Research Article

Assured Deletion: A Scheme Based on Strong Nonseparability

Zhenjie Xie, Wei Fu, Jianqiao Xu, and Tingting Zhu

Department of Information Security, Navy University of Engineering, Wuhan, China

Correspondence should be addressed to Wei Fu; lukeyoyo@tom.com

Received 20 December 2021; Accepted 1 March 2022; Published 16 March 2022

Academic Editor: Min Xia

Copyright © 2022 Zhenjie Xie et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Cloud storage services bring great convenience to users, but also make data owners lose direct control of their data. How to ensure that deleted cloud data can never be recovered by cloud servers or attackers is a key issue in the field of cloud storage security, which is important to protect user privacy and data confidentiality in the cloud environment. Most existing schemes have the drawbacks of overreliance on key destruction and having the ability to decrypt part of the ciphertext after cracking the key. To solve these problems, a novel cloud data assured deletion scheme based on strong nonseparability is presented. By combining XOR operation with the block cipher, the cipher data become strongly nonseparable; thus, destroying any piece of cipher data will result in unrecoverable original data. Theoretical analysis and experimental results both show that the scheme achieves the expected goals of assured deletion, which has obvious performance advantages and stronger security compared with similar schemes.

1. Introduction

In cloud storage application mode, users enjoy the benefits of flexible space, real-time sharing, disaster tolerance, and other benefits. However, they lose direct control over the data they own. When users want to delete data stored in the cloud, they need to encrypt the data using keys generated by their own means, upload the encrypted data to the remote server, and then destroy the keys. However, if the keys are compromised or the cipher is violently cracked, the deleted original data is still at risk of being recovered. Even if it cannot be decrypted, the complete cipher data can still be used for ciphertext analysis. In addition, under the common working mode of block ciphers (e.g., ECB and CBC), if an attacker has the key, the intercepted cipher fragments can be decrypted. To deal with these threats, this paper presents a cloud data assured deletion scheme based on strong nonseparability. By combining XOR operation with the block cipher, the cipher data are strongly nonseparable, and destroying any part of the cipher data will result in the original data not being recovered. Users host the key and some cipher data to a trusted third party, which destroys these cipher data and the key when the deletion operation is initiated. The strong nonseparability ensures that recovering any part of the original data without the complete cipher data is computationally infeasible, thus significantly enhancing the nonrecoverability of the data after the deletion operation. This scheme achieves the strong nonseparability by confusing data blocks directly participating in cryptographic operations through XOR operation, which reduces the number of cryptography computations. Therefore, it has obvious performance advantages and stronger security over existing similar assured deletion schemes.

2. Related Works

To realize the assured deletion of cloud data, Perlman first proposed the idea of "encrypt data, destroy key" [1]. Firstly, the user encrypts the original data by generating a random key, then uploads the ciphertext to the remote server, and...
hosts the encryption key to a trusted third party. When performing data deletion, the trusted third party destroys the key to ensure that the cipher data on the remote server cannot be decrypted, thus achieving assured deletion. Tang et al. proposed FADE system [3, 4], which implements a policy-based assured deletion mechanism for cloud data. Mo et al. proposed a user-based assured deletion scheme [5] that does not rely on third parties; users only need to keep a small number of keys to achieve fine-grained assured deletion. Geambasu et al. proposed a scheme [6] which uses a threshold secret sharing mechanism to divide the encryption key and save it on a DHT (Distributed Hash Table) node, using the automatic update mechanism of DHT to delete the encryption key regularly. Feng and Tan proposed a data assured deletion scheme [7] based on trust value for cloud storage; the core is to evaluate the trustworthiness of DHT nodes, selecting a node with higher trustworthiness to store the key component, and reduce the probability that users cannot access their sensitive data during the authorized time.

Rivest pointed out that for existing encryption systems, as long as the key is mastered, part of the cipher data obtained can be decrypted [2]. This facilitates the attacker. On the one hand, any cipher fragments have a direct value because they can be decrypted to obtain the corresponding plaintext. On the other hand, any cipher fragment can provide useful information for cracking the key. To address these issues, Rivest proposed the concept of strongly nonseparable and proposed a way to construct a strongly nonseparable encryption scheme by existing cryptography technology and AONT (All-Or-Nothing-Transform) and an implementation scheme called Package Transform to accommodate scenarios with higher security requirements.

On the basis of Rivest’s Package Transform scheme, Luo proposed a strongly nonseparable encryption scheme [8]. Luo’s scheme first converts the original message sequence into a strongly inseparable pseudomessage sequence using AONT conversion and then encrypts the pseudomessage sequence using a random key to generate a cipher sequence. To improve the computational efficiency, Luo’s scheme uses pseudorandom function (PRF) and hash operation to, respectively, replace two times block encryption in the Package Transform scheme. It is important to note that the Package Transform scheme implements AONT, which is a preprocessing process used to achieve strong nonseparability and cannot replace encryption operations by itself, while Luo’s scheme is a complete encryption scheme which implements strong nonseparability. The scheme proposed by Zhang et al. [9, 10] uses a similar approach, but the use of bilinear mapping may result in high computational overhead.

Liu et al. proposed a blockchain-based verification scheme for deletion operation in the cloud [11]. Users invoke smart contracts to prove identity to the cloud server, and then, the cloud server deletes the data and generates a blockchain with deletion evidence embedded, while users can verify the results of data deletion without a trusted third party. Du et al. proposed a cloud data assured deletion scheme based on cipher reencryption combined with override verification [12]. By reencrypting and changing the access control strategy corresponding to the ciphertext, data fine-grained deletion can be achieved. A searchable path hash binary tree based on dirty data block override is constructed to verify the correctness of the data to be deleted after overwriting. Zhang proposed an instantaneous deterministic deletion method of cloud storage data based on feature iteration [13], which extracts and classifies the features of redundant data in cloud storage data, iterates until convergence, and achieves high-performance deletion of redundant data. In addition, some assured deletion or self-destruction schemes for cloud data based on DHT [10, 14], attribute-based encryption [15–18], and identity-based encryption [19, 20] are proposed in recent years. Xiong et al. [21] made an in-depth analysis and comments on the related research work in recent years from three aspects: trusted execution environment, key management, and access control strategy.

The above schemes use some new technologies, but require considerable changes to the cloud storage service framework, making it difficult to directly apply them to existing cloud storage systems, complex to implementation, and significantly increasing the computational time cost or number of interactions, while the scheme in this paper does not change the existing system architecture, but fully exploits the mature cryptography technology, which achieves the goals of assured deletion by making the ciphertext strongly nonseparable.

3. Assured Deletion Scheme Based on Strong Nonseparability

The cloud storage system in reality is full of various security threats. To achieve assured deletion of cloud data, an efficient assured deletion scheme for cloud data based on strong nonseparability is proposed. This section first describes the assured deletion system from the perspective of the model, combs the real threats and proposes the expected design goals, then introduces the concept of strongly nonseparable, next describes the main algorithms, processes, and assured deletion mechanisms in detail, and finally gives the performance analysis of the scheme.

3.1. System Model

3.1.1. System Composition. To illustrate the assured deletion of cloud data, an assured deletion system with four entities is introduced, as Figure 1 illustrated.

Among them, the cloud storage server (CSS) provides users with cloud storage and data sharing services. Trusted third party (TTP) is the entity trusted by users, which is responsible for hosting keys and partial cipher data and for completely destroying the corresponding key and cipher data when performing the deletion operation. Data owner (DO) performs a specific encryption algorithm on the original data, uploads the output to CSS and TTP, and authorizes access for some users. Authorized users (AU) are those who are allowed to download and decrypt specific data. Assured deletion can be initiated by the DO or triggered by a strategy such as shelf life.
follows security threats:

(a) CSS does not perform data deletion operations, but only prohibits user access
(b) Attackers or malicious cloud service providers attempt to decrypt or mine the user data they hold
(c) Attackers intercept or crack the data encryption key to decrypt the acquired cipher data

### 3.1.2. Reality Threats.

For traditional cloud storage systems, cloud data may still be retained after a deletion request, which poses a threat to user privacy and data confidentiality.

Trusted third party Authorized users

![Figure 1: Assured deletion system model.](image)

**Algorithm 1**

| Input: $F$, $K$, $C'$ |
|-----------------------|
| Output: $K'$ |
| Randomly generate an initial vector $m_0'$ |
| Randomly generate a data encryption key $K$ |
| for $i$ in $[1, n]$: |
| $m_i' = m_{i-1}'$ |
| $c_i' = \text{Enc}_K(m_i')$ |
| end for |
| $C' = (c_0', c_1', c_2', \ldots, c_n')$ |

In fact, there is no credible third-party entity in some cloud storage systems. At this time, DO can retain the partial cipher data and destroy it when performing deletion.

### 3.1.3. Expected Goals.

Within the framework of this system, in order to respond to these real threats and be operable, assured deletion should meet the following objectives:

(a) Assured deletion must be valid, that is, no one (including the DO) can recover all or part of the original data with limited computing resources after performing the data deletion
(b) Even if an attacker acquires all data stored by the CSS or if the cloud service provider itself is malicious, it cannot restore any of the original data
(c) Even if an attacker has the data encryption key, he cannot decrypt incomplete cipher data fragments
(d) The algorithm introduced by assured deletion must be easy to implement and will not significantly burden users

### 3.1.4. Safety Assumptions.

To make the scheme feasible, the following assumptions need to be made:

(a) TTP is trustworthy and technically can ensure that the corresponding keys and data are securely destroyed
(b) The temporary data blocks generated by users will not be saved, that is, attackers cannot obtain the decrypted plaintext through the users

#### 3.2. Strongly Nonseparable.

Strongly nonseparable [2] introduced by Rivest guarantees that any missing cipher data will result in the failure to decrypt any part of the plaintext, as defined below.

Hypothesis $Γ$ is a block cipher mechanism used to convert $s$ plaintext sequence $m_1, m_2, \ldots, m_s$ to $t$ cipher sequence $c_1, c_2, \ldots, c_t$. If it is satisfied that restoring any plaintext data before obtaining and decrypting all $t$ cipher blocks is computationally infeasible, it is called the encryption mechanism $Γ$ can make a sequence strongly nonseparable.

In the face of a strongly nonseparable encryption mechanism, the cipher fragments obtained by the attacker no longer have the value of direct decryption, nor can they provide a useful reference for cracking the key, so the security is significantly enhanced. Based on the definition of strongly nonseparable, we can use existing cryptographic technology to construct encryption schemes that satisfy the strong nonseparability.

#### 3.3. Scheme Implementation.

This section describes in detail the algorithms and processes for data encryption and data decryption of the scheme and gives the mechanism and security analysis of data deletion. Make $\text{Enc}$ a block encryption algorithm, $\text{Dec}$ the corresponding decryption algorithm for $\text{Enc}$, and $F = (m_1, m_2, \ldots, m_n)$ the original message sequence.

##### 3.3.1. Data Encryption.

The pseudocode for the data encryption algorithm is as follows.

The data encryption algorithm is executed by the DO. The execution process is shown in Figure 2. After execution, the DO randomly selects some data blocks in the cipher sequence $C'$ to form a set $Ψ$. Specifically, make $Ψ = \{c_0'\}$; DO hosts $Ψ$ and key $K$ together to a TTP. The remaining
3.3.2. Data Decryption. The pseudocode for the data decryption algorithm is as follows.

Data decryption algorithm is executed by AU when accessing cloud data. The execution process is shown in Figure 3. AU first retrieves the key $K$ and the partial cipher data set $\Psi$ from the TTP, then retrieves the remaining cipher data set $\Omega$ from the CSS, merges $\Psi$ and $\Omega$ to get the complete cipher sequence $C'$, and then executes the data decryption algorithm to recover the original message sequence $F$.

It is worth mentioning that if all $m_i$ appearing in the above algorithms are fixed to the initial vector $m_0'$, the expected security characteristics can also be achieved. This simplifies the scheme representation further, but may result in the same plaintext blocks being converted to the same cipher blocks. Therefore, it is necessary to use the block cipher working mode other than ECB to hide the formatting rules and statistical characteristics of the plaintext. Experiments show that this simplification does not achieve performance improvement, so it will not be repeated later.

3.3.3. Data Delete Mechanism. When performing the deletion operation, the key $K$ and the partial cipher data set $\Psi$ are destroyed by the TTP. Due to the strong nonseparability of cipher data, any missing cipher data will result in the failure to decrypt any part of the plaintext.

The following security description analyzes whether the scheme achieves the expected security goals after performing deletion operations from the point of view of the strongest attacker.

TTP destroys the key $K$ and the partial cipher data set $\Psi$ (make $\Psi = \{c_0'\}$) after deletion. Assume that the strongest attacker $A$ gets all remaining data $\Omega$ (make $\Omega = \{c_1', c_2', \ldots, c_n'\}$) stored by CSS and has the key $K$. However, $A$ is infeasible to recover $m_0'$ without $c_0'$; thus, no fragment of the original message $F$ can be recovered. Even if the attacker $A$ gets part of the plaintext blocks (make it $m_i'$), because $A$ does not know $m_i'$, it is not possible to compute $m_i'$ contained in it; thus, no more plaintext blocks can be obtained.

In Luo’s scheme [8], $m_i' = m_i \oplus f_{rk}(i)$; $f$ is a pseudorandom function. Attacker $A$ can get $m_i'$ by decrypting $c_i'$. In the case of obtaining the plaintext block $m_i$, $f_{rk}(i)$ can be obtained. Pseudorandom functions usually do not satisfy the characteristics of cryptographic algorithms. As long as the seed $rk$ is calculated inversely, other plaintext blocks can be restored.
Table 1: Performance analysis of schemes.

| Scheme                  | Storage overhead | Calculation overhead | Communication overhead | Strong nonseparability | Confidentiality |
|-------------------------|------------------|----------------------|------------------------|------------------------|-----------------|
| Encryption only         | $n\beta + k$     | $n\text{ENC}$        | $n\beta + k$           | No                     | Yes             |
| Package Transform [2]   | $(n + 1)\beta$   | $3n\text{XOR} + 2n\text{ENC}$ | $(n + 1)\beta$         | Yes                    | No              |
| Luo’s scheme [8]        | $(n + 1)\beta + k$ | $2n\text{XOR} + n\text{HASH} + (n + 1)\text{ENC} + n\text{PRF}$ | $(n + 1)\beta + k$     | Yes                    | Yes             |
| This scheme             | $(n + 1)\beta + k$ | Enc: $3n\text{XOR} + n\text{ENC} + 1\text{PRF}$ | Dec: $3n\text{XOR} + n\text{ENC}$ | $(n + 1)\beta + k$ | Yes             |

Table 2: The configuration of experimental computer.

| Item                  | Configuration                      |
|-----------------------|------------------------------------|
| Machine type          | PC                                 |
| Operation system      | Windows 10 64-bit                  |
| CPU                   | Intel Core i7-10710U               |
| CPU quantity          | 6                                  |
| Memory                | 16 GB LPDDR3 2133 MHz              |
| Hard disk             | SAMSUNG MZVLB512HBJQ-000L7         |

Table 3: The key operation implementation.

| Operation            | Implementation                     |
|----------------------|------------------------------------|
| Block encryption     | Python-Crypto.Cipher.AES           |
| Pseudorandom         | Python-random                      |
| Hash                 | Python-hashlib.md5                 |
| Data block XOR       | Python large integer bitwise XOR (“^”) |

To sum up, with limited computing resources and existing technology, an attacker cannot obtain any plaintext after performing the deletion operation under the assumptions. Therefore, the scheme achieves data assured deletion and has higher security than Luo’s scheme.

3.4. Performance Analysis. This section makes a theoretical analysis of the scheme performance from three aspects: storage, calculation, and communication. Table 1 shows the performance analysis results of encryption only, Package Transform [2], Luo’s scheme [8], and this scheme.

3.4.1. Storage Overhead. As mentioned earlier, for the original message sequence $F$ with $n$ blocks, the cipher sequence $C'$ generated by the encryption algorithm has $(n + 1)$ blocks. Assuming that the size of each plaintext and cipher data block is $\beta$, the size of key $K$ is $k$, and the partial cipher data set $\Psi$ stored by TTP includes only one block. Thus, for each file stored, the storage overhead of CSS is $n\beta$, while it is $(\beta + k)$ for TTP. The overall storage overhead of the scheme is $(n + 1)\beta + k$. The DO does not need to save any data locally after uploading.

3.4.2. Calculation Overhead. The DO of this scheme needs $3n \cdot \text{XOR} + n \cdot \text{ENC} + 1 \cdot \text{PRF}$ operations to perform the data encryption algorithm, while AU needs $3n \cdot \text{XOR} + n \cdot \text{ENC}$ operations to perform the data decryption algorithm. Among them, XOR stands for data block XOR operation, ENC means block encryption/decryption operation, and PRF stands for pseudorandom operation (including randomly generating an initial vector). Luo’s scheme [8] requires $2n \cdot \text{XOR} + n \cdot \text{HASH} + (n + 1) \cdot \text{ENC} + n \cdot \text{PRF}$ operations to perform file upload or download algorithms, where HASH refers to the hash operation. It can be seen that, compared with Luo’s scheme, this scheme significantly reduces pseudorandom operations and avoids hash operations, thus significantly improving the computational efficiency.

3.4.3. Communication Overhead. After performing the data encryption algorithm, the DO uploads $n\beta$ data to CSS and $(\beta + k)$ data to TTP. Accordingly, AU downloads $n\beta$ data from CSS and $(\beta + k)$ data from TTP. The total communication overhead for a complete file upload or file access is $(n + 1)\beta + k$ regardless of the handshake cost required to establish the communication channel.

4. Experiments and Analysis

To verify the validity of this scheme, the main algorithms and processes of this scheme are implemented using Python programming; also, the schemes of encryption only, Package Transform [2], and Luo’s [8] are implemented. The computational efficiency of the data encryption/decryption algorithms of each scheme is expected to be compared through comparative experiments.

4.1. Environment and Parameters. The configuration of the experimental computer is listed as Table 2.

128-bit key ($k = 16$B) was used in the experiments. The original file size was 10 MB (using 1 MB, 100 MB, 1 GB, and other sizes will get similar results; we use 10 MB file as an example), and the data block size $\beta$ was 1 KB, 2 KB, 4 KB, 8 KB, and 16 KB in turn. Experiments with the same parameters were repeated 50 times, and their average values were taken as the final results.

Some Python libraries are used for key operations such as block encryption, hash, and pseudorandom involved in the algorithm. The specific implementation method is shown in Table 3.

To ensure the accuracy of the comparison, each scheme should be as consistent as possible in programming style, be optimized as possible in its own scheme framework, and use the same parameters.
4.2. Data Encryption Test. In the data encryption test, each scheme executes the encryption algorithm for a random file of 10 MB in size, outputs a 10 MB file to upload to CSS, and also uploads a data block to TTP (except for encryption only). The time cost of performing the encryption algorithm of each scheme at different block sizes is shown in Figure 4.

Figure 4 shows that this scheme takes significantly less time cost to execute the encryption algorithm than Package Transform and Luo’s scheme (about 2/3 to 3/4 of Luo’s), which is about two to three times as long as encryption only. In addition, the time cost of each scheme decreases as the block size increases, because an increase in the block size
means a decrease in the total number of blocks, so the number of operations performed decreases.

4.3. Data Decryption Test. In the data decryption test, each scheme executes the decryption algorithm on the 10 MB file generated by its own execution of the encryption algorithm. The output file of each scheme is consistent with the original file through hash verification. The time cost of performing the decryption algorithm of each scheme at different block sizes is shown in Figure 5.

As can be seen from Figure 5, the decryption algorithm time consumed by each scheme is essentially the same as the encryption algorithm.

Combining the above test results, this scheme is an assured deletion encryption scheme based on strong nonseparability. It achieves greater security with only two to three times the computation overhead of encryption only. Because the hash operation is avoided and pseudorandom operation is reduced, the computation overhead is about 2/3 to 3/4 of Luo’s scheme.

5. Conclusion

Aiming at the assured deletion of cloud data, a novel scheme based on strong nonseparability is presented. Compared with the traditional schemes of “encrypt data, destroy key,” making the ciphertext strongly nonseparable is a promising way to achieve the assured deletion. By destroying the key and part of the cipher data through a TTP, the nonrecoverability of the original data after deletion operation is significantly enhanced. Compared with the existing similar schemes, this scheme achieves strong nonseparability by adding XOR operation instead of hash operation. The theoretical analysis and experimental results show that the proposed scheme achieves the expected design goals of assured deletion with less computation overhead, having obvious performance advantages and stronger security in similar schemes.

In fact, refining key management and realizing strong nonseparability of ciphertext are two ideas to realize assured deletion, which are not exclusive. If the two ideas are combined, the “credibility” of cloud data deletion will be further improved.

Data Availability

The codes and data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This paper is supported by NSF of China (No. 61672531).

References

[1] R. Perlman, “File system design with assured delete,” in In Third IEEE International Security in Storage Workshop, San Francisco, CA, USA, 2005.
[2] R. L. Rivest, “All-or-nothing encryption and the package transform,” in In International Workshop on Fast Software Encryption. pp. 210–218, Springer, Berlin, Heidelberg, 1997.
[3] Y. Tang, P. Lee, J. Lui, and R. Perlman, “Fade: secure overlay cloud storage with file assured deletion,” in Security and Privacy in Communication Networks of Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. S. Jajodia and J. Zhou, Eds., pp. 380–397, Springer Berlin Heidelberg, 2010.
[4] Y. Tang, P. Lee, J. Lui, and R. Perlman, “Secure overlay cloud storage with access control and assured deletion,” IEEE Transactions on Dependable and Secure Computing, vol. 9, no. 6, pp. 903–916, 2012.
[5] Z. Mo, Y. Qiao, and S. Chen, “Two-party fine-grained assured deletion of outsourced data in cloud systems, in: distributed computing systems (ICDCS),” in In 2014 IEEE 34th International Conference on Distributed Computing Systems, pp. 308–317, Madrid, Spain, 2014.
[6] R. Geambasu, T. Kohno, A. A. Levy, and H. M. Levy, “Vanish: increasing data privacy with self-destructing data,” USENIX Security Symposium, vol. 316, p. 10-5555, 2009.
[7] F. Guilan and T. Liang, “Data assured deletion scheme based on trust value for cloud storage,” Computer Science, vol. 41, no. 6, pp. 108–122, 2014.
[8] Y. C. Luo, Research on public auditing for shared data and assured deletion in cloud storage. [M.S. Thesis], National University of Defense Technology, Changsha, China, 2015.
[9] Z. Kun, Y. Chao, M. Jianfeng, and J. W. Zhang, “Novel cloud data assured deletion approach based on ciphertext sample slice,” Journal on Communications, vol. 36, no. 11, pp. 108–117, 2015.
[10] W. Lina, Z.-W. Ren, R.-W. Yu, F. Han, and Y.-F. Dong, “A data assured deletion approach adapted for cloud storage,” Acta Electronica Sinica, vol. 40, no. 2, pp. 266–272, 2012.
[11] L. Yining, Z. Yuanjian, L. Rushi, and C. Tang, “Blockchain-based verification scheme for deletion operation in cloud,” Journal of Computer Research and Development, vol. 55, no. 10, pp. 2199–2207, 2018.
[12] D. Ruizhong, S. Pengliang, and H. Xinfeng, “Cloud data assured deletion scheme based on overwrite verification,” Journal on Communications, vol. 40, no. 1, pp. 130–140, 2019.
[13] Z. Xinhua, “Instantaneous deterministic deletion of cloud storage data based on feature iteration,” Application Research of Computers, vol. 37, no. 9, pp. 2840–2843, 2020.
[14] L. Chaoling, C. Yue, and Z. Yanzhou, “A data assured deletion scheme in cloud storage,” China Communications, vol. 11, no. 4, pp. 98–110, 2014.
[15] W. Lina, Z. Ren, Y. Dong, R. Yu, and R. Deng, “A management approach to key-used times based on trusted platform module in cloud storage,” Journal of Computer Research and Development, vol. 50, no. 8, pp. 1628–1636, 2013.
[16] Y. Yu, L. Xue, Y. Li, X. du, M. Guizani, and B. Yang, “Assured data deletion with fine-grained access control for fog-based industrial applications,” IEEE, vol. 14, no. 10, pp. 4538–4547, 2018.
[17] L. Xue, Y. Yu, Y. Li, M. H. Au, X. du, and B. Yang, “Efficient attribute-based encryption with attribute revocation for assured data deletion,” *Information Sciences*, vol. 479, pp. 640–650, 2019.

[18] X. Jinbo, Y. Zhiqiang, M. Jianfeng, F. Li, X. Liu, and Q. Li, “A secure self-destruction scheme for composite documents with attribute based encryption,” *Acta Electronica Sinica*, vol. 42, no. 2, pp. 366–376, 2014.

[19] Y. Zhiqiang, X. Jinbo, M. Jianfeng, Q. Li, and X. Liu, “A secure electronic document self-destructing scheme in cloud computing,” *Journal of Computer Research and Development*, vol. 51, no. 7, pp. 1417–1423, 2014.

[20] X. Jinbo, Y. Zhiqiang, M. Jianfeng, F. Li, and X. Liu, “A secure self-destruction scheme with IBE for the Internet content privacy,” *Chinese Journal of Computers*, vol. 37, no. 1, pp. 139–150, 2014.

[21] X. Jinbo, L. Fenghua, W. Yanchao, J. Ma, and Z. Yao, “Research progress on cloud data assured deletion based on cryptography,” *Journal on Communications*, vol. 37, no. 8, pp. 167–184, 2016.