one by one. The second shortcoming is regarded as the main challenge of our work.

The main challenge of our work is how to improve the performance of revoking authorized users' permissions. Suppose that a DU with the attributes set S is granted an access to the file F, which means that F is encrypted with k and k is encrypted with an access structure T (S satisfies T). If the DU's access to F is revoked, access to the key k may have already been gained. Thus, the secure and complete revocation procedure (named 'complete revocation') needs to be carried out as follows: firstly, DO retrieves the ciphertext and decrypts it to obtain F; secondly, DO chooses another random key k' and produces a new ciphertext $E_{k'}(F)$; then, DO computes the new access structure T' (S does not satisfy T') and produces the new ciphertext $E'_{T'}(k')$; finally, DO publishes $E_{k'}(F)$ and $E'_{T'}(k')$ to the cloud storage and deletes $E_k(F)$ and $E'_T(k)$. Obviously, the complete revocation will consume a lot of bandwidth and computing resources.

We consider a data storage and delivery system (such as the digital library) as the motivating application. In such a system, the owner publishes the digital resources once. Different subscribers are granted to access the digital resources to which they have already subscribed. Since the subscription relationship changes dynamically, the owner has to update the access policies opportunely. Therefore, the revocation operation will be a sure performance killer. Furthermore, revocation operations will increase customers' economic costs since many commercial cloud storage services (e.g., Amazon's Simple Storage Service (Amazon, 2012)) charge the data transfer under 'pay-per-use' terms.

The cost of the revocation process includes two main aspects. On the one hand, the symmetric key k needs to be updated to k'; i.e., DO will replace $E'_T(k)$ with $E'_{T'}(k')$. On the other hand, DO needs to re-encrypt the file F. To date, there have already been a number of revocation algorithms that can re-encrypt the data by a proxy. However, these algorithms are inefficient, and are usually used for re-encrypting the keys. How to re-encrypt the file F (which may be of huge size) efficiently remains a problem which we try to resolve in this study.

At a high level, our efficient revocation model is a compromise between the complete and the 'lazy' revocation (Backes *et al.*, 2005) schemes. In a lazy revocation scheme, the data re-encryption caused by a revocation will not be executed until the data have been modified for the first time after the revocation (see Section 2 for more details). From a simple view, the complete revocation model carries out all related operations immediately; in contrast, the lazy one postpones as many operations as possible until the data are updated. Our scheme enforces the new policy immediately similar to what the complete revocation does, but it consumes only a little extra bandwidth and a few extra computing resources.

At a technical level, the main idea of our scheme is to perform the revocation operation on a small part of the affected data instead of the entire data. This objective is attained though the following steps. Firstly, the sensitive data F are encrypted with a symmetric key k and split into n slices via an (n, n) secret sharing scheme (Shamir, 1979), and these slices are published to cloud storage. Secondly, k is distributed to the authorized users via the CP-ABE based access control system. Finally, if F is affected by the revocation, we retrieve only one slice of F, and re-encrypt and re-publish it.

Our main contribution lies in that we can reduce the costs of revocation without harming the original security or requiring CSP's assistance. Compared with related works, our scheme has the following advantages:

1. It does not require the CSP be trusted.

2. It requires only a little extra storage.

3. It is computationally secure as the key-based encryption algorithm.

However, our scheme will increase the cost of data publication and retrieval. The data publication is performed by DO, so DO will benefit from this scheme if the revocation operation occurs frequently. Furthermore, DU will takes up more time in the data retrieval process. This is an acceptable sacrifice because DU is spreading in the Internet and he/she can bear some additional latency.

2 Related works

2.1 Revocation schemes

In a fine-grained cryptographic access control system, the flexibility of the access policies is mainly restricted by the performance of the revocation solution. The simplest model for reducing the revocation's consumption is named the lazy revocation scheme (Backes *et al.*, 2005). The main idea of this scheme is to postpone the data retrieval, re-encryption, and re-publication operations until the data are updated. For instance, suppose that a DU is revoked to access the file F. DO will record this change in local storage instead of executing immediately. The expired ciphertexts $E_k(F)$ and $E'_T(k)$ in cloud storage will be updated to $E_{k'}(F')$ and $E'_{T'}(k')$, respectively, when F is modified to F'. Lazy revocation is suitable for some unstrict situations where the files are allowed to remain encrypted with old keys and revocation operations take place occasionally (Blanchet and Chaudhuri, 2008; Kumbhare *et al.*, 2011). However, lazy revocation just evades the revocation problem, which cannot support the secure policies enforcing.

In most actual cases, the new access policy needs to be enforced immediately after a revocation takes place. To support efficient but secure revocation, some researchers proposed key revocation (Yu *et al.*, 2010b; Jahid *et al.*, 2011; Xu and Martin, 2012), proxy re-encryption (Liang *et al.*, 2009; Libert and Vergnaud, 2011), and over-encryption (di Vimercati *et al.*, 2007) schemes. These schemes are running securely under the assumption that there is an 'honest-but-curious' (Samarati and di Vimercati, 2010) third party (such as the CSP or the proxy). An honest-but-curious entity means that it will operate DO's commands honestly, but it still has the curiosity to read the content. In other words, the entity will strictly perform data storing, transferring, deleting, and encrypting operations, but it may disclose the content (both ciphertext and plaintext) in its view.

Key revocation, also named attribute revocation, is accompanied by CP-ABE. This issue is first addressed by adding an expiration date to users' attributes (Bethencourt *et al.*, 2007). This solution disables a user's secret key at a designated time, but cannot revoke the key. Yu *et al.* (2010b) proposed the first formal key revocation solution, where the re-encryption is performed by a proxy. The key revocation algorithm resides in an asymmetric algorithm, so it is too inefficient to be operated on the large-size data. Therefore, the key revocation scheme is suitable only for re-encrypting short message and keys. Jahid *et al.* (2011) adopted this scheme in online social networks and Xu and Martin (2012) employed it for revoking users in cloud storage.

Proxy re-encryption means that the ciphertext can be re-encrypted from one key to another by a proxy without leaking the plaintext. This concept was first introduced by Blaze *et al.* (1998). In this scheme, a proxy is given a re-key, allowing it to translate the data F encrypted under different public keys bi-directionally. Libert and Vergnaud (2011) proposed the first unidirectional proxy re-encryption scheme satisfying chosen-ciphertext security in the standard model. The development of the proxy re-encryption scheme promotes its applications in access control systems (Xu *et al.*, 2012). However, the proxy re-encryption scheme is inefficient for processing large data. Usually, it is used for re-encrypting the symmetric key in the cryptographic cloud storage (Yu *et al.*, 2010a).

In an over-encryption system, the data are encrypted by two layers: they are firstly encrypted by the base encryption layer for providing initial protection, and then encrypted by the surface encryption layer to reflect policy modifications (di Vimercati *et al.*, 2007). The first encryption layer is performed by DO, and the second encryption layer is performed by CSP over the already encrypted data to enforce the access policy changes. After a revocation is raised, DO will not retrieve the affected data or re-encrypt them. Instead, a key and a command are sent to CSP, and then CSP will encrypt

the ciphertext with k' according to the command. Using over-encryption can reduce the revocation's overhead. However, it is not secure in the face of collusion attacks (Foresti, 2010). For example, if CSP reveals the key k' to the revoked users, they will still be able to access the data. What is more, the assumption that there is an honest CSP may not hold in the actual systems.

There are also some data revocation schemes which can revoke users' access policies forever. Lewko *et al.* (2010) discussed how to implement revocation systems with very small private keys. Its techniques can be used to realize attribute-based encryption systems for improving performance. Geambasu *et al.* (2009) and Tang *et al.* (2012) discussed how to achieve data revocation via assured deletion.

2.2 CP-ABE algorithm

Attribute-based encryption (ABE) is derived from identity-based encryption (IBE) (Boneh *et al.*, 2005), where IBE can be viewed as a special case of ABE. Sahai and Waters (2005) introduced fuzzy identity-based encryption as a primitive work of ABE. Later, the ABE systems were divided into key-policy attribute-based encryption (KP-ABE) and CP-ABE by Goyal *et al.* (2006). In KP-ABE systems, an encrypted ciphertext is associated with a set of attributes, and the user's private key represents the access structure. Contrary to KP-ABE, in CP-ABE systems, the user's private key is associated with attributes and an encrypted ciphertext will reflect the access policy. In KP-ABE and CP-ABE systems, a user will be able to decrypt the ciphertext if and only if the attributes set satisfies the access policy.

In the CP-ABE algorithm, the access policy is determined by DO, so it is suitable for access control applications. The CP-ABE scheme consists of four probabilistic polynomial-time algorithms as follows:

1. Setup(λ): This is a randomized algorithm that takes the security parameter λ as input. It outputs the public parameter PK and a secret master key MK.

2. Encrypt(PK, M, A): This algorithm takes the public parameter PK, a message M, and an access structure A over the universe of attributes as input. The encrypting procedure produces a ciphertext CT such that only a user's own set of attributes that satisfy A will be able to decrypt the message. We will assume that the ciphertext implicitly contains access policy A.

3. KeyGen(MK, S): This algorithm takes as input the master key MK and a set of attributes S. It outputs a private key SK described by S.

4. Decrypt(PK, CT, SK): This algorithm takes as input the public parameters PK, a ciphertext CT, and a private key SK. If the attributes set contained in SK satisfies the access policy A in CT, this algorithm will decrypt CT and return a message M.

2.3 Secret sharing schemes

Secret sharing schemes (SSSs) (Shamir, 1979), also named (k, n) threshold schemes, can split a file F into n distinct shares, and a user must retrieve at least k of n shares to reconstruct the file. An attacker with t ($t < n$) shares will get any information about F. Different from key-based encrypting schemes, SSS provides data secrecy without requiring keys updating in long-term storage. To date, many storage systems have used SSS for data secrecy protecting, such as CleverSafe (Resch and Plank, 2011).

In this study, we will focus on a special case of the (k, n) threshold scheme where $k = n$. We can obtain an (n, n) threshold scheme from (k, n) ones by simply letting $k = n$. Also, we can build the (n, n) threshold scheme directly. For instance, an element D can be shared via the following simple scheme (Storer *et al.*, 2007): we first generate random elements $X_1, X_2, \cdots, X_{n-1}$, where the size of X_i is equal to that of D; then we set $X_n = X_1 \oplus X_2 \oplus \cdots \oplus X_{n-1} \oplus D$ and publish n shares $X_1, X_2, \cdots, X_n$. Suppose that the size of D is b bytes. This simple scheme will consume nb bytes, which is equal to that of Shamir (1979). Some (k, n) schemes consume less extra storage; for example, Rabin (1989)'s scheme requires only nb/k bytes, but its security is far less than Shamir (1979)'s (which is information theoretic security).

Similar to the (n, n) threshold scheme, there is an interesting encrypting scheme named all-or-nothing transform (AONT) proposed by Rivest (1997). The AONT scheme can be viewed as an $(n + 1, n + 1)$ threshold scheme, which encodes an original file F into $n+1$ shares. In the AONT scheme, one must decrypt the entire ciphertext before one can determine even one message block. Therefore, unless an attacker can obtain $n + 1$ shares or guess the key correctly, he/she cannot get any information about

F. Furthermore, AONT's performance of encoding operation is only $O(n)$, which is faster than other schemes. This is why AONT has been widely used in data secrecy applications.

3 Revocation optimization

3.1 CP-ABE based cryptographic access control

To describe the efficient revocation scheme more clearly, we will take the CP-ABE based cryptographic cloud storage as a concrete platform. Notice that our discussion is limited in this paper. In fact, the efficient revocation scheme can be applied to other cryptographic access control models.

Fig. 1 shows the outline of the CP-ABE based cryptographic access control model. There are three participants in this model: the data owner, the data user, and the cloud storage. In some realistic systems, the Setup and KeyGen procedures are implemented as an independent third party to reduce DO's workload. In this study, we assume that DO takes control of the whole system, which means DO will start up the system, generate the private keys, and manage all the data and keys.

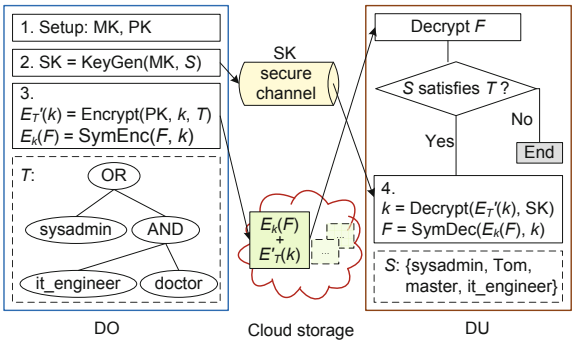

Fig. 1 CP-ABE based access control model in cryptographic cloud storage

The general steps of CP-ABE based access control is presented as follows:

1. DO runs the Setup algorithm and generates the public parameter PK and a secret master key MK.

2. DO runs the KeyGen algorithm and creates each DU's private key SK. This operation embeds DU's attributes set S into SK. DO then sends SK to DU via a secure channel.

3. DO runs the Encrypt algorithm to encrypt a random key k and produce the ciphertext $E'_T(k)$. The ciphertext implicitly contains access tree T. Then, DO can use the key k to encrypt the corresponding data F via a symmetric encryption algorithm SymEnc. Finally, both the ciphertexts $E_k(F)$ and $E'_T(k)$ are published to the cloud storage.

4. DU can perform the Decrypt algorithm if and only if its attributes set S satisfies the access structure T. If S satisfies T, DU decrypts the ciphertext $E'_T(k)$ and obtains k. Then, DU can get the data F via a symmetric decryption algorithm SymDec.

In the above description, we emphasize the fundamental process of the CP-ABE scheme, including system startup, data publication, and data retrieval procedures. The procedure of revocation is the same as in the complete revocation scheme. Now, we begin to discuss how to optimize the performance for the revocation procedure.

3.2 Revocation optimization model

As pointed out earlier, the efficient revocation scheme splits the original data F into a number of slices. To simplify the presentation, we use dynamic data to represent the special slice which is chosen for re-encryption, and the rest of these slices are referred to as static data. Fig. 2 outlines the model of the revocation optimization. The process of CP-ABE based access control has already been illustrated in Section 3.1, so it is removed in this figure for emphasizing the revocation optimization model.

The revocation optimization model consists of two stages: the data publication stage and the revocation operation stage. In the data publication stage, DO first splits the original data F into n slices via a special (n, n) threshold scheme. Second, DO selects a random slice as the dynamic data DF, and binds the remaining slices together as the static data SF. The dynamic data DF are then over encrypted by a key-based encryption algorithm such as the Advanced Encryption Standard (AES), and the key k is encrypted by the CP-ABE algorithm. Both the ciphertexts $E_k(\mathrm{DF})$ and $E'_T(k)$ will be published to the cloud storage. Finally, the static data are published to the cloud storage directly without extra encryption.

The second stage explains how to improve the performance of the revocation procedure. When a revocation takes place, it will be handled in the following steps: firstly, DO retrieves $E_k(\mathrm{DF})$ and $E'_T(k)$

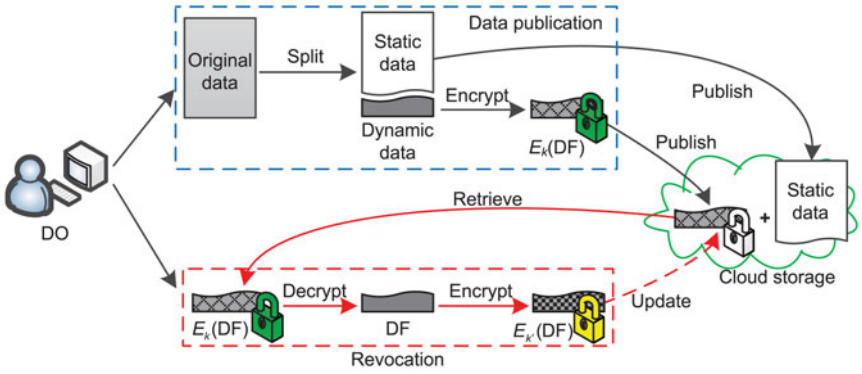

Fig. 2 Revocation optimization model

from the cloud storage and decrypts them to obtain DF; secondly, DO chooses another random key k' and encrypts DF with k', and then encrypts k' by the CP-ABE algorithm; finally, the new ciphertexts $E'_k(\mathrm{DF})$ and $E'_{T'}(k')$ are published to the cloud storage for replacing the expired ones. During the whole process, the static data are not affected by the revocation operation.

The revocation optimization model performs the revocation operation only on the dynamic data instead of the entire data, so its overhead will be about $1/n$ of the complete scheme's. However, the (n, n) threshold scheme usually consumes more time and CPU resources than the symmetric encryption algorithm, and thus this model will increase the overhead of data publication. Intuitively, this model is very beneficial when the revocation occurs frequently.

In the above model, all the revocation operations are performed on the same slice after it is selected as the dynamic data. This restriction may lead to a security issue: if an authorized DU stores a copy of the dynamic data in its local storage, then the DO will be unable to revoke this DU's permission. To address this problem, DO needs to select a slice randomly as the dynamic data for every revocation (The case for which the entire data are copied by the DU to its local storage is not considered in this paper, because there is no valid scheme that can revoke DU's permission to access its local files).

3.3 Efficient revocation scheme

According to the revocation optimization model, the efficient revocation scheme is designed and implemented in the CP-ABE based cryptographic cloud storage. This scheme is described by a series of algorithms, including data splitting, data publishing, data retrieving, permission granting, and permission revoking.

The data splitting algorithm employed in this scheme is an enriched Robin (1989)'s information dispersal algorithm (IDA). We will use a special case of the original IDA where $m = n$. The core of this IDA is a matrix-vector product (Fig. 3). The original data F are broken into n blocks with the same length. A generator matrix $\boldsymbol{G}$ is created with n rows and n columns. The matrix $\boldsymbol{G}$ is multiplied by the n-element vector $\boldsymbol{F}$ to yield an n-element vector $\boldsymbol{S}$. Each element of $\boldsymbol{S}$ is a result slice. The data F can be reconstructed by multiplying $\boldsymbol{S}$ with the inverted matrix $\boldsymbol{G}^{-1}$. More details about the dispersal algorithm can be found in Rabin (1989) and Resch and Plank (2011).

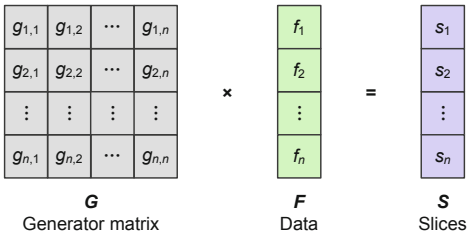

Fig. 3 Core of an (n, n) information dispersal algorithm

In the data splitting algorithm, the original data F are first processed by a variant of the AONT scheme, and then the result data are split into n slices by the (n, n) IDA. Suppose that the original data consist of t words, each of which has w bits in length. Furthermore, there is a publicly known key-based encryption algorithm E in this scheme. The

data splitting algorithm is presented as Algorithm 1.

Algorithm 1 Data splitting algorithm

Input: the original data $m_1, m_2, \cdots, m_t$, the number of slices n

Output: the key K_1, the shares $s_1, s_2, \cdots, s_n$

1: let m_{t+1} be the hash of the original data: $m_{t+1} = \text{hash}(m_1, m_2, \cdots, m_t)$;
2: choose a random key K temporarily for the key-based encryption algorithm E;
3: compute the shares $c_1, c_2, \cdots, c_t, c_{t+1}$ by letting $c_i = m_i \oplus E_K(i)$, where $1 \le i \le t+1$;
4: compute the key $K_1 = K \oplus h_1 \oplus h_2 \oplus \cdots \oplus h_{t+1}$, where $h_i = \text{hash}(c_i)$;
5: package $c_1, c_2, \cdots, c_t, c_{t+1}$ together as input to an (n, n) IDA;
6: run the (n, n) IDA and generate n shares $s_1, s_2, \cdots, s_n$;

This algorithm is served as a variant of the (n, n) threshold scheme. The original data with t words are encoded into $t+1$ words, where the word c_{t+1} is used for checking the integrity of the reconstructed data. Different from AONT, the reconstruction procedure requires not only all the words but also the key K_1. The key K_1 will be encrypted by a CP-ABE algorithm in the further process.

It is easy to reconstruct the data from $s_1, s_2, \cdots, s_n$ and K_1. This reconstruction algorithm is presented as Algorithm 2.

Algorithm 2 Data reconstructing algorithm

Input: the key K_1, the shares $s_1, s_2, \cdots, s_n$

Output: the original data $m_1, m_2, \cdots, m_t$

1: take $s_1, s_2, \cdots, s_n$ as input and run the (n, n) IDA to reconstruct the packaged data;
2: un-package the IDA's result to obtain $t+1$ words: $c_1, c_2, \cdots, c_t, c_{t+1}$;
3: compute the key $K = K_1 \oplus h_1 \oplus h_2 \oplus \cdots \oplus h_{t+1}$, where $h_i = \text{hash}(c_i)$;
4: compute the blocks $m_1, m_2, \cdots, m_t, m_{t+1}$ by letting $m_i = c_i \oplus E_K(i)$, where $1 \le i \le t+1$;
5: let m'_{t+1} be the hash of the other reconstructed data: $m'_{t+1} = \text{hash}(m_1, m_2, \cdots, m_t)$;
6: **if** m'_{t+1} is not equal to m_{t+1} **then**
7: data reconstruction failure;
8: **end if**

After data splitting, the shares will be published to the cloud storage. As discussed in the revocation optimization model, the data publishing procedure is described in Algorithm 3.

Algorithm 3 Data publishing algorithm

Input: the original data $m_1, m_2, \cdots, m_t$, the number of slices n

Output: the publication data's uniform resource locators URLs

1: run Algorithm 1 to split the original data into $s_1, s_2, \cdots, s_n$, and obtain K_1;
2: choose a random slice s_i, where $1 \le i \le n$;
3: select a random key K_2 for the key-based encryption algorithm E;
4: compute $E_{K_2}(s_i)$;
5: compute $E'_T(K_1 + K_2)$, where E' is a CP-ABE algorithm and T is the access structure;
6: publish $E'_T(K_1 + K_2)$, $E_{K_2}(s_i)$, and $s_1, s_2, \cdots, s_n$ to the cloud storage;
7: return the URLs of $E'_T(K_1 + K_2)$, $E_{K_2}(s_i)$, and $s_1, s_2, \cdots, s_n$;

Since the published data about F have already been encrypted, the uniform resource locators (URLs) are no longer sensitive; thus, they can be delivered though unsafe channels. Every DU may be able to get the published data, but only the authorized DU can decrypt the ciphertexts. The procedure of retrieving the original data is presented as Algorithm 4.

Algorithm 4 Data retrieving algorithm

Input: DU's private key SK, URLs

Output: the original data $m_1, m_2, \cdots, m_t$

1: obtain $E'_T(K_1 + K_2)$, $E_{K_2}(s_i)$, and $s_1, s_2, \cdots, s_n$ by the URLs;
2: **if** S does not satisfy T **then**
3: failure and abort;
4: **end if**
5: decrypt $E'_T(K_1 + K_2)$ to obtain K_1 and K_2;
6: decrypt $E_{K_2}(s_i)$ to obtain s_i;
7: take K_1, $s_1, s_2, \cdots, s_n$ as input, and call Algorithm 2 to reconstruct the original data;

After the CP-ABE based cryptographic cloud storage is started up, DO may need to modify the access policy, such as granting a DU u_i to access the object o_j, or revoking u_j's permission to access o_i. The permission granting operation can be completed simply by updating the ciphertext $E'_T(K + K_1)$. Thus, DO needs only to re-encrypt K and K_1 with the new access structure T' and re-publish the new ciphertext.

The permission revoking procedure is more complicated, and it is presented as Algorithm 5. In the revocation algorithm, DO will choose a random

slice as the dynamic data for each revocation operation. Therefore, caching the dynamic data in its local storage cannot enable a DU to attack the system.

Algorithm 5 Revocation algorithm

Input: F's new access policy T', URLs

retrieve $E'_T(K_1+K_2)$, $E_{K_2}(s_i)$ by the URLs;
decrypt $E'_T(K_1+K_2)$ to obtain K_1 and K_2;
decrypt $E_{K_2}(s_i)$ to obtain s_i;
select a random key K temporarily for the key-based encryption algorithm E;
let $K_2=K$ and compute $E'_{T'}(K_1+K_2)$;
select a random index from $[1,2,\cdots,n]$;
if $i\neq j$ **then**
    retrieve the share s_j by the URLs;
    compute $E_{K_2}(s_j)$;
    replace $E'_T(K_1+K_2)$, $E_{K_2}(s_i)$, s_j in cloud with $E'_{T'}(K_1+K_2)$, s_i, $E_{K_2}(s_j)$ respectively;
else
    compute $E_{K_2}(s_i)$;
    replace $E'_T(K_1+K_2)$, $E_{K_2}(s_i)$ in cloud with $E'_{T'}(K_1+K_2)$, $E_{K_2}(s_i)$ respectively;
end if

3.4 Security analysis

In Section 3.3, we have described the efficient revocation scheme by a series of algorithms, and now we begin to analyze its security. The ciphertext $E'_T(K_1+K_2)$ is produced by the CP-ABE algorithm, and it is provable security under the decisional bilinear Diffie-Hellman assumptions (Waters, 2011). The ciphertexts $s_1,s_2,\cdots,s_n$ are produced by the AONT scheme and the attacker cannot get any useful information from them without s_i; therefore, they are computationally secure. The ciphertext $E_{K_2}(s_i)$ stored in the cloud storage has already been encrypted twice: AONT encryption (AONE) and IDA encryption (IDAE).

In the AONE stage, the data are encrypted by a variant of the AONT scheme, and the key K_1 is encrypted by a CP-ABE algorithm later. No one is able to decode the result of AONE without K_1. However, if an attacker owns K_1 and the shares $s_1,s_2,\cdots,s_n$, then the attacker can easily obtain the original data via Algorithm 2. The difficulty of figuring out K_1's value is determined by the AONT and CP-ABE algorithms. Fortunately, the two encoding functions both guarantee that enumeration is the only way to discover the value of K_1. An attacker must test up to $2w'$ (w' is the bit length of K_1) potential values of K_1 to discover its real value, and thus the AONE is computationally secure.

In the IDAE stage, the result data of AONE are broken into n slices, and then one of these slices is encrypted by a key-based encryption algorithm (such as AES). The key K_2 is also encrypted by the CP-ABE algorithm later. If an attacker wants to decode the ciphertext, all the shares $s_1,s_2,\cdots,s_n$ will have to be obtained. Thus, the attacker needs to estimate the key K_2 or the dynamic data s_i ($1\leq i\leq n$) directly. Due to the security of the IDA and key-based encryption algorithm, the attacker has to enumerate all potential values of K_2 or s_i. The complexities of these attacks are $2w_1$ and $2w_2$, where w_1 and w_2 are the bit lengths of K_2 and s_i, respectively. Therefore, the IDAE is also computationally secure.

From the above analysis, we know that all the ciphertexts in the cloud storage are at least computationally secure. Now, we discuss two types of attackers in the CP-ABE based cryptographic cloud storage. One is the untrusted CSP who may leak customers' content actively or passively. The other is the non-authorized or revoked DU who wants to access the unauthorized data.

Conservatively, CSP is assumed to own the ability of storing all data indefinitely. In other words, CSP has the ability to keep the data for an infinite time after it is published to the cloud storage. In the data publishing procedure, F has been encrypted by AONE and IDAE (actually we perform only both AONE and IDAE on dynamic data, but it can be regarded as complete data encryption). Thus, the servers have the locked view of F, which means CSP cannot obtain any useful information about F. If a revocation occurs and DO re-encrypted s_j instead of s_i where $i\neq j$, CSP will have an aone_locked view of F. CSP has already obtained s_j before the revocation, and now s_i is obtained as well. The result is that CSP can perform the (n,n) IDA to reconstruct the data (that is, the output of AONE) without knowing K_2. Fig. 4 shows the possible views of CSP and the transform from locked view to aone_locked one.

We assume that DU will keep the keys (e.g., K_1 and K_2) in its local memory after it obtains them. Fig. 5 illustrates DU's possible views of F. If a DU, u, is granted to access F, u will be able to obtain K_1 and K_2. Thus, u's view of F will transform from locked to open. Once DO revokes u's permission to

access F by updating the dynamic data, u still keeps the key K_1. Thus, u will move from open view to idae_locked view. Consequently, u is still able to reach the open view if it is granted access to F in the future.

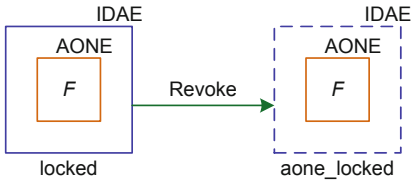

Fig. 4 CSP's possible views of data F

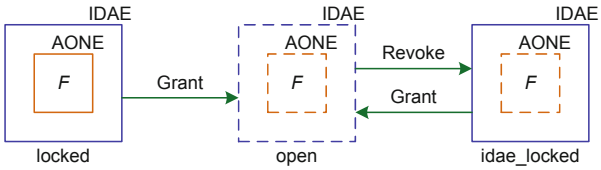

Fig. 5 DU's possible views of data F

From the above analysis, we know that neither CSP nor unauthorized or revoked DU can read the data F. The efficient revocation scheme is computationally secure.

However, if we relax the restriction of CSP and DU, there are still two types of threats that may disclose F. One is the collusion attack that CSP and revoked DU combine their knowledge to access F. Another is the caching attack, which means the revoked DU stores a set of shares in its local storage for guessing the possible dynamic data.

The collusion attack is practicable. This is because, if the attacker combines CSP's aone_locked view and the revoked DU's idae_locked view, there is the possibility of reaching the open view, and thus to F. Generally speaking, no revocation scheme can resist such a collusion attack, unless DO re-encrypts the whole data and deletes the old version in the cloud storage completely. Thus, the efficient revocation scheme has already reached the approximate maximum security in front of this collusion attack.

The caching attack is performed by the revoked DU. If the revoked DU keeps m $(0 \leq m < n)$ shares in its local storage, it will have the m/n probability to break up the IDAE. This attack cannot be resisted unless DO re-encrypts the whole data F with a new K_1. However, the attacker's advantage can be reduced by increasing n and encrypting a set of shares instead of only one share. For example, supposing that the size of F is 100 MB and the size of the dynamic data is 10 MB, the attacker will cache 10 MB data for guessing the correctly dynamic data. If $n = 10$ and DO encodes one share, the attacker's success probability is $1/\mathrm{C}_{10}^{1} = 1/10$. If $n = 20$ and DO encrypts two shares, the attacker has to cache two shares for guessing the right dynamic data, so the success probability is $1/\mathrm{C}_{20}^{2} = 1/190$. Furthermore, the caching attack could be ignored in practical systems. This is because once the revoked DU has the ability to store some shares, it will also be able to keep the original data F (or part of them) in local storage directly.

4 Performance evaluation

4.1 Theoretical analysis

We evaluate the theoretical performance of the efficient revocation scheme via mathematical methods. The symbols used in our analysis are listed in Table 1. For some of these variables, we give the assumed values.

Table 1 Symbols used in performance evaluation

Symbol	Assumed value	Description
n		Number of shares
w	8	Length of the word (bytes)
w_{k}	32	Length of the key (bytes)
N		Size of the original data (words)
B	10	DO's bandwidth (MB/s)
E_{aes}	100	Key-based encryption algorithm's encoding rate (MB/s)
D_{aes}	100	Key-based encryption algorithm's decoding rate (MB/s)
E_{cpabe}	1.5	CP-ABE algorithm's encoding rate (MB/s)
D_{cpabe}	3	CP-ABE algorithm's decoding rate (MB/s)
E_{ida}		IDA's splitting rate (MB/s) (which depends on n)
D_{ida}		IDA's reconstructing rate (MB/s) (which depends on n)
E_{hash}	80	Hash function's encoding rate (MB/s)
E_{xor}	600	Bandwidth of performing $\oplus$ operations (MB/s)

Let T_{split} represent the time that Algorithm 1 takes to split the original data F into n shares. It

can be calculated as follows:

$$T_{\text{split}} = \frac{Nw}{E_{\text{hash}}} + \frac{(N+1)w}{E_{\text{aes}}} + \frac{2(N+1)w}{E_{\text{xor}}} + \frac{(N+1)w}{E_{\text{hash}}} + \frac{(N+1)w}{E_{\text{ida}}}. \quad (1)$$

The time to reconstruct the shares is

$$T_{\text{reconstruct}} = \frac{(N+1)w}{D_{\text{ida}}} + \frac{(N+1)w}{E_{\text{hash}}} + \frac{Nw}{E_{\text{hash}}} + \frac{(N+1)w}{E_{\text{aes}}} + \frac{2(N+1)w}{E_{\text{xor}}}. \quad (2)$$

Using T_{split} and $T_{\text{reconstruct}}$, we can calculate the time of publishing and retrieving data as in the following way:

$$T_{\text{publish}} = T_{\text{split}} + \frac{(N+1)w}{nE_{\text{aes}}} + \frac{2w_{\text{k}}}{E_{\text{cpabe}}} + \frac{(N+1)w+2w_{\text{k}}}{B}, \quad (3)$$

$$T_{\text{retrieve}} = \frac{(N+1)w+2w_{\text{k}}}{B} + \frac{2w_{\text{k}}}{D_{\text{cpabe}}} + \frac{(N+1)w}{nD_{\text{aes}}} + T_{\text{reconstruct}}. \quad (4)$$

The time for the revocation operation is calculated as follows:

$$T_{\text{revoke}} = \frac{(N+1)w/n+2w_{\text{k}}}{B} + \frac{2w_{\text{k}}}{D_{\text{cpabe}}} + \frac{(N+1)w}{nD_{\text{aes}}} + \frac{2w_{\text{k}}}{E_{\text{apabe}}} + \frac{1}{n}\left[\frac{(N+1)w}{nE_{\text{aes}}} + \frac{(N+1)w/n+2w_{\text{k}}}{B}\right] + \frac{n-1}{n}\left[\frac{(N+1)w}{nB} + \frac{(N+1)w}{nE_{\text{aes}}} + \frac{2(N+1)w/n+2w_{\text{k}}}{B}\right]. \quad (5)$$

Notice that Eqs. (1)–(5) are the corresponding time consumption to Algorithms 1–5. The size of $E'_T(K_1+K_2)$ is usually larger than that of K_1+K_2, but we simply let the size of $E'_T(K_1+K_2)$ be $2w_{\text{k}}$ in Eqs. (3)–(5).

In this evaluation, we compare the efficient revocation scheme with the lazy and complete revocation models. The publishing and retrieving times for both the lazy and complete revocation schemes can be calculated as follows:

$$T_{\text{PUB2}} = \frac{Nw}{E_{\text{aes}}} + \frac{w_{\text{k}}}{E_{\text{cpabe}}} + \frac{Nw+w_{\text{k}}}{B}, \quad (6)$$

$$T_{\text{RET2}} = \frac{Nw+w_{\text{k}}}{B} + \frac{w_{\text{k}}}{D_{\text{cpabe}}} + \frac{Nw}{D_{\text{aes}}}. \quad (7)$$

In the lazy revocation scheme, DO does not need to revoke the permission immediately, so the time of revocation is 0. In the complete revocation scheme, the time of revocation is

$$T_{\text{REV2}} = T_{\text{PUB2}} + T_{\text{RET2}}. \quad (8)$$

We assume that the size of the original data F is 100 MB, the number of shares is 10, and IDA's splitting and reconstructing rate is 40 MB/s for each (i.e., N=13 107 200, n=10, E_{ida}= D_{ida}=40). The performance comparison among the efficient, lazy, and complete revocation schemes is shown in Fig. 6.

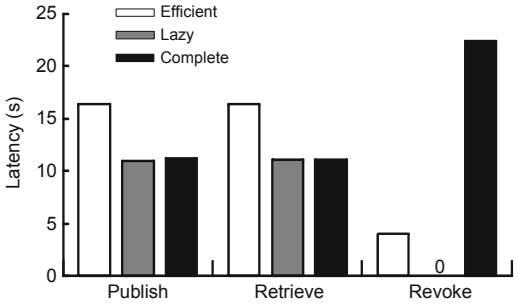

Fig. 6 Performance comparison among the efficient, lazy and complete revocation schemes

This comparison shows that the efficient revocation scheme has achieved a significant improvement over the complete one (It is of no sense to compare the efficient revocation scheme with the lazy one, because the lazy revocation scheme just bypasses the revocation operation). However, the efficient revocation scheme consumes more time for publishing and retrieving data than both the lazy and complete schemes.

The purpose of adopting the efficient revocation scheme is to reduce DO's workload. Supposing DO needs to enforce x revocation operations on data F after publishing it to cloud, the amount of time that DO costs is

$$T_{\text{manage}} = T_{\text{publish}} + xT_{\text{revoke}}. \quad (9)$$

Fig. 7 presents how DO's T_{manage} increases with x under the efficient, lazy, and complete schemes.

This figure illustrates that the efficient revocation scheme is beneficial if and only if the revocation operations occur frequently. In other words, there is a constant threshold x_0 (x_0 is an integer) such that the efficient revocation scheme will perform better than the complete one if $x \geq x_0$. In this assumed scenario, $x_0 = 1$.

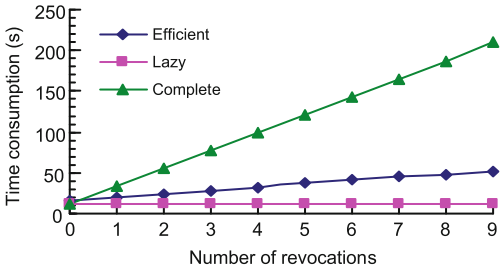

Fig. 7 Amount of time consumed by DO with different x's

4.2 Measured performance

We measure the experimental performance of the efficient revocation scheme in a CP-ABE based cryptographic access control prototype. This prototype is a C++ program running in Linux platform. To implement the cryptographic components, we have brought in some existing works in the prototype. For example, we use OpenSSL (2012)'s Crypto library to implement the AES-192 and SHA256 algorithms, employ the CP-ABE toolkit (Bethencourt *et al.*, 2012) to implement the CP-ABE scheme, and modify the Jerasure library's routines (Plank *et al.*, 2008) to implement the IDA component.

The prototype is composed of two programs: the manager and the client. DO runs the manager for publishing data and revoking access policies, and DU runs the client for retrieving the authorized data. The CP-ABE based cryptographic access control is also enforced in the manager's program. Both the manager and client programs are implemented in single threads for measuring our scheme's absolute time consumption. These programs are run on Ubuntu desktop version 10.04 with the kernel version 2.6.32. We use a local Hadoop Distributed File System (HDFS) (Hadoop, 2012) as the storage servers in our system. The version of Hadoop is 0.23.1. The servers' platform is Ununtu server version 10.04.

Each manager or client machine has a 4-core Intel® Core™ Q6600 processor running at 2.40 GHz with 8 MB cache and 2 GB RAM. The storage server consists of three workstations, each having a 4-core Intel® Xeon® E5506 processor at 2.13 GHz with 4 MB cache and 4 GB ECC RAM, and there are two 1-TB Seagate SATA drives for storage. Each experimental machine has a 1000-Mbps network interface card, and is connected by a 100-Mbps Ethernet switch.

We first evaluate the storage overhead of our scheme. Since the sync encryption and dispersal algorithm will not increase data size, we need only to consider the additional storage for storing encrypting keys. The size of the symmetric key (e.g., k) is 192 bits. Each file is attached with a CP-ABE ciphertext $E'_T(k)$, and the size of the ciphertext increases with the size of T. For a typical access tree with hundreds of nodes, the size of the ciphertext is about tens of KB. Compared to the data size (MB), the storage overhead can be ignored.

We measure the performance of publishing data, retrieving data, and revoking permission under different revocation schemes (including the efficient, lazy, and complete revocation schemes). The size of the data changes from 25 to 800 MB. The size of shares is a constant equal to 5 MB; as a result, the number of shares changes with N. The measured results are presented in Fig. 8. The measured results of data publishing and retrieving operations show that the latency of the efficient revocation scheme is about 1.8 times that of the lazy and complete ones. The revoking latency of the efficient revocation scheme is steadily about 2 s, which is much less than that of the complete scheme.

We also measure the revoking latency of different revocation operations of a fixed-size file by varying the number of shares. The size of the file is 1 GB, and the number of shares changes from 10 to 200. The measured results are shown in Fig. 9, where the revoking latency of the complete revocation scheme is also presented for comparison. These results show that DO can reduce the revoking latency of the efficient revocation scheme by increasing the number of shares.

The threshold x_0 in all the above test cases is 1, which means the efficient revocation scheme will achieve a better performance than does the complete scheme when a revocation occurs. If x_0 is a real number, we can calculate the average value, which is 0.462. Therefore, DO will benefit from the efficient revocation scheme if the probability of performing

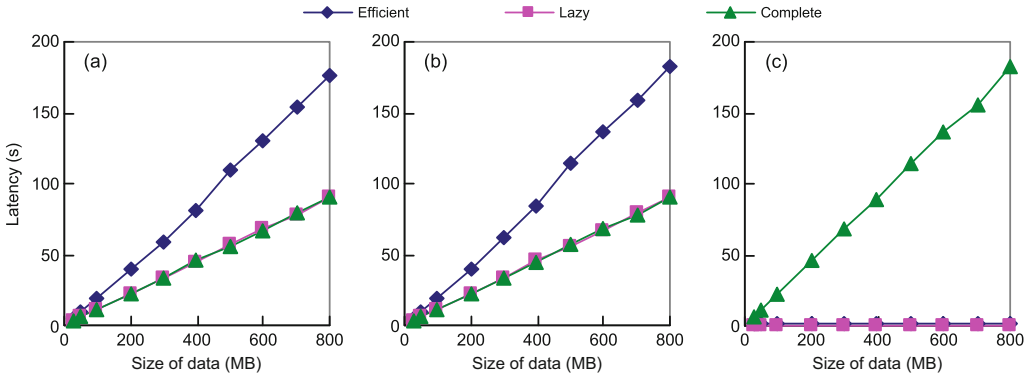

Fig. 8 Measured performances of data publishing (a), retrieving (b), and permission revoking (c)

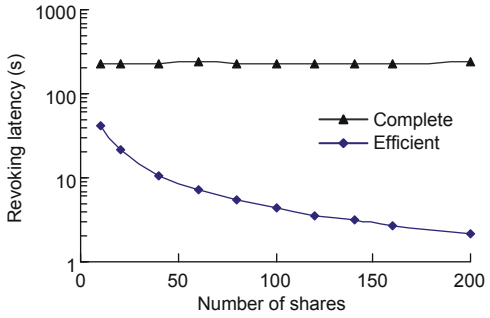

Fig. 9 Revoking latency of the efficient revocation scheme under different numbers of shares

revocations is more than 46.2%. Generally speaking, different cryptographic cloud storages have different thresholds, and DO needs to measure such a threshold before making the decision to adopt the efficient revocation scheme.

5 Conclusions

We presented the first revocation scheme which is efficient, secure, and unassisted. The scheme can promote the usage of cryptographic cloud storage by reducing DO's workload. We described how to apply the efficient scheme to a CP-ABE based cryptographic access control system, and evaluated its security and performance. The evaluated results show that the efficient revocation scheme is computationally secure, and its revoking performance is better than that of the complete scheme if the revocation occurs frequently. Measured performance shows that DO will benefit from the efficient revocation scheme if the average number of revocations is greater than 0.462. In summary, the efficient revocation scheme provides an optimal tradeoff in terms of security and efficiency.

Acknowledgements

We express our thanks to Wang-yi HAN who give advice on the writing of the manuscript.

References

Amazon, 2012. Amazon Simple Storage Service. Available from http://aws.amazon.com/s3/ [Accessed on June 11, 2012].

Backes, M., Cachin, C., Oprea, A., 2005. Lazy Revocation in Cryptographic File Systems. Proc. 3rd IEEE Int. Security in Storage Workshop, p.1-11. [doi:10.1109/SISW.2005.7]

Bethencourt, J., Sahai, A., Waters, B., 2007. Ciphertext-Policy Attribute-Based Encryption. IEEE Symp. on Security and Privacy, p.321-334. [doi:10.1109/SP.2007.11]

Bethencourt, J., Sahai, A., Waters, B., 2012. Ciphertext-Policy Attribute-Based Encryption. Available from http://acsc.cs.utexas.edu/cpabe/ [Accessed on June 10, 2012].

Blanchet, B., Chaudhuri, A., 2008. Automated Formal Analysis of a Protocol for Secure File Sharing on Untrusted Storage. IEEE Symp. on Security and Privacy, p.417-431. [doi:10.1109/SP.2008.12]

Blaze, M., Bleumer, G., Strauss, M., 1998. Divertible protocols and atomic proxy cryptography. *LNCS*, **1403**:127-144. [doi:10.1007/BFb0054122]

Boneh, D., Gentry, C., Waters, B., 2005. Collusion resistant broadcast encryption with short ciphertexts and private keys. *LNCS*, **3621**:258-275. [doi:10.1007/11535218_16]

di Vimercati, S.D.C., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P., 2007. Over-Encryption: Management of Access Control Evolution on Outsourced Data. Proc. 33rd Int. Conf. on Very Large Data Bases, p.123-134.

Foresti, S., 2010. Preserving Privacy in Data Outsourcing. Springer. [doi:10.1007/978-1-4419-7659-8]

Geambasu, R., Kohno, T., Levy, A., Levy, H.M., 2009. Vanish: Increasing Data Privacy with Self-Destructing Data. Proc. 18th USENIX Security Symp., p.299-333.

Goyal, V., Pandey, O., Sahai, A., Waters, B., 2006. Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data. Proc. 13th ACM Conf. on Computer and Communications Security, p.89-98. [doi:10.1145/1180405.1180418]

Hadoop, 2012. HDFS Architecture Guide. Available from http://hadoop.apache.org/docs/hdfs/current/hdfs_design.html [Accessed on June 28, 2012].

Jahid, S., Mittal, P., Borisov, N., 2011. Easier: Encryption-Based Access Control in Social Networks with Efficient Revocation. Proc. 6th ACM Symp. on Information, Computer and Communications Security, p.411-415. [doi:10.1145/1966913.1966970]

Kamara, S., Lauter, K., 2010. Cryptographic Cloud Storage. Proc. 14th Int. Conf. on Financial Cryptography and Data Security, p.136-149. [doi:10.1007/978-3-642-14992-4_13]

Kumbhare, A.G., Simmhan, Y., Prasanna, V., 2011. Designing a Secure Storage Repository for Sharing Scientific Datasets Using Public Clouds. Proc. 2nd Int. Workshop on Data Intensive Computing in the Clouds, p.31-40. [doi:10.1145/2087522.2087530]

Lewko, A., Sahai, A., Waters, B., 2010. Revocation Systems with Very Small Private Keys. IEEE Symp. on Security and Privacy, p.273-285. [doi:10.1109/SP.2010.23]

Liang, X., Cao, Z., Lin, H., Shao, J., 2009. Attribute Based Proxy Re-encryption with Delegating Capabilities. Proc. 4th Int. Symp. on Information, Computer, and Communications Security, p.276-286. [doi:10.1145/1533057.1533094]

Libert, B., Vergnaud, D., 2011. Unidirectional chosen-ciphertext secure proxy re-encryption. *IEEE Trans. Inf. Theory*, **57**(3):1786-1802. [doi:10.1007/978-3-540-78440-1_21]

OpenSSL, 2012. OpenSSL: Cryptography and SSL/TLS Toolkit. Available from http://www.openssl.org/ [Accessed on July 26, 2012].

Plank, J.S., Simmerman, S., Schuman, C.D., 2008. Jerasure: a Library in C/C++ Facilitating Erasure Coding for Storage Applications, Version 1.2. Available from http://web.eecs.utk.edu/~plank/plank/papers/CS-08-627.html [Accessed on June 29, 2012].

Rabin, M.O., 1989. Efficient dispersal of information for security, load balancing, and fault tolerance. *J. ACM*, **36**(2):335-348. [doi:10.1145/62044.62050]

Resch, J.K., Plank, J.S., 2011. AONT-RS: Blending Security and Performance in Dispersed Storage Systems. Proc. 9th Usenix Conf. on File and Storage Technologies, p.191-202.

Rivest, R., 1997. All-or-nothing encryption and the package transform. *LNCS*, **1267**:210-218. [doi:10.1007/BFb0052348]

Sahai, A., Waters, B., 2005. Fuzzy identity-based encryption. *LNCS*, **3494**:557-557. [doi:10.1007/11426639_27]

Samarati, P., di Vimercati, S.D.C., 2010. Data Protection in Outsourcing Scenarios: Issues and Directions. Proc. 5th ACM Symp. on Information, Computer and Communications Security, p.1-14. [doi:10.1145/1755688.1755690]

Shamir, A., 1979. How to share a secret. *Commun. ACM*, **22**(11):612-613. [doi:10.1145/359168.359176]

Storer, M.W., Greenan, K.M., Miller, E.L., Voruganti, K., 2007. Potshards: Secure Long-Term Storage without Encryption. Proc. USENIX Annual Technical Conf., p.1-11.

Tang, Y., Lee, P., Lui, J., Perlman, R., 2012. Secure overlay cloud storage with access control and assured deletion. *IEEE Trans. Depend. Sec. Comput.*, **9**(6):903-916. [doi:10.1109/TDSC.2012.49]

Waters, B., 2011. Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. *LNCS*, **6571**:53-70. [doi:10.1007/978-3-642-19379-8_4]

Xu, L., Wu, X., Zhang, X., 2012. CL-PRE: a Certificateless Proxy Re-encryption Scheme for Secure Data Sharing with Public Cloud. Proc. 7th ACM Symp. on Information, Computer and Communications Security, p.1-10.

Xu, Z., Martin, K.M., 2012. Dynamic User Revocation and Key Refreshing for Attribute-Based Encryption in Cloud Storage. Proc. 11th Int. Conf. on Trust, Security and Privacy in Computing and Communications, p.844-849. [doi:10.1109/TrustCom.2012.136]

Yu, S., Wang, C., Ren, K., Lou, W., 2010a. Achieving Secure, Scalable, and Fine-Grained Data Access Control in Cloud Computing. IEEE INFOCOM, p.1-9. [doi:10.1109/INFCOM.2010.5462174]

Yu, S., Wang, C., Ren, K., Lou, W., 2010b. Attribute Based Data Sharing with Attribute Revocation. Proc. 5th ACM Symp. on Information, Computer and Communications Security, p.261-270. [doi:10.1145/1755688.1755720]