Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 329626, 8 pages
http://dx.doi.org/10.1155/2015/329626

*Research Article*

# A Secure Ciphertext Self-Destruction Scheme with Attribute-Based Encryption

**Tonghao Yang, Junquan Li, and Bin Yu**

*Zhengzhou Institute of Information Science and Technology, Zhengzhou 450000, China*

Correspondence should be addressed to Tonghao Yang; youngtonghao@163.com

Received 17 June 2015; Revised 29 September 2015; Accepted 5 October 2015

Academic Editor: Mark Leeson



The secure destruction of expired data is one of the important contents in the research of cloud storage security. Applying the attribute-based encryption (ABE) and the distributed hash table (DHT) technology to the process of data destruction, we propose a secure ciphertext self-destruction scheme with attribute-based encryption called SCSD. In SCSD scheme, the sensitive data is first encrypted under an access key and then the ciphertext shares are stored in the DHT network along with the attribute shares. Meanwhile, the rest of the sensitive data ciphertext and the shares of access key ciphertext constitute the encapsulated self-destruction object (EDO), which is stored in the cloud. When the sensitive data is expired, the nodes in DHT networks can automatically discard the ciphertext shares and the attribute shares, which can make the ciphertext and the access key unrecoverable. Thus, we realize secure ciphertext self-destruction. Compared with the current schemes, our SCSD scheme not only can support efficient data encryption and fine-grained access control in lifetime and secure self-destruction after expiry, but also can resist the traditional cryptanalysis attack as well as the Sybil attack in the DHT network.

## 1. Introduction

Cloud storage has attracted much attention from both industry and academia for its low cost, flexible deployment, and strong extensibility in recent years. The cloud storage system is composed of massive storage resource on the Internet as well as the resource management and access control mechanism for the resource accessing transparency of users [1]. With friendly user interface and strong extensibility, the cloud storage system can provide users with unlimited storing space; thus, it can form a new delivery model called storage as a service [2]. Cloud storage brings new opportunities for efficiency increasing, cost saving, and green computing in the area of information technology; however, it is also faced with some security challenges.

In the service model of cloud storage, data is outsourced to the storage server which performs as the third party. So, data is out of the control of data owner and the security of data highly depends on the server. Due to the dishonesty of cloud storage server, the data owner will first encrypt the original sensitive data and then outsource the ciphertext to the cloud in order to keep the confidentiality of data. The encryption key is kept by the data owner privately. However, even if the data is stored by cloud in the form of ciphertext, there are some security risks. For example, in order to improve the service reliability, the cloud may make several backups for the user's data and distribute them to different storage servers [3]. On this condition, when the data has expired and the owner needs to delete the data from the storage servers, the cloud server may not destruct all the backups of data. Once adversaries get the encryption key and the backups of the ciphertext from cloud, the sensitive data can be recovered and the confidentiality is destroyed. Therefore, the assured destruction of expired data, namely, the thorough deletion and the permanent elimination of ciphertext, is one of the important contents in the research of cloud storage security [4].

In this paper, applying the attribute-based encryption and the distributed hash table (DHT) technology to the process of data destruction in the cloud storage environment, we propose a secure ciphertext self-destruction scheme with

attribute-based encryption called SCSD. In SCSD scheme, the sensitive data is first encrypted under an access key, and then the access key is encrypted using an attribute-based encryption method. The ciphertext of sensitive data is extracted and transformed in order to get the ciphertext shares, which are stored in the DHT network along with the attribute shares. Meanwhile, the rest of the sensitive data ciphertext and the shares of access key ciphertext constitute the encapsulated self-destruction object (EDO), which is stored in the cloud. When the sensitive data is expired, the nodes in DHT networks can automatically discard the ciphertext shares and the attribute shares, which can make the ciphertext of sensitive data and the access key unrecoverable. Thus, we realize secure ciphertext self-destruction. Compared with the current schemes, our SCSD scheme can resist the traditional cryptanalysis attack as well as the Sybil attack in the DHT network.

The rest of the paper is organized as follows. In Section 2, we introduce some related works of the secure data destruction. Then, in Section 3, we review some preliminaries. Next, we introduce the system and security model and the detailed construction of our SCSD scheme in Section 4. In Section 5, we make an evaluation for the scheme in security analysis and scheme performance. Finally, concluding remarks and future work are given in Section 6.

## 2. Related Works

In cloud storage system, some data is stored in the servers for a long time, which can be compromised by adversaries, because the data may be backed up by the cloud servers and these backups may still exist after the delete command of users. It is difficult to destruct all the backups in the cloud, and the following works are some attempts to achieve the secure destruction of data.

Perlman is the first to focus on the secure deletion of documents [7]. Perlman designed an unrecoverable system for documents. The encryption key is deleted when it is expired; thus, the document encrypted under this key can not be recovered. However, this system considers only the lifetime of encryption key. Besides, this is a local-centered system and is unfit for the cloud environment. Then, following this idea, FADE [8], one secure overlap cloud storage system built under the existing cloud infrastructure, is developed. This system can assure the deletion of documents and can support different document access policies. Another feasible system is Ephemerizer [9], which needs a trusted server to store and manage the decryption key. In Ephemerizer, the data owner sets the expired time for the decryption key. The trusted server deletes the decryption key once the key is expired. Thus, the ciphertext is unreadable.

The above methods follow the idea of centralized solution, which has some limitations as follows. (1) The key management depends too much on the server. (2) When there is an investigation from government, the administrator needs to give up the right of key management. This condition makes the server no longer trusted. (3) There is a need for additional commands and operations to achieve the assured deletion of data.

In order to solve the problem brought by the centralized destruction scheme, Geambasu et al. propose an interesting data self-destruction system called Vanish [5]. The private data is encrypted under a symmetric key, which is divided into several key shares using threshold secret sharing scheme and then distributed to a large scale DHT P2P network. The nodes in the DHT network will automatically delete the key shares periodically, which will result in the unreadable ciphertext. Thus, it realizes the self-destruction of data and this needs trusted servers or additional operations. Wang et al. improve the Vanish system by extracting and distributing parts of ciphertext to the DHT network [6]. This improvement will resist the traditional cryptanalysis attack and brute-force attack more efficiently.

However, [10] points out that there are Sybil attacks against the Vuze DHT network adopted by Vanish system. Adversaries can get enough key shares to reconstruct the key before the ciphertext is expired. Thus, there are security problems in the schemes of [5, 6]. Besides, these decentralized solutions adopt the symmetric encryption algorithms, which will bring complex key management and distribution problems. To solve these problems, an improved system called SafeVanish is proposed [11]. RSA algorithm is adopted to firstly encrypt the symmetric key in order to resist the Sybil attack. But this system can not support fine-grained access control mechanism. Applying attribute-based encryption algorithm, Xiong et al. [12] firstly propose a secure self-destruction scheme, which can support fine-grained access control on documents. However, the direct adoption of attribute-based encryption algorithm on documents is not efficient.

Therefore, a secure sensitive data self-destruction scheme, which supports efficient data encryption and key management, fine-grained access control in lifetime and secure self-destruction after expiry, and traditional cryptanalysis attack and Sybil attack resistance, is needed in the cloud storage environment.

## 3. Preliminaries

*3.1. Distributed Hash Table.* Distributed hash table (DHT) [13] supports a distributed database storage model. And DHT network is comprised of large-scaled distributed infrastructures in the P2P networks which support the query, storage, retrieval, and management of data without servers. Every node in the DHT network is responsible for a small-scaled routing and can store parts of data. Thus, the whole DHT network realizes an addressing and storing of data. There are many DHT networks in Internet, such as Vuze, Chord, OpenDHT, and Pastry.

The index of every document stored in the DHT network can be expressed as a pair of $(K, V)$. $K$ is denoted as the hash value of name or other descriptive pieces of information of the document; $V$ can be denoted as the IP address or other descriptive pieces of information of the node that stored the document in DHT network. All of the index items compose a large document index hash table. When $K$ is specified, the location of document can be assured through the corresponding relationship.

Every DHT network has the following three important characteristics, which is suitable for constructing data self-destruction scheme in cloud storage environment:

(1) Data availability: DHT network can provide reliable distributed storage capacity, which assures the availability of the data stored in the nodes of DHT network in the lifetime. This is the foundation of constructing data self-destruction scheme.

(2) Automatic data deletion in the nodes in DHT network: nodes in DHT network can automatically remove the old data in order to store the new data periodically. Thus, the data stored in the nodes will be destroyed automatically after expiry, which provides a mechanism for ciphertext self-destruction.

(3) Large-scaled and global distribution: for example, there are more than one million of active nodes in Vuze network simultaneously, and these nodes are distributed to more than 190 countries all over the world. These completely distributed nodes in DHT network can provide attack resistance capability for self-destruction scheme.

*3.2. Attribute-Based Encryption.* Attribute-based encryption (ABE), a typical public key cryptography, was firstly proposed by Sahai and Waters in 2005 [14]. In an ABE scheme, the identifier for a user is a set of descriptive attributes rather than a string of characters in identity-based encryption (IBE). Every attribute can be mapped to an element in $\mathbb{Z}_p^*$ using a hash function. The ciphertext and user's key are both associated with the attributes. ABE can support threshold policy of attributes. Namely, if and only if the number of same attributes in both sets of attributes $\omega$ and $\omega^*$ is greater than or equal to a certain threshold value, a user with a set of attributes $\omega$ can decrypt the ciphertext successfully which is encrypted under a set of attributes $\omega^*$.

Specifically, an authority firstly defines a threshold value $k$ and generates the system public key, the length of which is related to the number of attributes in $\omega^*$. Then, the authority generates the private key for user with a set of attributes $\omega$. $\omega$ is associated with a random $k-1$ order polynomial $q(x)$. In a decryption process, if $\omega \cap \omega^* \geq k$, then the user chooses random $k$ attributes in the set $\omega \cap \omega^*$ and reconstructs the encryption key through Lagrange's interpolation on the associated polynomial $q(x)$. Thus, the user can decrypt the ciphertext and get the plaintext.

*3.3. Threshold Secret Sharing.* Threshold secret sharing scheme was first proposed by Shamir [15]. The main idea is to divide the secret data into $n$ shares and then distribute these shares to $n$ users. If there is $k$ or more than $k$ shares are extracted from these users, then the secret data can be generated. Otherwise, the secret data can not be generated. This method is called $(k, n)$ threshold secret sharing.

Generally, threshold secret sharing scheme can be achieved by using Lagrange's interpolation polynomial. If there is an interpolation polynomial

$$Q(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_2x^2 + a_1x^1 + a_0 \quad (1)$$

and there are $n$ different points $(x_0, y_0), \ldots, (x_i, y_i), \ldots, (x_{n-1}, y_{n-1})$ that satisfy the equation $Q(x) = y$, then $Q(x)$ is called Lagrange's polynomial, which is composed of the following basic polynomial $Q(x) = \sum_{j=0}^{n-1} y_j q_j(x)$, where $q_j(x) = \prod_{0 \leq i \leq n-1, i \neq j}((x - x_i)/(x_j - x_i))$.

Namely, given $n$ different points satisfying $Q(x) = y$, we can reconstruct a unique $n-1$ order polynomial $Q(x)$.

## 4. SCSD Scheme Construction

In this section, we first describe the system model of the secure ciphertext self-destruction (SCSD) scheme. Then, the detailed algorithm descriptions and the outline of scheme are introduced as follows.

*4.1. System Model.* The SCSD system comprises six different entities: authority, cloud storage servers, DHT network, data owners, data consumers, and adversaries, as shown in Figure 1.

*Authority*. Authority provides the system with security parameters setup and key generation processes. Besides, it also assigns attributes for each user.

*Cloud Storage Servers*. Cloud storage servers are responsible for storing the data sent by the users and assuring that only authenticated users can get access to the data.

*DHT Network*. Nodes in the DHT network are responsible for storing the ciphertext shares and the attribute shares and can automatically discard the stored data.

*Data Owners*. A data owner generates sensitive data and then encrypts it under a random access key. Ciphertext shares are sent by data owner to the DHT network along with the attribute shares. Besides, EDO is sent to cloud by data owner.

*Data Consumers*. The data consumer downloads ciphertext shares and attribute shares from the DHT network and EDO from the cloud. Then, he can decrypt the EDO if his attributes satisfy the ABE threshold policy.

*Adversaries*. Adversaries may try to capture the data in the cloud or in DHT network.

This paper is aiming at preventing the leakage of sensitive data stored in the cloud after expiry. For example, sensitive information in user's historic archive may leak out in the condition of an investigation from government. We assume that the data owner and other authenticated users trust each other. Thus, adversaries may try to compromise the EDO in the cloud after the lifetime of EDO. Or the adversaries may capture the ciphertext shares and the attribute shares stored in DHT network within the lifetime of EDO. So, in the security model of our scheme, we divide the behavior of adversaries into the following two kinds. (1) Adversaries compromise the EDO in the cloud after the lifetime of EDO. The adversary tries to analyze the sensitive data from the EDO. (2) Adversaries compromise the ciphertext shares and the attribute shares stored in DHT network within the

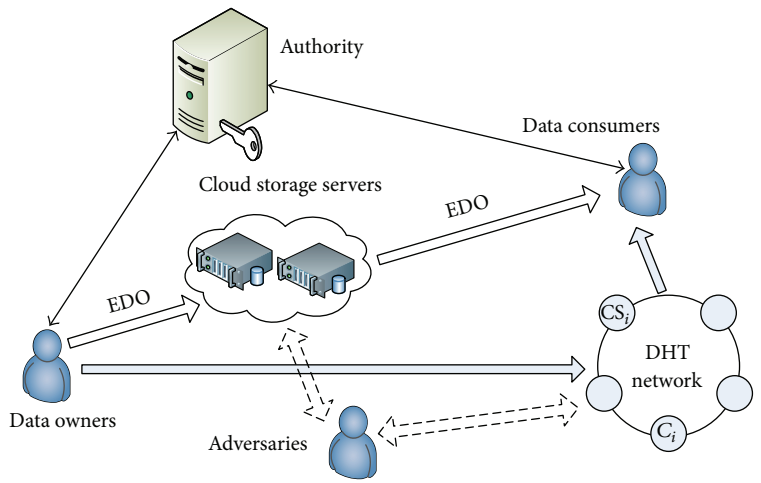


Figure 1: The system model of SCSD scheme.

lifetime of EDO. The adversary tries to decrypt the ciphertext and get the sensitive information according to the shares.

*4.2. Algorithm Descriptions.* Algorithms of our SCSD scheme are described as follows.

(1) Setup($\lambda$) $\rightarrow$ (MSK, PK, USK): given a security parameter $\lambda$, the authority firstly generates the master secret parameters MSK $= (r_1, \ldots, r_n, y)$, which are all chosen randomly from $\mathbb{Z}_p^*$. Then, the authority generates the public parameters PK $=$ (Attri, $g, G_1, G_2, e, k, n, t, s, a, b, H, E$, Dec, $R_1, \ldots, R_n, Y$), where Attri is the set of total $n$ attributes of users and each attribute in Attri is associated with one unique element in $\mathbb{Z}_p^*$. $G_1$ is a multiplicative cyclic group with the generator $g$. $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear map. $k$ is the threshold value for the total $n$ attributes of users. $t$ is the threshold value for the total $s$ ciphertext shares. $a$ is the number of bits in each associated ciphertext extraction. $b$ is the times of extraction, $H : \{0,1\}^* \rightarrow \{0,1\}^m$ is a hash function. $E : (K, M) \rightarrow C$ is a symmetric encryption algorithm and Dec $: (K, C) \rightarrow M$ is the corresponding decryption algorithm. $R_1 = g^{r_1}, \ldots, R_n = g^{r_n}$, $Y = e(g, g)^y$. Besides, the authority also generates secret key for user with attribute set $\text{Attri}_u$. The authority chooses a polynomial $q(x)$ with $k-1$ degree and sets $q(0) = y$. Then, the user's secret key is generated as USK $= (S_i)_{i \in \text{Attri}_u}$, where $S_i = g^{q(i)/r_i}$.

(2) Enc($M$) $\rightarrow$ ($C_M, C_K$): given sensitive data $M$, a data owner with an attribute set $\text{Attri}_o$ firstly chooses a random access key $K \in \mathbb{Z}_p^*$ and generates the ciphertext of $M$ as $C_M = E(K, M)$. Then, the data owner chooses a random value $\gamma \in \mathbb{Z}_p^*$ and generates the ciphertext of $K$ as $C_K =$ ($\text{Attri}_o, C^* = KY^\gamma, \{C_i = R_i^\gamma\}_{i \in \text{Attri}_o}$), where $\{C_i = R_i^\gamma\}_{i \in \text{Attri}_o}$ are the attribute shares.

(3) Associpher($C_M$) $\rightarrow$ ($A$): given a ciphertext $C_M$, the data owner firstly divides the ciphertext into blocks of $m$ bits. If the last block is less than $m$ bits, then several bits of "0" are added to the end until the length of the last block is $m$ bits. Suppose the ciphertext is divided as $A_1 \parallel A_2 \parallel \cdots \parallel A_d$; the data owner associates the blocks as follows:

$$\begin{aligned}
A_1' &= A_1 \oplus H(A_2 \parallel \cdots \parallel A_d) \\
A_2' &= A_2 \oplus H(A_1' \parallel A_3 \parallel \cdots \parallel A_d) \\
&\vdots \\
A_i' &= A_i \oplus H(A_1' \parallel \cdots \parallel A_{i-1}' \parallel A_{i+1} \cdots \parallel A_d) \\
&\vdots \\
A_d' &= A_d \oplus H(A_1' \parallel \cdots \parallel A_i' \parallel \cdots \parallel A_{d-1}').
\end{aligned} \tag{2}$$

Then, the associated ciphertext is $A = A_1' \parallel \cdots \parallel A_i' \parallel \cdots \parallel A_d'$.

(4) DeAssocipher($A$) $\rightarrow$ ($C_M$): this is the inverse algorithm of Associpher($C_M$) $\rightarrow$ ($A$). Given an associated ciphertext $A = A_1' \parallel \cdots \parallel A_i' \parallel \cdots \parallel A_d'$, a data consumer performs as follows:

$$\begin{aligned}
A_d &= A_d' \oplus H(A_1' \parallel \cdots \parallel A_i' \parallel \cdots \parallel A_{d-1}') \\
A_{d-1} &= A_{d-1}' \oplus H(A_1' \parallel \cdots \parallel A_i' \parallel \cdots \parallel A_{d-2}' \parallel A_d) \\
&\vdots \\
A_i &= A_i' \oplus H(A_1' \parallel \cdots \parallel A_{i-1}' \parallel A_{i+1} \cdots \parallel A_d) \\
&\vdots \\
A_1 &= A_1' \oplus H(A_2 \parallel \cdots \parallel A_d).
\end{aligned} \tag{3}$$

Then, the data consumer gets the ciphertext $C_M$ from the association $A_1 \parallel A_2 \parallel \cdots \parallel A_d$.

(5) CipherShareGen$(A) \rightarrow (\mathrm{CS}, C')$: given the associated ciphertext $A$, for $i = 1, 2, \ldots, b$, the data owner firstly extracts the bits located in $[1, a \cdot t]$ in $A^{(i)}$, where $A^{(i)}$ is the remaining associated ciphertext after the $(i-1)$th extraction from $A$. Note that $A^{(1)} = A$. All of the extracted ciphertext is denoted by $\mathrm{EC} = (m_1, m_2, \ldots, m_b)$, where $m_i = m_{[i][0]} \parallel m_{[i][1]} \parallel \cdots \parallel m_{[i][t-1]}$ is the $i$th extracted ciphertext from $A$. The remaining associated ciphertext after the $b$th extraction from $A$ is denoted by $C'$. Then, the data owner generates $b$ polynomials as follows:

$$\begin{aligned} f_1(x) &= m_{[1][t-1]}x^{t-1} + m_{[1][t-2]}x^{t-2} + \cdots + m_{[1][1]}x^1 \\ &\quad + m_{[1][0]} \\ &\vdots \\ f_i(x) &= m_{[i][t-1]}x^{t-1} + m_{[i][t-2]}x^{t-2} + \cdots + m_{[i][1]}x^1 \\ &\quad + m_{[i][0]} \\ &\vdots \\ f_b(x) &= m_{[b][t-1]}x^{t-1} + m_{[b][t-2]}x^{t-2} + \cdots + m_{[b][1]}x^1 \\ &\quad + m_{[b][0]}. \end{aligned} \tag{4}$$

The data owner chooses $s$ different integers $x_1, x_2, \ldots, x_s$ and then computes the value of $f_1(x_i), f_2(x_i), \ldots, f_b(x_i)$ for $i = 1, 2, \ldots, s$. Finally, the data owner gets the ciphertext shares $\mathrm{CS} = (\mathrm{CS}_1, \mathrm{CS}_2, \ldots, \mathrm{CS}_s)$, where $\mathrm{CS}_i = (x_i, f_1(x_i), f_2(x_i), \ldots, f_b(x_i))$ for $i = 1, 2, \ldots, s$.

(6) ShareDistribute$(\mathrm{CS}, C_K) \rightarrow (\mathrm{CI}, \mathrm{AJ})$: given the ciphertext shares CS and the attribute shares $\{C_i = R_i^{\gamma}\}_{i \in \mathrm{Attri}_o}$ from $C_K$, the data owner firstly chooses a random index CI for CS as a seed to a pseudorandom number generator. Then, the data owner runs the generator to generate $s$ indices $I_1, I_2, \ldots, I_s$. For $i = 1, 2, \ldots, s$, each ciphertext share $\mathrm{CS}_i$ is stored in the node indexed by $I_i$ in the DHT network. Similarly, for the attribute shares $\{C_i = R_i^{\gamma}\}_{i \in \mathrm{Attri}_o}$ from $C_K$, the data owner firstly chooses a random index AJ as a seed to a pseudorandom number generator. Then, the data owner runs the generator to generate $n$ indices $J_1, J_2, \ldots, J_n$. For $i = 1, 2, \ldots, n$, each attribute share $C_i$ is stored in the node indexed by $J_i$ in the DHT network.

(7) EDOGen$(\mathrm{Attri}_o, C^*, C', \mathrm{CI}, \mathrm{AJ}) \rightarrow (\mathrm{EDO})$: given the attribute set of the data owner $\mathrm{Attri}_o$, $C^*$ from $C_K$, $C'$, CI, and AJ, the data owner generates the encapsulated self-destruction object $\mathrm{EDO} = (\mathrm{Attri}_o, C^*, C', \mathrm{CI}, \mathrm{AJ})$ and then sends the EDO to the cloud.

(8) KeyRecover$(\mathrm{EDO}, \mathrm{USK}) \rightarrow (K)$: before the expiration timestamp of EDO, a data consumer, with a secret key USK and an attributes set $\mathrm{Attri}_c$, firstly gets the EDO from the cloud. Then, the data consumer runs the pseudorandom number generator to generate $n$ indices $J_1, J_2, \ldots, J_n$ of attribute shares $\{C_i = R_i^{\gamma}\}_{i \in \mathrm{Attri}_o}$ under the seed AJ. Then, the data consumer gets as many $C_i = R_i^{\gamma}$, $i \in \mathrm{Attri}_o$, as possible from the DHT network according to the indices $J_1, J_2, \ldots, J_n$. In order to recover the access key $K$, the data consumer chooses a set of $k$ attribute shares $\mathrm{Att} \in \mathrm{Attri}_o \cap \mathrm{Attri}_c$. Note that if there are no more than $k$ attribute shares in the set of $\mathrm{Attri}_o \cap \mathrm{Attri}_c$, the data consumer can not recover the access key $K$ since he can not satisfy the ABE threshold policy. If there is a set of attribute shares Att, the data consumer firstly gets Lagrange's coefficient $\Delta_{i,\mathrm{Att}}(x) = \prod_{i,j \in \mathrm{Att}, i \neq j}((x-j)/(i-j))$ and then recovers the access key as follows:

$$\begin{aligned} &\frac{C^*}{\prod_{i \in \mathrm{Att}} \left(e\left(S_i, C_i\right)\right)^{\Delta_{i,\mathrm{Att}}(0)}} \\ &= \frac{\mathrm{KY}^{\gamma}}{\prod_{i \in \mathrm{Att}} \left(e\left(g^{q(i)/r_i}, g^{r_i \gamma}\right)\right)^{\Delta_{i,\mathrm{Att}}(0)}} \\ &= \frac{\mathrm{Ke}\left(g, g\right)^{\gamma y}}{\prod_{i \in \mathrm{Att}} \left(e\left(g, g\right)^{\gamma q(i)}\right)^{\Delta_{i,\mathrm{Att}}(0)}} = \frac{\mathrm{Ke}\left(g, g\right)^{\gamma y}}{e\left(g, g\right)^{\gamma q(0)}} = K. \end{aligned} \tag{5}$$

(9) PlainRecover$(\mathrm{EDO}, K) \rightarrow (M)$: given the EDO from the cloud, the data consumer runs the pseudorandom number generator to generate $s$ indices $I_1, I_2, \ldots, I_s$ of ciphertext shares $\mathrm{CS} = (\mathrm{CS}_1, \mathrm{CS}_2, \ldots, \mathrm{CS}_s)$ under the seed CI. Then, the data consumer gets more than $t-1$ $\mathrm{CS}_i$, $i = 1, 2, \ldots, s$, from the DHT network. From these $\mathrm{CS}_i$, $i = 1, 2, \ldots, s$, the data consumer can reconstruct the polynomials $f_1(x), f_2(x), \ldots, f_b(x)$ using Lagrange's interpolation. Then, the data consumer gets $\mathrm{EC} = (m_1, m_2, \ldots, m_b)$ from these polynomials and generates the associated ciphertext $A$. Finally, the original ciphertext $C_M$ is generated by running DeAssocipher$(A) \rightarrow (C_M)$ algorithm. The plaintext is recovered from $M = \mathrm{Dec}(K, C_M)$.

*4.3. Outline of SCSD Scheme.* There are two main phases of SCSD scheme, namely, the data encapsulation phase and the data reconstruction phase. The outline of SCSD scheme is illustrated in Figure 2.

In data encapsulation phase (Phase I), the data owner firstly runs the algorithm Enc$(M) \rightarrow (C_M, C_K)$ to generate the ciphertext of sensitive data under ABE. Then, the data owner runs the algorithms Associpher$(C_M) \rightarrow (A)$, CipherShareGen$(A) \rightarrow (\mathrm{CS}, C')$, and ShareDistribute$(\mathrm{CS}, C_K) \rightarrow (\mathrm{CI}, \mathrm{AJ})$ in turn to get the ciphertext shares and attribute shares and then distributes the shares to the DHT network. Besides, the data owner runs the algorithm EDOGen$(\mathrm{Attri}_o, C^*, C', \mathrm{CI}, \mathrm{AJ}) \rightarrow (\mathrm{EDO})$ to get the EDO and then sends the EDO to the cloud.

In data reconstruction phase (Phase II), the data consumer firstly runs the algorithm KeyRecover$(\mathrm{EDO}, \mathrm{USK}) \rightarrow (K)$ to generate the access key of ciphertext before the EDO expires. Note that if the data consumer does not satisfy the ABE threshold policy defined by the data owner, he can not recover the access key successfully. Then, the data consumer runs the algorithm PlainRecover$(\mathrm{EDO}, K) \rightarrow (M)$ to get the ciphertext and finally recovers the sensitive data.

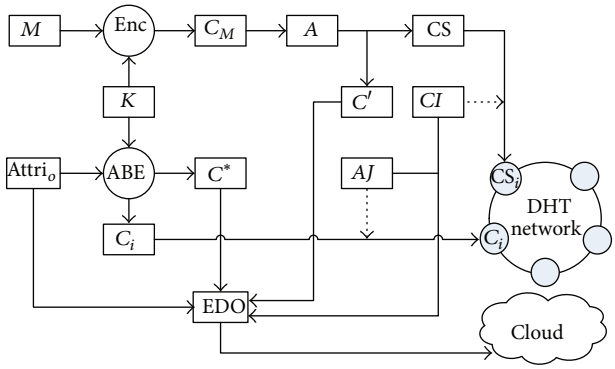


Figure 2: The outline of SCSD scheme.

Table 1: Comparisons of security properties.

| Characteristics | Scheme | | |
|---|---|---|---|
| | SCSD | Reference [5] | Reference [6] |
| Key destruction | √ | √ | √ |
| Ciphertext destruction | √ | × | √ |
| Cryptanalysis attack resistance | √ | × | √ |
| Brute-force attack resistance | √ | × | √ |
| Sybil attack resistance | √ | × | × |
| Asymmetric encryption | √ | × | × |
| Fine-grained access control | √ | × | × |
| Simple key management | √ | × | × |

## 5. Analysis and Performance

In this section, we evaluate our SCSD scheme by modularizing it into two parts, namely, security analysis and scheme performance.

*5.1. Security Analysis.* In the applications of our scheme, because adversaries can not specify the particular object of attack before the expiration timestamp, we assume that the copies of EDO stored in the cloud are secure during this time. Besides, because the attribute shares and ciphertext shares stored in the DHT network will be discarded after the expiry of EDO, once the DHT network is updated periodically, the contents of EDO copies will be unreadable.

There are mainly two kinds of attack aiming at our scheme. The first one is cracking the expired EDO copies stored in the cloud through cryptanalysis attack and brute-force attack. Despite the fact that the attribute shares and ciphertext shares are discarded, there are still EDO copies stored in the cloud. The other kind of attack is aiming at collecting the attribute shares and ciphertext shares in the DHT network before the expiration timestamp of EDO, and these shares will be used in the tracing attack against the EDO copies stored in the cloud.

Therefore, the security of our scheme is mainly affected by two aspects. One is the security of encryption algorithm used in the sensitive data encryption under the access key, which depends on the capability of resisting the cryptanalysis attack and brute-force attack. The other is the security of DHT network that stored the attributes shares and ciphertext shares, which depends on the capability of resisting sniffing attack, hopping attack, and other DHT Sybil attacks. So, we make the security analysis of our scheme based on these two aspects as follows. The brief comparisons of security properties of our SCSD scheme [5, 6] are summarized in Table 1.

*5.1.1. The Security of Encryption Algorithm.* The brute-force attack is implemented by trying any possible decryption keys on the ciphertext to recover the plaintext. This kind of attack is based on the integrity of ciphertext. So, adversaries should first get the integrated ciphertext before implementing the brute-force attack. In our scheme, however, the sensitive data is first encrypted under the random access key and then the ciphertext is associated and extracted. Because every block of the associated ciphertext is correlated with each other, once some of the blocks are extracted, the remaining blocks will be no more integrated. Therefore, without the integrated ciphertext, adversaries can not recover the sensitive data by the brute-force attack.

Besides, implementing the traditional cryptanalysis attack is also based on an integrated ciphertext. Because the remaining ciphertext blocks stored in the cloud are incomplete, the traditional cryptanalysis attack had no effect on our scheme.

*5.1.2. The Security of DHT Network.* In the following, we will discuss whether adversaries can crack the EDO copies by attacking the DHT network before the expiration timestamp of EDO. Because adversaries can not specify the particular object of attack before the expiration timestamp, the adversaries may try to get as many attribute shares and ciphertext shares as possible during this time. For example, the adversaries may keep on attacking the DHT network in order to get enough shares. However, this kind of attack will bring expensive cost to the adversaries.

Due to the characteristic of DHT network, the method of attacking the DHT network to get the attribute shares and ciphertext shares is very difficult. Reference [5] has made detailed analysis aiming at various kinds of DHT attacks by performing simulations in the Vuze DHT network. The result shows that it is impossible for the adversaries to get enough shares from DHT network by implementing sniffing attack, hopping attack, and other DHT attacks. Therefore, in the same way, the adversaries in our scheme also can not get enough attribute shares or ciphertext shares by attacking the DHT network in order to crack the EDO copies stored in the cloud.

*5.2. Performance and Optimization.* In this section, we first make a performance evaluation of SCSD on the time cost in both the data encapsulation phase and the data reconstruction phase, respectively. Then, we implement the parameter optimization by analyzing the tradeoff between security and availability of our scheme.

*5.2.1. Performance Evaluation.* In Phase I, the communication overhead is mainly caused by the distribution of ciphertext shares and attribute shares to the DHT network. The computation overhead is mainly caused by the ABE algorithm on the access key, the symmetric encryption algorithm on sensitive data, and the association and the shares generation algorithm on ciphertext. In Phase II, the communication overhead is also mainly caused by the collection of ciphertext shares and attribute shares from the DHT network. The computation overhead is mainly caused by the reconstruction of the access key and the ciphertext.

Based on the above analysis, we execute our SCSD scheme and measure the times spent in the two main phases. For the sake of simplicity, we set the total shares $n = s$ and the threshold $k = t$ for the ciphertext shares and attribute shares, respectively. The evaluation uses an Intel G2130 3.2 GHz with 4 GB of RAM, Java 1.6, and a broadband network. The times of the two main phases are shown in Figure 3.

Figure 3 shows that the data collection and reconstruction phase is relatively fast. The time cost of data encapsulation and distribution, however, is quite large. Fortunately, a simple pretreatment, pregenerating the access key and prepushing shares into the DHT network, can be implemented. As shown in Figure 3, this pretreatment can lead the time of data encapsulation phase to a fixed 1.6 s. Thus, the performance of SCSD scheme is relatively effective and efficient.

*5.2.2. Parameter Optimization.* Next, we assume that the adversaries have comprised 5% of the nodes in a thousand-node DHT network. We will show how the security and the availability of our scheme are affected by the parameters $n$ and the threshold $k$. The probability that an adversary captures sufficient shares to reconstruct the ciphertext shares is shown in Figure 4. It is clear that increasing the number of shares can decrease the adversary's success probability. Furthermore, the security can also be enhanced as the threshold increases.

As shown in Figure 5, the availability is also affected by the parameters. The maximum timeout gets longer as the number of shares increases. And longer timeout can also be supported by smaller threshold since the scheme can tolerate more share loss. So, the choice of threshold can represent a tradeoff between security and availability. High threshold can provide more security and low threshold can provide longer lifetime. Therefore, by choosing the proper share number and threshold, we can get a tradeoff of high security and good availability.

Besides the parameters, there are other kinds of optimizations for our scheme. Because of the adoption of ABE algorithm, our SCSD scheme can implement one-to-many authorization and access control flexibly. Moreover, the access key can be used repeatedly in the condition of timely processing huge volume of data while the security requirement is lower. And if the requirement of security is higher, the ciphertext shares $CS = (CS_1, CS_2, \ldots, CS_s)$ and the attribute shares $\{C_i = R_i^{\gamma}\}_{i \in \text{Attri}_o}$ can also be distributed to different DHT networks, respectively, one to Vuze and the other to OpenDHT [16], which will improve the security of our scheme significantly.

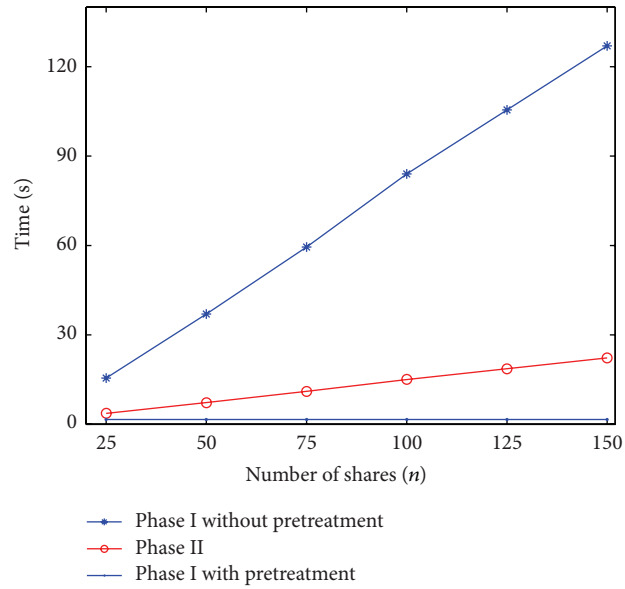


Figure 3: Performance of SCSD scheme.

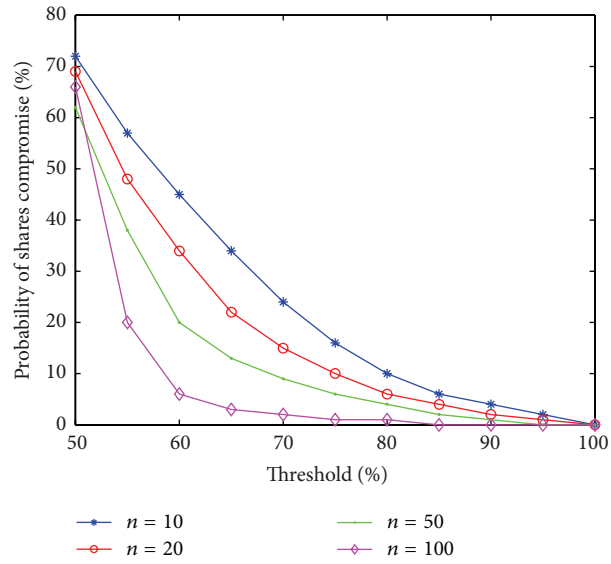


Figure 4: Parameters and security.

## 6. Conclusion

In cloud storage system, secure data destruction is one of the problems that need to be addressed in data security. Many data destruction schemes have been proposed in recent years. However, there are still some limitations. In this paper, we mainly focus on the ciphertext destruction and propose a secure ciphertext self-destruction scheme with attribute-based encryption called SCSD, which applies the attribute-based encryption and the distributed hash table technology to the process of data destruction in the cloud

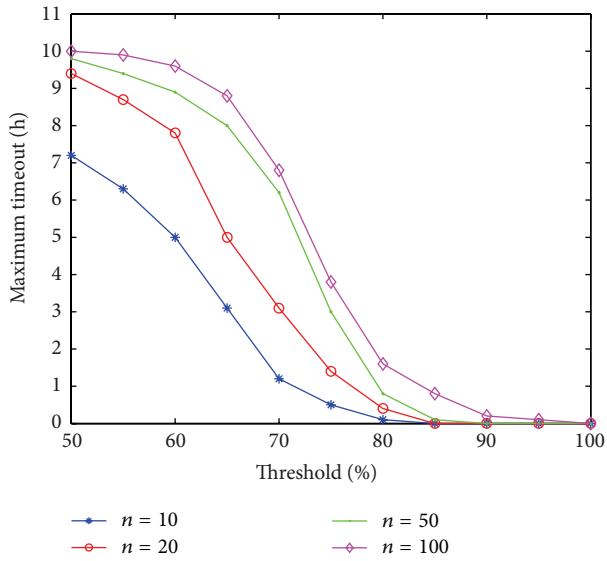


Figure 5: Parameters and availability.

storage environment. Compared with the current schemes, our scheme can resist the traditional cryptanalysis attack as well as the Sybil attack in the DHT network. Besides, the performance of SCSD scheme is relatively effective and efficient.

## Conflict of Interests

The authors declare that they have no conflict of interests regarding the publication of this paper.

## Acknowledgments

This work was supported by the School Innovation Foundation and the Doctorial Foundation under Grant 2014JY170. The authors thank the anonymous reviewers for their useful comments and suggestions.

## References

[1] M. Armbrust, A. Fox, R. Griffith et al., "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] D. P. Shah and P. A. Ganatra, "Comparative study of data possession techniques for data storage as a service (DSaaS)," *International Journal of Computer Applications*, vol. 80, no. 4, pp. 38–42, 2013.

[3] D. Catteddu, "Cloud computing: benefits, risks and recommendations for information security," in *Web Application Security*, p. 17, Springer, 2010.

[4] L. M. Kaufman, "Data security in the world of cloud computing," *IEEE Security and Privacy*, vol. 7, no. 4, pp. 61–64, 2009.

[5] R. Geambasu, T. Kohno, A. Levy et al., "Vanish: increasing data privacy with self-destructing data," in *Proceedings of the 18th USENIX Security Symposium*, pp. 299–315, Montreal, Canada, August 2009.

[6] G. Wang, F. Yue, and Q. Liu, "A secure self-destructing scheme for electronic data," *Journal of Computer and System Sciences*, vol. 79, no. 2, pp. 279–290, 2013.

[7] R. Perlman, "File system design with assured delete," in *Proceedings of the 3rd IEEE International Security in Storage Workshop (SISW '05)*, pp. 83–88, IEEE, San Francisco, Calif, USA, December 2005.

[8] Y. Tang, P. P. C. Lee, J. C. S. Lui, and R. Perlman, "FADE: secure overlay cloud storage with file assured deletion," in *Security and Privacy in Communication Networks: 6th Iternational ICST Conference, SecureComm 2010, Singapore, September 7–9, 2010. Proceedings*, vol. 50 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 380–397, Springer, Berlin, Germany, 2010.

[9] R. Perlman, "The Ephemerizer: making data disappear," *Journal of Information System Security*, vol. 1, no. 1, pp. 21–32, 2005.

[10] S. Wolchok, S. O. Hofmann, N. Heninger et al., "Defeating vanish with low-cost Sybil attacks against large DHT," in *Proceedings of the 17th Annual Network & Distributed System Security Conference (NDSS '10)*, pp. 1–15, San Diego, Calif, USA, February 2010.

[11] L. Zeng, Z. Shi, S. Xu, and D. Feng, "SafeVanish: an improved data self-destruction for protecting data privacy," in *Proceedings of the IEEE 2nd International Conference on Cloud Computing Technology and Science*, pp. 521–528, IEEE, Indianapolis, Ind, USA, December 2010.

[12] J. Xiong, Z. Yao, J. Ma et al., "A secure document self-destruction scheme: an ABE approach," in *Proceedings of the 15th IEEE International Conference on High Performance Computing and Communications (HPCC '13)*, pp. 59–64, Zhangjiajie, China, November 2013.

[13] D. Frank, *A Distributed Hash Table*, Massachusetts Institute of Technology, 2005.

[14] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology—EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22–26, 2005. Proceedings*, vol. 3494 of *Lecture Notes in Computer Science*, pp. 457–473, Springer, Berlin, Germany, 2005.

[15] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[16] S. Rhea, B. Godfrey, B. Karp et al., "OpenDHT: a public DHT service and its uses," *Computer Communication Review*, vol. 35, no. 4, pp. 73–84, 2005.