A Lifecycle Controlling Method of Documents Based on Deadline and Access Times

Jiaqi Yang *
Computer and Information College
Hohai University
Nanjing, China
E-mail: 519604076@qq.com
* Corresponding Author

Xiao Fu
Computer and Information College
Hohai University
Nanjing, China
E-mail: 542098167@qq.com

Zhijian Wang
Computer and Information College
Hohai University
Nanjing, China
E-mail: W51178@sina.com

Yu Wang
Computer and Information College
Hohai University
Nanjing, China
E-mail: Won9805@gmail.com

Abstract—With the growing popularity of the online-service in cloud, users' data security protection has gradually become a new challenge. After one user sends his documents to the other, the files are no longer under the user's control. Once he needs the receiver to destroy data, he cannot ensure that the data is really thoroughly destructed. This paper provides a lifecycle controlling method of files, which is based on deadline and times. Deadline, times and destruction strategies are as attributes in the header of files. Files are made in new ones which contain storage layer and controller layer by double hash calculations. The method does not require intervention of any user or a trusted third one. A file will destruct itself according to the destruction strategy in the attributes after its deadline or times of access. The model presented in this paper has been proven simple and practical in practice and applied in three companies.

Keywords- Documents; Lifecycle; Deadline; Times; Destruction Strategies

I. INTRODUCTION

Cloud is one of the most important technologies in IT for the past few years. It can share the basic institutions, greatly develop the distributed processing, parallel processing and grid computing technology, etc[1]. With the aid of computer and Internet, the enterprise online business and management could be implemented. All business data are to deal with by online system so that to become business intelligence and present forms of KPI, charts and the traceability reports. What's more, all kinds of technical scheme of communication is also in the form of electronic documents, for example, txt, xml, word, etc. In the result of that, electronic documents have become the main carrier of information exchange from one enterprise to another one or from the inner to the outer in the enterprise. Windchill, for example, is famous software which contains a variety of electronic documents. These electronic documents can be used, checked in or checked out by remote revocation which will cause documents copy residual, and transmission in the insecure channel will also cause information disclosure[2]. How to protect the security of electronic documents is becoming an important content in the field of information security.

Encryption is an effective means to protect the data, in recent years, some scholars has presented encryption methods based on the properties. For example, the encryption algorithm based on attribute[3-6] and multiple encryption based on agent[7-9]. In great degree, they have realized data encryption security, However, neither the data lifecycle concept is put forward and they cannot make documents or other files destroy themselves. A Dissolver[10] system for data self-destruction strategy has the concept of lifecycle, but not realize that documents sent to others automatically destroy themselves. Xiong [11-13] put forward a combination encryption of documents which achieved the control of the documents during lifecycle to a large extent. However, his method was based on DHT which was difficult to meet the need of time and performance in office system or p2p network. Yue[14] puts forward a method to split a file into cipher and the key, which was also based on DHT network. The key was selected in random from cipher and then stored in DHT so that it could disappear after the document lifecycle finished, as a result, cipher would not be read any more. However, the randomness of the key is decided by the length of cipher, so researchers can see the complexity of the key is according to cipher and random functions. If most of our documents are small, then the cipher are also very small, as a result, the key are simple relatively. Wang[15] and Zeng[16] successively put forward Vanish and SafeVanish system which improved the efficiency of key management through building a tree structure. However, two of these systems could be attacked by Sybil, and if others get enough weights of key then they can get the key. In special, SafeVanish system increases the length of key and increases the time consumption of encryption and decryption algorithm.

This paper is based on symmetric encryption algorithm (Rijndael) and asymmetric encryption algorithm (2048 - bit RSA algorithm) and presents an idea similar to layers in J2EE so that its structure could be clear. The

electronic documents encryption process is divided into storage layer encryption and control layer encryption, and then these two layers are merged for data persistence layer. Different layers use different encryption algorithm according to the characteristics of the two kinds of encryption algorithms. The header of electronic documents is added attributes of time limit (*deadline*), *access times* and *destruction strategies*. After double hash transformation, it can realize the control of documents lifecycle. The deadline as the attribute added to the header, is a symbol of the end of the life cycle or not and is also as the cipher in the second hash to increase the complexity of cracking documents by brute force. After the ending of an electronic document lifecycle, the document destroys itself by destruction steps through checking the destruction strategy. Cipher and the key are controlled by conditions which are set up according to the practical situation or your decisions so that it can realize fine-grained access control of a document lifecycle. In this paper, the first section describes the requirements of the model and the algorithms design. The second section has multiple safety analysis for this model. The third section, embarked from the system implementation, argues space and time consumption the model through experimental data. The fourth section will compare this model with other method to achieve the feasibility of this model. The last section reviews and prospects this paper, and looks forward to improve the model in the back work.

II. REQUIREMENTS AND MODEL

A. Requirement of design

Three problems need to be solved within the lifecycle of documents. The first one is to prevent documents being intercepted and tampered with during transmitting procedure. The second one is to protect data from leaking so that only a particular user would read it during the lifecycle of documents. And the third problem is to guarantee documents destroying themselves after the lifecycle.

To solve the first problem, the documents need to be encrypted and decrypted to guarantee themselves being unreadable during transmitting procedure, but they would be readable again after arriving at the terminal. To solve the second problem, researchers need set a control conditions into documents to guarantee them being readable during the lifecycle and destroying themselves after the lifecycle. To solve the third one, researchers need give a self-destruction strategy to the documents and DOD 5220-22-M is a perfect one. It can make documents into unreadable and unrecoverable ones.

B. Design of documents encryption

Documents or files are composed of many bits of data, and they are a piece of data in essence. Documents after encryption are divided into two parts: the header and the body. Researchers call data in the header 'Header' and the other one 'Body'. The model of a document after encryption as the following Fig. 1:

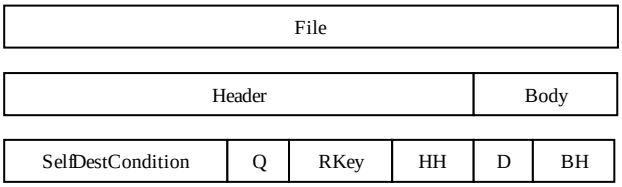

Figure 1. Model of A Document after Encryption

1) Design of encryption procedure: Detailed flow chart of encryption is shown in Fig. 2.

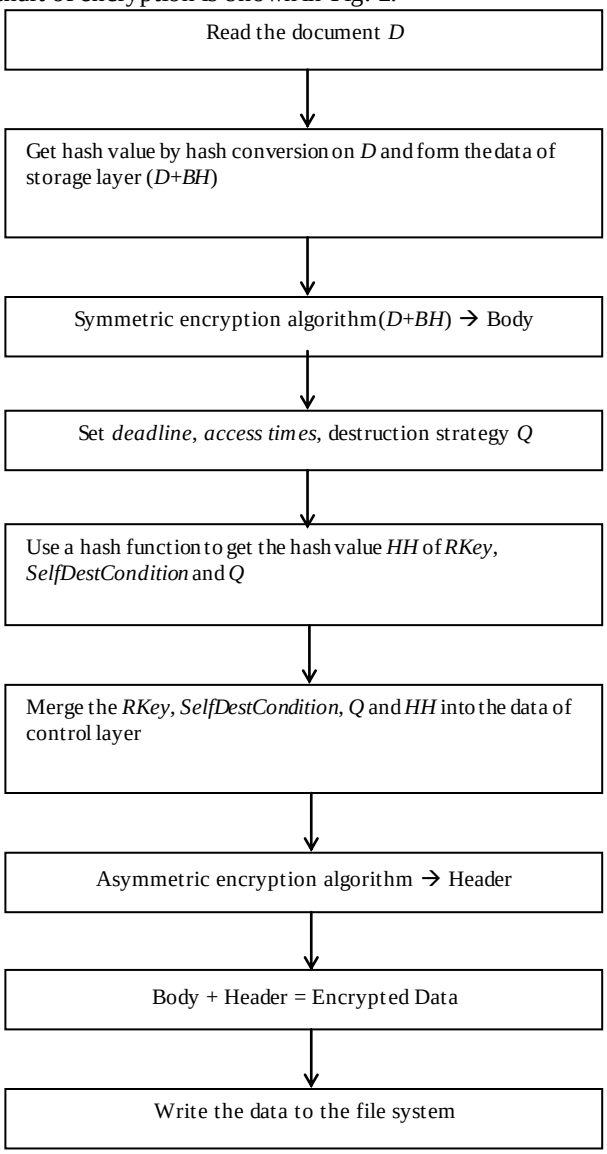

Figure 2. The flow chart of document encryption

- (1)Read the document *D*, then use a Hash function to convert its hash to get the hash value of document data, called *BH*;
- (2)Merge the document *D* and the hash value *BH* into data of the storage layer(*D+BH*);
- (3) Randomly generate a byte array by pseudo random function, Random() and make the array as key of *D+BH*, called *RKey*, then use symmetric encryption algorithm to encrypt *D+BH* to get the Body;

- (4) Generate two variables called *deadline* and *access times* which are used to store the file to use at the end of time and the times of remaining access set by users. If the access times are set 0, it means the terminal user can access the document any times. Both variables are collectively called *SelfDestCondition*;
- (5) Generate a string type variable, which is used to save the destruction logo of a document, called document destruction strategy *Q*. According to the level of documents security, the destruction strategy is divided into three types: one time, three times and seven times in a random sequence coverage;
- (6) Use a hash function to get the hash conversion of the combination of *RKey*, *SelfDestCondition* and *Q*, researchers called the hash value as control parameter *HH*;
- (7)Merge the *RKey*, *SelfDestCondition*, *Q* and *HH* into the data of control layer;
- (8)According to the *PrivateKey* provided by the terminal users, use asymmetric encryption algorithm encrypt control layer data to get the Header;
- (9)Merge the encrypted Header and the Body to form the data of a document File;
- (10) The File is written to the file system and data encryption is completed.

*2) Design of symmetric encryption algorithm:*Assume *D* as the data of a document and then use a hash function to convert the hash. This hash process to get hash information *BH* is called function BHash (*Data*). Process of pseudo random to generate Random key *RKey* is called the function Random (). *RKey* is responsible for the Body, and stored in the Header. Symmetric encryption process is called SymmetricEncrypt (*PlainText*, *Key*) and researchers get Body by the following formulas:

$$BH = BHash(D) \quad (1)$$

$$RKey = Random() \quad (2)$$

$$Body = SymmetricEncrypt(D+BH, RKey) \quad (3)$$

3) Design of asymmetric encryption algorithm: Assume that the judgment conditions after the task finished is called *SelfDestCondition*(including *deadline* and *access times*) and destruction strategy. Use a hash function to convert the hash again and the hash process to get hash value *HH* is called HHash (*Data*), Asymmetric encryption process is called AsymmetricEncrypt (*PlainText*, *PrivateKey*) and the process to generate *PublicKey*, *PrivateKey* is called Generator(). Then take the *PrivateKey* to encrypt the Header and the *PublicKey* is used to decode. The Header is obtained by the following formulas:

$$HH = HHash(SelfDestCondition+Q+RKey) \quad (4)$$

$$(PublicKey, PrivateKey) = Generator() \quad (5)$$

$$Header = AsymmetricEncrypt(SelfDestCondition + Q + RKey + HH, PrivateKey) \quad (6)$$

C. Design of decryption

The decryption flow chart shown in Fig. 3:

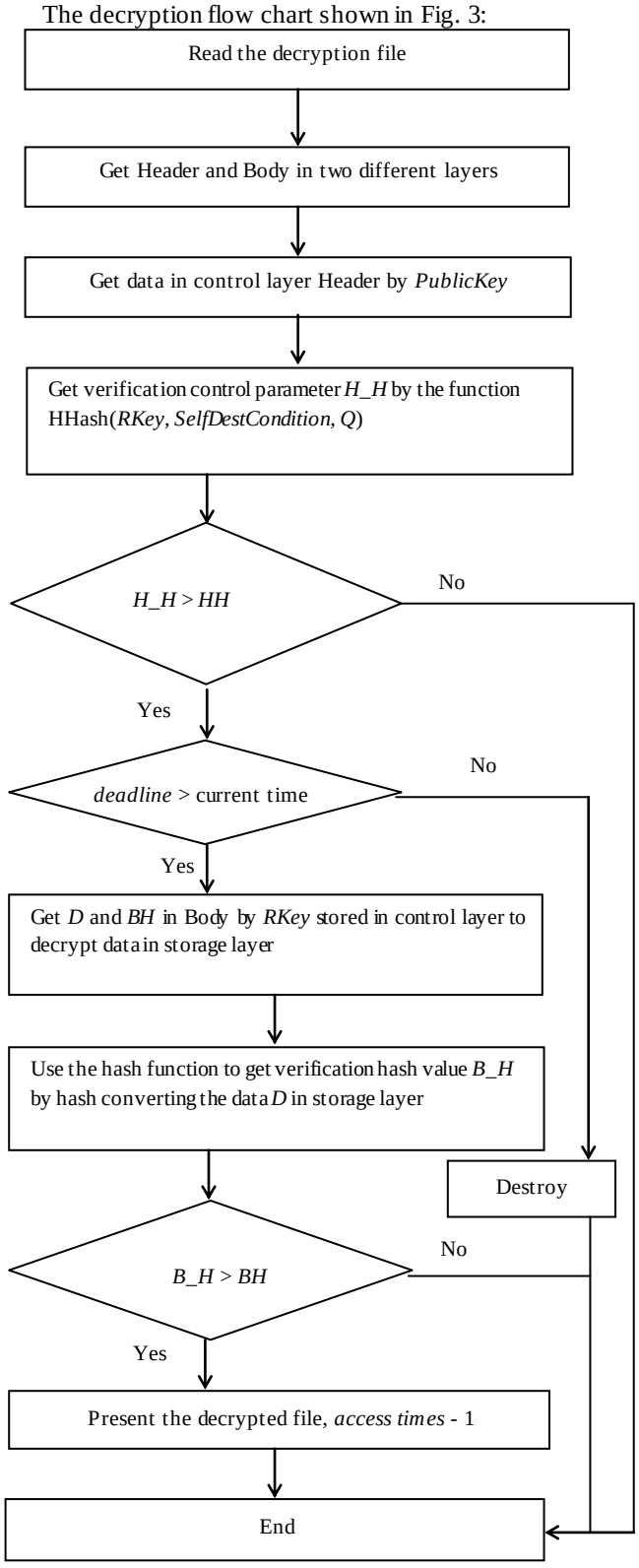

Figure 3. Decryption flow chart

Decryption process isn't a simple inverse process of encryption but a process of validation and decryption:

- (1) Read the data prepared to be decrypted which is visible in Fig. 1;
- (2) Get the control layer data Headers and the storage layer data Body from file;
- (3) Use asymmetric encryption algorithm to decrypt control layer data by the *PublicKey*;
- (4) Use a hash function to get verification control parameter *H_H* by hash converting the combination of *RKey*, *SelfDestCondition* and destruction strategy *Q* in control layer;
- (5) Compare *H_H* generated from (4) to *HH* stored in control layer. If different, it means the data of control layer may be tampered with and then stop the decryption process. If they are the same string, continue;
- (6) Compare the deadline of document in *SelfDestCondition* to the current time. If it is not more than the current time, it means the usage time of document is due and then call an appropriate way to destroy file by checking the destruction strategy *Q* in control layer; Else continue;
- (7) Use symmetric encryption algorithm to decrypt the data in storage layer to get *D* and its hash value *BH* in Body by getting the storage key *RKey* in control layer;
- (8) Use the same hash function to get verification hash value *B_H* by hash converting the data *D* in storage layer;
- (9) Compare the *B_H* generated from (8) to the *BH* stored in storage layer. If different, it means the data *D* in storage layer and then stop the decryption process; If same, the access times in *SelfDestCondition* minus 1 and the decryption process is finished. The data *D* is presented to the terminal users;
- (10) When the terminal users close the decryption document after access, it judges whether the int value greater than 0 according to the *access times* in *SelfDestCondition*. If greater, it means this document can be used next time and closed directly; Else it would call an appropriate way to destroy encryption file by checking the destruction strategy *Q* in control layer.

III. SECURITY ANALYSIS OF MODEL

A. Time limitation : deadline

Model in this paper judges whether the document file is due by comparing deadline in *SelfDestCondition* to the current time. However, as researchers all know, regardless of the operating system clock or hardware clock in mainboard, time in the terminal system would be changed by the terminal user so that they could bypass the destruction control. To resist such attacks, this model has connected to the Internet date/time server to keep synchronization. So it uses the world standard time rather than the terminal system as the current time. Internet date/time server using Daytime Protocol[17], to return a standard format of Daytime value after the terminal computer initiates the TCP connection request.

B. Transmission security

The benefit of that an encrypted file is divided into two parts(Header and Body) is that it can only get the data in Header by *PublicKey* and then get the *RKey* to decrypt the data in Body. Owing to the symmetric key generated in pseudo random process, obtaining the data in Body by direct Brute-force Attack is equivalent to the difficulty of encryption algorithm by ciphertext-only cracking symmetric encryption process (SymmetricEncrypt). The only way to attack the data in Header for *PublicKey* is the bypass attack. Otherwise, it uses Brute-force Attack, that is to say it's an equivalent problem of the encryption algorithm that ciphertext-only cracks asymmetric encryption process(AsymmetricEncrypt). This paper uses the Rijndael algorithm as the symmetric encryption algorithm. At the same time, the length of RKey is 256bit and the size of a block is 128bit so that it could comply with FIPS PUB 197 on National Secret level data encryption requirements[18]. Using the RSA as the symmetric encryption algorithm conforming to the key length requirements of the CA (Certificate Authority) center security level in the SET (Secure Electronic Transaction) protocol.

This model has realized double hash transformation. Even if encrypted file got by the man-in-the-middle, verification hash *H_H* and *B_H* generated by tamper data are different from *BH* and *HH* in Header and Body. So it would make the terminal file failure and send the encrypted file again. If someone wants to change the functions BHash() and HHash(), as researchers all know, there are many different hash functions and researches, so it is almost not possible to get the same data decrypted by different hash functions on same encrypted files. Double hash transformation further assures the tamper-proof of encrypted files and strengthens the security of the data.

C. Completely self-destruction of data

When the due time of a file is greater than or equal to the current time, prototype system would destroy the data. Deleting a file in our file system does not mean that its data is destroyed but make a "delete" tag on the file to tell the systemthis space could be covered.

In order to prevent the attacker to recover the deleted files in the file system by technical means, it must call safe destruction strategy to clean up and destroy the mapping file content rather than simply deletion. United States Department of Defense recommends a destruction method aimed at destroying the information written in the media: Use a character, radix-minus-one complement of the character and a random character in turn to cover all addressable areas in media[19]. Based on the principle of the method, the prototype system introduces the DOD 22-5220 - M standard as the security destruction strategy to realize destroying the file data after deadline or access times in file system. The specific algorithm process is as follows:

- (1)Use 0x00 to cover each corresponding cluster in the mapping file;
- (2)Use radix-minus-one complement of 0x00 which is 0xFF to cover the clusters again;
- (3)Use a random character generated by pseudorandom function to cover clusters for the last time.

The above three steps is a wipe. When the three steps all succeed, the encrypted files in file system would be deleted completely. This model could set one time of wipe, three times of wipe or seven times of wipe up to users' decisions. The file data destroyed after the security strategy *Q* could not be recovered again by simple means because the file data stored in the clusters (or blocks if in a distributed file system) has been covered completely although the index could be recovered. So attackers would not rebuild the deleted data and they could only get meaningless random character list in the last process (3) of covering.

IV. SYSTEM DESIGN

A. Pseudo code design

This section makes an example of Windows system to explain how to add files self-destruction system so it can realize the function of files self-destruction. Researchers write codes of the prototype system designed in the above self-destruction layer in the integrated development environment of Microsoft Visual Studio.Net. Then compile it into a Dynamic Link Library form. There are three functions of self-destruction system design:

- To encrypt the source files;
- To decrypt the encrypted files;
- To destroy a encrypted file completely after its deadline;

The designed pseudo code of the model are defined as follows:

```
Public Class File
    Header as Byte () = AsymmetricEncrypt (Serialize (Header), PublicKey)
    Body as Byte () = SymmetricEncrypt (Serialize (Body), Header. RKey)
End Class

Public Class Header
    SelfDestCondition as Deadline+Times+Q
    RKey as Byte () = Random()
    HH as Byte () = HHash (SelfDestCondition+RKey)
End Class

Public Class Body
    D as Stream
    BH as Byte() = BHash (D)
End Class
```

Among them, the AsymmetricEncrypt uses RSA 2048, SymmetricEncrypt uses Rijndael while both HHash and BHash use MD5[20]. The dynamic link library only expose two interfaces to external: one is to transform the source file data for the encrypted file and the other is to decrypt encrypted file data to the source data. The namespaces and interfaces are as follows:

```
Public Shared Function SelfDestructingData.DataToFile (Data as Stream, SelfDestCondition as Deadline+Times+Q, PrivateKey as Byte ()) as Filestream

Public Shared Function SelfDestructingData.FileToData (Filepath as String, PublicKey as Byte ()) as XMLStream
```

Among them, SelfDestructingData.DataToFile is provided for senders in the system while SelfDestructingData.FileToData is for receivers.

B. Experimental data

In this paper, according to existing test methods and combining with the experimental feasibility analysis of this system, it is added time overhead code in the source code at the same time of realizing the prototype system. Because the electronic document(doc, docx, txt .etc) has good observability, it takes a method of creating 10 new textfiles with special sizes and every test of given size doing 100 times. Then take the average time of encryption and decryption as time overhead while obtain the contrast of sizes before and after encryption as space overhead.

1) Time overhead: Due to the features of symmetric encryption and asymmetric encryption algorithm, compared to its encryption and decryption in asymmetric encryption, using symmetric encryption algorithm to encrypt the Body saves much more time. Especially when the file is bigger, symmetric encryption algorithm efficiency is much higher than asymmetric encryption algorithm in encrypting or decrypting the body. It creates 10 files of txt with the size of 1M/2M…/10M, encryption and decryption overhead are shown in Fig. 4 as below:

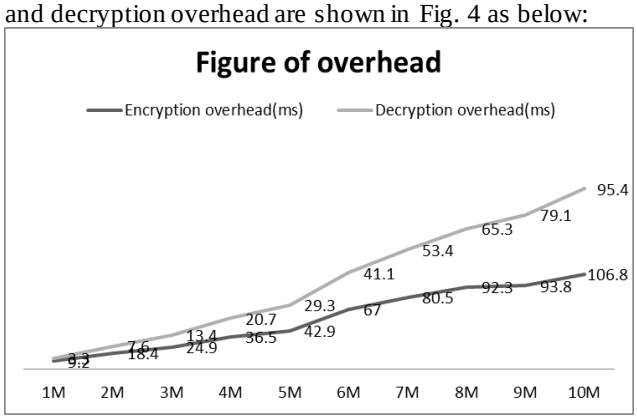

Figure 4. Encryption and decryption overhead

2) Space overhead: At space overhead, this model add the header and hash value *HH* in the header, so the encrypted files increase 1KB of size. In order to better highlight the contrast, researchers list 5 types of size shown in the TABLE I as below:

TABLE I. SPACE OVERHEAD

Size of source files(KB)	1	10	100	1024	10240
Size after encryption(KB)	2	11	101	1025	10241
Size of increase(KB)	1	1	1	1	1
Proportion	0.5	0.0909	0.0099	0.00098	9.76E-05

V. CONTRAST TO THE EXISTING TECHNOLOGIES

A. Functional contrast

Compared with the existing lifecycle control methods of documents, method in this paper adds time limit (deadline), access times and destruction strategy so that to realize the active and effective control of data in lifecycle

and completely destruction of documents after lifecycle. The realization of this method is simpler than that of Xiong[11-13], and the algorithms have good commonality in practice. The method of passive control from Yue[14] is converted to active setting control conditions of deadline and access times while this method does not need Lagrange theorem for encrypting and checking the documents every 8 hours. Embodied in the following four points:

- It realizes hash conversion of files Body data and verification conditions in Header by double hash. Multiple security mechanisms ensure the ciphertext data security. The *RKey* of symmetric encryption is encapsulated in the Header and then use the *PrivateKey* of asymmetric encryption encrypts the Header to prevent the *RKey* lost or stolen which would lead to data leaks;
- In the process of encryption, considering usage standard of files data, it imports the time limit(*deadline*), *access times* and destruction strategy *Q* in Header. So that it could realize effective control of files and avoid receivers from accessing data for an indefinite period or any times;
- If detecting that the usage time or access times of files is over, it invoke the destruction strategy in destruction layer automatically to destroy files. According to security level of files, destruction strategy is divided into three types: one time, three times and seven times in a random sequence coverage(wipe);
- After reaching the judgment of conditions, it automatically destroyed itself without intervention of senders or the trusted third ones. It could avoid leaks, deleting of data caused by accidents and to hedge the risks of man-made leaks.

B. Performance contrast

Because of the composite design of symmetric encryption algorithm and asymmetric encryption algorithm in this method, it increases some time overhead. However, compared with the time overhead of common files opening, the increasement of time overhead is in an acceptable range for most users.

In space overhead, this method increases the size of 1KB in files. Compared to a file with the size of 1M, the increased size after encryption is only a millesimal.

Compared to the method in existing files self-destruction paper, this method does not need trusted servers and trusted third party and greatly reduce handling stress in the server for encryption and decryption. Because of its simple encryption process and low requirements of the machine performance, researchers could move destruction layer to personal terminal and doing encryption or decryption in our terminal. This method is suitable for most format of current electronic files(doc, docx, txt, etc) so that it would not change the habits of reading. The encrypted files make "-protected" as suffix.

C. Contrast of key complexity

In the papers of Xiong's[11-13] and Yue's[14], the keys are drawn from ciphertext and stored in DHT. The keys need to transfer through various channels, some are trusted and some are not. If attackers get enough weights of keys, then the keys would be cracked. The complexity of the keys is affected by the length of ciphertext. That's to say the randomness itself of random drawing keys is very small and cracking cost is greatly reduced if cipher text itself is very short(sometimes documents likes xml is very small). Compared with the former ones, the method presented in this paper assumes that encryption is in the case of the same key length, then make time limit as a part of Body in the second hash process. In a 64 - bit system, the possibility of cracking is $1/2^{64}$ whether the length of ciphertext nor *PublicKey*(*PrivaryKey*).

VI. CONCLUSIONS

THE NATURAL SCIENCE FOUNDATION OF JIANGSU PROVINCE(GRANT NO. BK20130852)

This paper is based on a common phenomenon of shared file leaks in our real life and it puts forward a control method of data lifecycle to reduce the leak risk of sharing files data. Especially in a cloud environment for workers often need to share files with others or develop programs collaboratively, they should pay more attention to the vulnerability and potential safety hazard of plaintext as early as possible, then take relevant measures. The method put forward in this paper is still based on traditional cryptography theory, but increases security through multiple fine-grained encryption. Follow-up work will focus on the researches of the related technology that is more suitable for the distributed environment, such as Peer-to-Peer, improvement of Distributed Hash Table, etc. As a result, Researchers hope to extend the scheme to the data protection in cloud.

REFERENCES

[1] Quan Chen, Qianni Deng.Cloud Computing and its Key Techniques[J]. Journal of Computer Applications, 2009,29(9):2562-2567.

[2] Balinsky H Y, Simske S J. Differential access for publicly-posted composite documents with multiple workflow participants[C]//Proceedings of the 10th ACM symposium on Document engineering. ACM, 2010: 115-124.

[3] Xu D, Luo F, Gao L, et al. Fine-grained document sharing using attribute-based encryption in cloud servers[C]// Innovative Computing Technology (INTECH), 2013 Third International Conference on. IEEE, 2013:65-70.

[4] Li M, Yu S, Zheng Y, et al. Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption[J]. IEEE Transactions on Parallel & Distributed Systems, 2013, 24(1):131-143.

[5] Emura K, Miyaji A, Nomura A, et al. A Ciphertext-Policy Attribute-Based Encryption Scheme With Constant Ciphertext Length[J]. Information Security Practice & Experience, 2009, 5451(1):46-59.

[6] Zavattoni E, Perez L J D, Mitsunari S, et al. Software Implementation of an Attribute-Based Encryption Scheme[J]. IEEE Transactions on Computers, 2015, 64:1429-1441.

[7] Green M, Ateniese G. Identity-Based Proxy Re-encryption[M] // Applied Cryptography and Network Security. Springer Berlin Heidelberg, 2007:288-306.

[8] PublicKey Cryptography PKC. Public-Key Cryptography -- PKC 2014[J]. Springer Berlin, 2014.

[9] Weng J, Yang Y, Tang Q, et al. Efficient Conditional Proxy Re-Encryption With Chosen Cipher Text Security[J]. International Journal of Network Security & Its Applications, 2014, 2(4):104 - 110.

[10] Jing Xu, Jianqiao Yu, Youdian Zhu.Research and implementation of data destruction technologies based on remote control [J]. Computer Engineering and Design,2008,29(9):2206-2208.

[11] Xiong J B, Yao Z Q, Jian-Feng M A, et al. A Secure Self-Destruction Scheme with IBE for the Internet Content Privacy[J]. Chinese Journal of Computers, 2014.

[12] Xiong J B, Yao Z Q, Jian-Feng M A, et al. A Secure Self-Destruction Scheme for Composite Documents with Attribute Based Encryption[J]. Acta Electronica Sinica, 2014, 42(2):366-376.

[13] Xiong J, Yao Z, Ma J, et al. A Secure Document Self-Destruction Scheme with Identity Based Encryption[C]// Intelligent Networking and Collaborative Systems (INCoS), 2013 5th International Conference on. IEEE, 2013:239-243.

[14] Fengshun Yue. Self-Destructing Scheme for Electronic Data in Cloud Computing[D].Central South University,2011.DOI:10.7666/d.y1917084.

[15] Lina Wang, Zhengwei Ren, Rongwei Yu, etc.A Data Assured Deletion Approach Adapted for Cloud Storage[J].Acta Electronica Sinica,2012,40(2):266-272.DOI:10.3969/j.issn.0372-2112.2012.02.010.

[16] Lingfang Zeng, Zhan Shi, Shengjie Xu, et al. Safevanish: An improved data self-destruction for protecting data privacy[C]//Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on. IEEE, 2010: 521-528.

[17] J.Postel, RFC 867: Daytime protocol(S), 1983

[18] FPUB NIST，FIPS PUB 197: Advanced Encryption Standard (AES) (S), 2001

[19] DOE DoD, NRC CIA, DoD 5220.22-M: National Industrial Security Program Operating Manual(S), 1995

[20] R. Rivest, RFC 1321: The MD5 Message-Digest Algorithm(S), 1992.