International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 05 | May-2016 www.irjet.net p-ISSN: 2395-0072

SECURE SENSITIVE DATA SHARING ON A BIG DATA AND CLOUD PLATFORM USING ADVANCED SECURITY SYSTEM

N.Mamtha[1], G.Dharani Devi[2], A. Abdul Rasheed[3]

[1]*Student, Department of Computer Applications, Valliammai Engineering College, Tamil Nadu, India*
[2]*Assistant Professor, Department of Computer Applications, Valliammai Engineering College, Tamil Nadu, India*
[3]*Professor & Head, Department of Computer Applications, Valliammai Engineering College, Tamil Nadu, India*

Abstract - ***The Users store vast amounts of sensitive data on a big data and cloud platform. Sharing sensitive data will help enterprises reduce the cost of providing users with personalized services and provide value-added data services. However, secure data sharing is problematic. This paper proposes a framework for secure sensitive data sharing on a big data platform, including secure data delivery, storage, usage, and destruction on a semi-trusted big data sharing platform. We present a proxy re-encryption algorithm based on heterogeneous ciphertext transformation and a user process protection method based on a virtual machine monitor, Twofish and Blowfish which provides support for the realization of system functions. The framework protects the security of users' sensitive data effectively and shares these data safely. At the same time, data owners retain complete control of their own data in a sound environment for modern Internet information security.***

Key Words: secure sharing; sensitive data; big data; Cloud; proxy re-encryption; Twofish; Blowfish; private space…

1. INTRODUCTION

With the rapid development of information digitization, massive amounts of structured, semi-structured, and unstructured data are generated quickly. By collecting, sorting, analyzing, and mining these data, an enterprise can obtain large amounts of individual users' sensitive data. These data not only meet the demands of the enterprise itself, but also provide services to other businesses if the data are stored on a big data platform. Traditional cloud storage merely stores plain text or encrypted data passively. Such data can be considered as "dead", because they are not involved in calculation. However, a big data platform allows the exchange of data (including sensitive data). It provides mass data storage and computational services. Computation services refer primarily to operations (such as encrypting data, conversion, or function encryption) on data used by participants, which can invigorate "dead" data. An example of such an application is shown in Fig. 1 to illustrate the flow process of sensitive data on such a platform.

In Fig.1, we consider user's preferences as sensitive data. When Alice submits a query (sportswear), the Search Engine Service Provider (SESP) first looks for Alice's preference on the big data platform. If the big data platform has collected and shared the user's personal preference information, "badminton", then the search engine returns personalized results (sportswear + badminton) to Alice.

When Alice sees her favourite badminton sportswear, she experiences a pleasant purchase. Consequently, this leads to a win- win situation. However, while data sharing increases enterprise assets, Internet insecurity and the potential of sensitive data leakage also create security issues for sensitive data sharing.

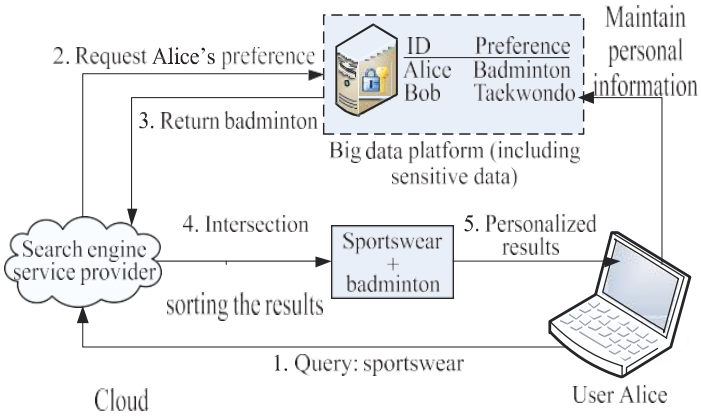

Fig. 1 Application of sensitive data (user's preferences).

Secure sensitive data sharing involves four primary safety factors. First, there are security issues when sensitive data are transmitted from a data owner's local server to a big data platform. Second, there can be sensitive data computing and storage security problems on the big data platform. Third, there are secure sensitive data use issues on the cloud platform. Fourth, there are issues involving secure data destruction. Some research institutions and scholars at home and abroad have made positive contributions to exploration and research aimed at solving these security problems.

Existing technologies have partially resolved data sharing and privacy protection issues from various perspectives, but they have not considered the entire process in the full data security life cycle. However, a big data platform is a complete system with multi- stakeholder involvement, and thus cannot tolerate any security breach resulting in sensitive data loss. In this paper, we analyze security issues involving the entire sensitive data sharing life cycle and describe a

system model created to ensure secure sensitive data sharing on a big data platform, to guarantee secure storage on the big data platform using Proxy Re-Encryption (PRE) technology, and to ensure secure use of sensitive data sharing using a private space process based on a Virtual Machine Monitor (VMM). Then, a security plug-in and a data self-destruction mechanism help to alleviate user concern regarding sensitive personal information leakage.

2. RELATED WORK

In this section, we focus on previous work on relevant topics such as encryption, access control, trusted computing, and data security destruction technology in a cloud storage environment.

A.Data encryption and access control of cloud storage

Fig 2 Twofish is a 128-bit block cipher that accepts a variable-length key up to 256 bits. The cipher is a 16-round Feistel network with a bijective F function made up of four key-dependent 8-by-8-bit S-boxes, a fixed 4-by-4 maximum distance separable matrix over GF(28), a pseudo-Hadamard transform, bitwise rotations, and a carefully designed key schedule. A fully optimized implementation of Twofish encrypts on a Pentium Pro at 17.8 clock cycles per byte, and an 8-bit smart card implementation encrypts at 1820 clock cycles per byte. We have extensively cryptanalyzed Twofish; our best attack breaks 5 rounds with 222.5 chosen plaintexts and 251 effort.

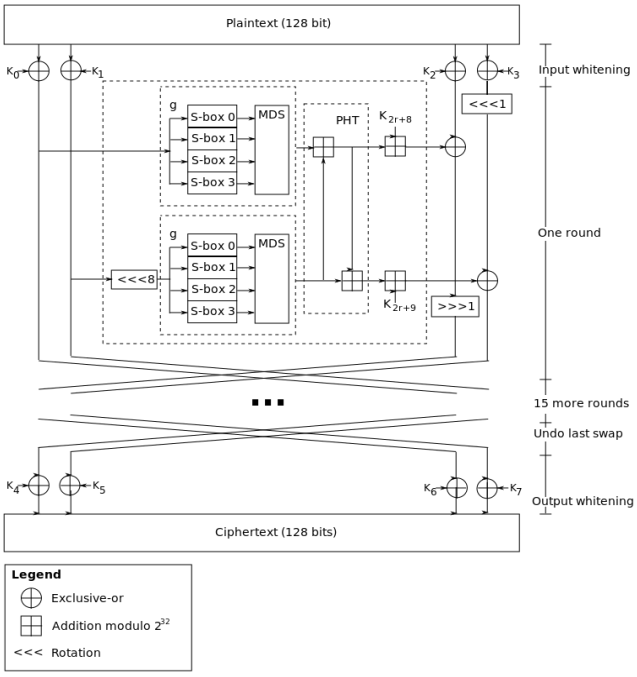

Fig. 2 The Twofish algorithm.

Blowfish is a symmetric-key block cipher, designed in 1993 by Bruce Schneier and included in a large number of cipher suites and encryption products. Blowfish provides a good encryption rate in software and no effective cryptanalysis of it has been found to date. However, the Advanced Encryption Standard (AES) now receives more attention. Schneier designed Blowfish as a general-purpose algorithm, intended as an alternative to the aging DES and free of the problems and constraints associated with other algorithms. At the time Blowfish was released, many other designs were proprietary, encumbered by patents or were commercial or government secrets.

Blowfish has a 64-bit block size and a variable key length from 32 bits up to 448 bits. It is a 16-round Feistel cipher and uses large key-dependent S-boxes. In structure it resembles CAST-128, which uses fixed S-boxes. The diagram to the left shows Blowfish's encryption routine. Each line represents 32 bits. There are five subkey-arrays: one 18-entry P-array (denoted as K in the diagram, to avoid confusion with the Plaintext) and four 256-entry S-boxes (S0, S1, S2 and S3).

Every round r consists of 4 actions: First, XOR the left half (L) of the data with the r th P-array entry, second, use the XORed data as input for Blowfish's F-function, third, XOR the F-function's output with the right half (R) of the data, and last, swap L and R.

The F-function splits the 32-bit input into four eight-bit quarters, and uses the quarters as input to the S-boxes. The S-boxes accept 8-bit input and produce 32-bit output. The outputs are added modulo 232 and XORed to produce the final 32-bit output Feg 3

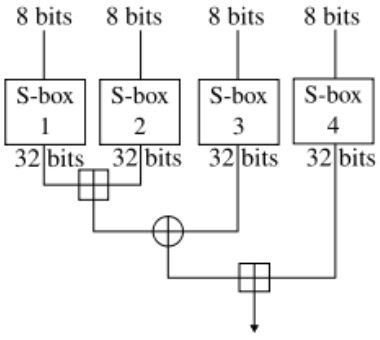

Fig 3 The round function (Feistel function) of Blowfish

3. SYSTEMATIC FRAMEWORK FOR SECURE SENSITIVE DATA SHARING ON A BIG DATA PLATFORM

Issuing and renting sensitive data on a semi- trusted big data platform requires a data security mechanism. Building secure channels for a full sensitive data life cycle requires consideration of four aspects of safety problems: reliable submission, safe storage, riskless use, and secure

destruction. A systematic framework for secure sensitive data sharing on a big data platform is shown in Fig. 4.

A common and popular method of ensuring data submission security on a semi-trusted big data platform is to encrypt data before submitting data to the platform. Some operations (such as encryption, decryption, and authorization) are provided using a security plug-in. A cloud platform service provider (such as an SESP) using data on a big data platform ensures data security by downloading and using the security plug-in.

To ensure secure storage, we designed Heterogeneous Proxy Re-Encryption (H-PRE), which supports heterogeneous transformation from Identity- Based Encryption (IBE) to Public-Key Encryption (PKE). H-PRE is compatible with traditional cryptography. The intent is to transform the cipher data that the owner uploads into ciphertext that the data user can decrypt using his or her own private key.We assume that the cloud cannot be trusted, and that the decrypted clear text will leak users' private information. Therefore, we need to adopt process protection technology based on a VMM, through a trusted VMM layer, bypassing the guest operating system and providing data protection directly to the user process. The key management module of the VMM is used for storing public keys of the new register program group. When a program is running, the symmetric key at the bottom of the main program will be decrypted dynamically by the key management module. All applications of the public and symmetric keys are stored in the memory of the VMM.

The archive, replication, and backup mechanism of cloud storage create data redundancy, requiring the use of a suitable data destruction scheme to delete the user's private personal data. To achieve high security, we designed a lease-based mechanism to destroy private data and keys thoroughly in a controlled manner. Cleartext and keys exist nowhere in the cloud, after the lease expires.

The basic flow of the framework is as follows. First, enterprises that have individual users' sensitive information pre-set those service providers that need to share this sensitive information and then submit and store the corresponding encrypted data on a big data platform using the local security plug- in. Second, we need to perform the required operation with the submitted data using PRE on the big data platform. Then, cloud platform service providers who want to share the sensitive information download and decrypt the corresponding data in the private process space using the secure plug-in with sensitive privacy data running in that space. Last, we use a secure mechanism to destroy used data still stored temporarily in the cloud.

In short, the framework protects the security of the full sensitive data life cycle effectively. Meanwhile, data owners have complete control over their own data. Next, we discuss the most critical PRE algorithm based on heterogeneous cipher-text transformation and user process protection methods using the VMM.

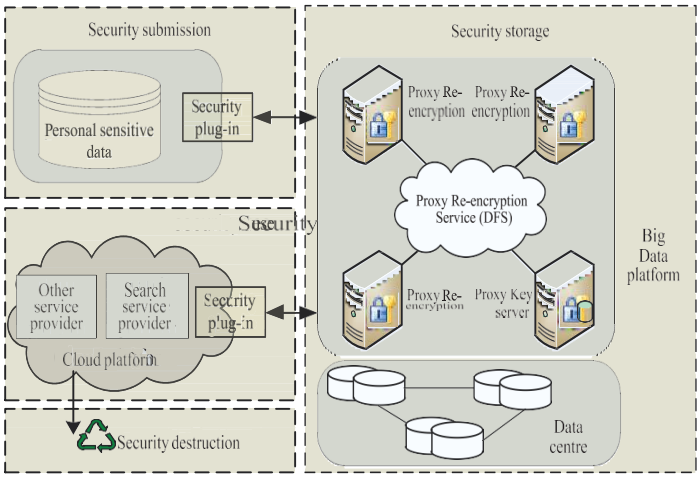

Fig. 4 Systematic framework

3.1 SECURE SUBMISSION AND STORAGE OF SENSITIVE DATA BASED ON PRE

PRE involves three types of algorithm, traditional identity-based encryption (including SetupIBE, KeyGenIBE, EncIBE, and DecIBE), re-encryption (including KeyGenRE, ReEnc, and ReDec functions), and the last one is the traditional public key cryptosystems (including KeyGenPKE, EncPKE, and DecPKE). The basic H-PRE process is simple. The data owner encrypts sensitive data using a local security plug-in and then uploads the encrypted data to a big data platform. The data are transformed into the ciphertext that can be decrypted by a specified user after PRE services. If an SESP is the specified user, then the SESP can decrypt the data using its own private key to obtain the corresponding clear text.

We complete the following steps to implement the H-PRE algorithm.

1. SetupIBE.k/: Input security parameters k, generate randomly a primary security parameter mk, calculate the system parameter set params using a bilinear map and hash function.
2. KeyGenIBE.mk, params, id/: When the user requests the private key from the key generation center, the

key generation center obtains the legal identity (id) of the user and generates the public and private keys(pkid, skid) for the user using params and mk.

3. KeyGenPKE.params/: When a user submits a request, the key management center not only generates the identity-based public and private keys, but also generates the public and private keys of the traditional public key system (pk0 , sk0).
4. EncIBE.pkid; skid; params; m/: When the user encrypts data, the data owner encrypts the clear-text (m) into the ciphertext (c D .c1; c2/) using the user's own (pkid, skid) and a random number (r 2 RZ*).
5. KeyGenRE.skidi ; sk0idi; pk0idj; params/: When the data owner (user i) grants user j permissions, using skidi , sk0idi , and pk0idj , user i computes the PRE key (rkidi!idj), completing the transformation from user I to user j .
6. ReEnc.ci ; rkidi -idj ; params/: This process is executed transparently on the big data platform. The function re-encrypts the ciphertext that user i encrypted into ciphertext that user j can decrypt. It inputs ci .ci D.ci1; ci2//, the PRE key (rkidi -idj), and related system parameters, and then the big data platform computes and outputs the PRE ciphertext (cj D .cj1; cj 2/).
7. DecPKE.cj ; sk0idj; params/: This is a function for decrypting the PRE ciphertext. After receiving the PRE ciphertext (cj D .cj1; cj 2/) from the proxy server of the big data platform, user j determines the clear-text (m0 D m) of the data using his or her own sk0idj .

3.2 THE SUBMISSION, STORAGE, AND EXTRACTION OPERATIONS OF SYSTEM SENSITIVE DATA

The data owner encrypts data locally, first using the Advanced Encryption Standard (AES) symmetric encryption algorithm to encrypt the submission data and then using the PRE algorithm to encrypt the symmetric key of the data. These results are all stored within the distributed data. In the meantime, if the data owner shares the sensitive data with other users, the data owner must authorize the sensitive data locally and generate the PRE key, which is stored in the authorization key server and transforms the original cipher using the PRE key. Then, PRE ciphertext, which can be encrypted by the (authorized) data users, is generated. If the data user wants to use the data on the big data platform, the data user will send data requests to the platform and then query whether there is corresponding data in the shared space. If such data exist, the data user accesses and downloads it. The operation on the big data platform is independent and transparent to users. Moreover, the computing resources of the big data platform are more powerful than those of the client. Hence, we can put PRE computational overhead on the big data platform to improve user experience. The PRE system includes data submission, storage (sharing), and data extraction operations.

A.DATA SUBMISSION AND STORAGE (SHARING) OPERATIONS

After receiving the data uploading request, the Web browser invokes the security plug-in and provides data uploading services for the data owner, in accordance with the following detailed steps. The browser

(1) Reads the data files uploaded by the data owner, generates randomly an AES transparent encryption key (Symmetric Encryption Key, SEK), and then use the AES algorithm to encrypt the data files;

(2) Uses the PRE algorithm to encrypt the SEK and store the data ciphertext and SEK ciphertext in the data centers;

(3) Identifies from the data owner the users designated to share the data;

(4) Uses the security plug-in to read the private key of the data owner and obtain the data user's public key from the big data platform;

(5) Uses the security plug-in to generate the corresponding PRE key using the EncIBE function and to upload the PRE key to the authorization key server of the big data platform; and

(6) Re-encrypts the data using the ReEnc function on the big data platform, thereby generating PRE ciphertext.

B. DATA EXTRACTION OPERATIONS

After receiving the data download request, the Web browser invokes the security plug-in and provides data download services for the data user, in accordance with the following detailed steps. The browser

(1) Queries whether there is authorization for the data user on the PRE server of the big data platform, and if an authorization is in effect, proceeds to Step (2);

(2) uses the download plug-ins to send data download requests to the big data platform, which then finds PRE ciphertext data in the data center;

(3) Pushes the PRE ciphertext to the secure data plug-in on the big data platform;

(4) Invokes a data user's download plug-in to read the user's private key and prepares to decrypt data;

(5) Invokes a data user's download plug-in to decrypt received SEK ciphertext using the DecPKE function and obtain the AES symmetric key; and

(6) Permits the data user to decrypt the data ciphertext using the AES symmetric key to obtain the required clear text.

The data extraction operation is put into the private space of a user process by the secure plug-in, a prerequisite for secure use of sensitive data.

4 SECURE USE OF SENSITIVE DATA ON VMM

4.1 THE PRIVATE SPACE OF A USER PROCESS BASED ON A VMM

To ensure secure running of an application in the cloud, we use the private space of a user process based on a VMM. We assume that some enterprise (such as an SESP) rents Infrastructure as a Service (IaaS) to complete some business. The business process needs to extract sensitive personal data on the big data platform. We call the protected program that extracts sensitive data from the big data platform a sensitive process. A threat model of a sensitive process on a cloud platform is shown in Fig. 5. A sensitive process must prevent threats from a management VMM and an unreliable operating system layer below it. Rented bottom hardware uses the TPM mode, ensuring that the VMM is trusted. In this case, the key management mechanism of the renter (such as an SESP) must build this relationship based on trusting a VMM, ensuring safe operation under the unreliable operating system. The introduction of virtualization and trusted computing technology ensures that service provider applications and a secure plug-in run in the process private space. This mode can protect the privacy of sensitive data and avoid interference from external programs, even the operating system. A safe operation process is shown in Fig. 6.

With PRE ciphertext calculated on a big data platform extracted onto a cloud platform, private memory space of processes on the cloud platform can guarantee data security in memory and on the Hard Disk Drive (HDD). First, the VMM provides private memory space for specifying a VMprocess.

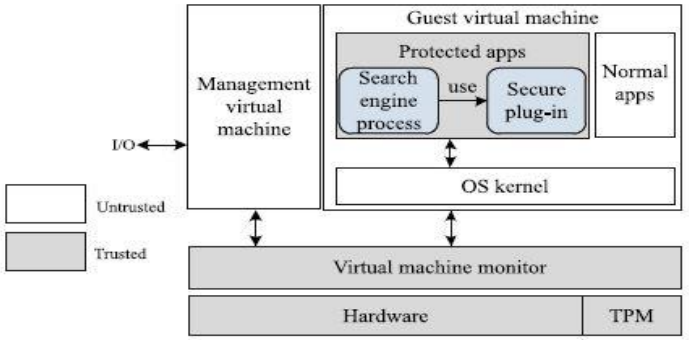

Fig. 5 Threat model of a sensitive process in a cloud Platform.

The process runs in private memory space whose memory cannot be accessed by the operating system or other applications. The method of memory isolation ensures data privacy and security in the memory. Furthermore, the data used and stored on disk is ciphertext. The VMM decrypts or encrypts when reading or writing data, respectively. As a result, a combination of these two measures can be protected using the VMM, whether the user program runs in memory or is stored on disk.

4.2 SECURE USE OF SYSTEM SENSITIVE DATA

We use process protection technology based on a VMM, through a trusted VMM layer, and bypass the guest operating system, providing data protection directly to the user process. To protect data security in the process of interaction on the cloud platform, the following steps must be completed.

(1) ESTABLISHING A CREDIBLE ENVIRONMENT AND CHANNELS

During the booting process, the cloud platform needs to measure startup software through trusted computing technology. Therefore, cloud users (SESPs) must ensure the integrity of the VMM, that is to say, cloud users must ensure that the VMM is trusted. After the booting process, the cloud server will store Basic Input/output System (BIOS), Grand Unified Boot loader (GRUB), and VMM measurements in the Platform Configuration Register (PCR) of the TPM chip, and then send a remote verification to the user to ensure the trust relationship between them. The SESP must establish a reliable channel with the VMM in the cloud, and then receive sensitive data safely from the big data platform.

The remote attestation and hand shaking protocol between the SESP and the VMM in the cloud is shown in Fig. 7. In fact, the VMM responds to the request at the cloud server end.

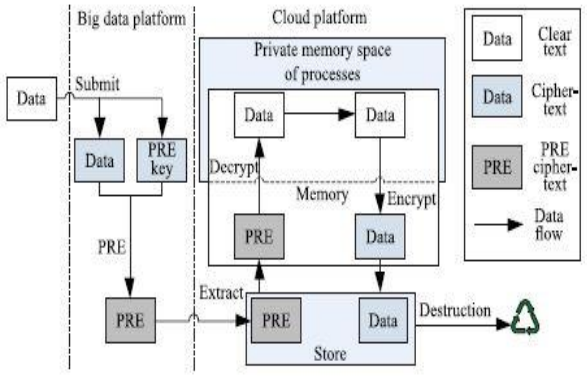

Fig. 6 Safe operation process.

First, the SESP sends an integrality request to the cloud server, including the SESP public key (PKid) and timestamp (TS). Second, the VMM generates a session key (Ksess) and computes the hashed value of TS, PKid, and Ksess using Secure Hash Algorithm (SHA1). Then, the VMM calls the TPM quote instruction and passes the hashed value and PCR as arguments to obtain the testimony (Quote) using the TPM private key signature. The VMM uses PKid to encrypt Ksess and then sends Ksess, Quote, and a Certification Authority (CA) certificate to the other side. The SESP verifies the value of TS, PKid, and Ksess after receiving this information. If the values are consistent, the communications are secure. As a result, both sides of the communications determine a session key. In the future, both sides of communication will be encrypted using the session key.

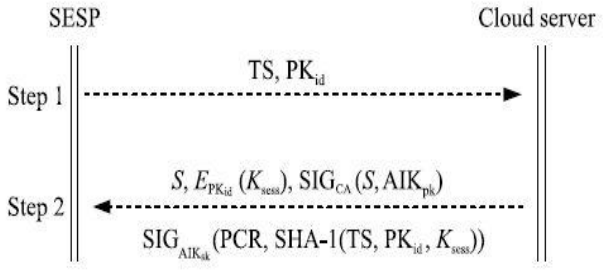

Fig. 7 Remote attestation and hand shaking protocol between the SESP and the VMM in the cloud.

(2) DATA UPLOAD AND EXTRACTION

The cloud users (SESP) extract the sensitive data from the big data platform through retrieval. We assume that the cloud is untrusted. The uploaded executable application and data must be encrypted before the SESP uses the cloud. The upload and extract protocol of the data is shown in Fig. 6. In Fig. 8, the SESP generates the AES symmetric key and a pair of asymmetric keys (PKapp, SKapp) using the tools, encrypts the executable files and data files using the AES symmetric key, and encrypts the AES key by the asymmetric keys, which are attached at the end of the application files. The data obtained from the big data platform are PRE ciphertext, which can be decrypted during runtime. The command format of the new program must be identified when registering the program. The user encrypts the PKid, registration command, application name, public key (PKapp), and predetermined lease using Ksess, and then sends them to the VMM. Finally, encrypted executable files and data files are uploaded to the cloud server.

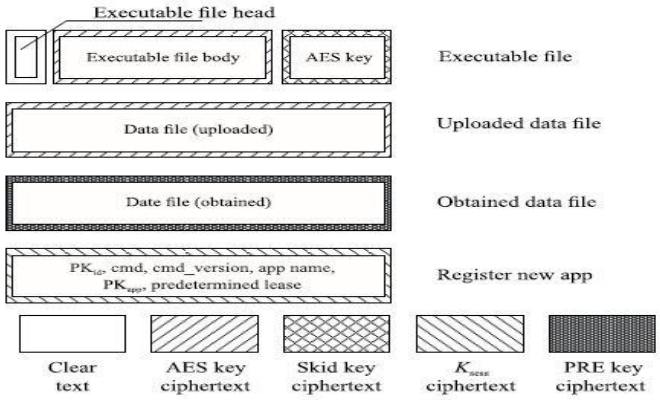

Fig. 8 Upload and extract protocol of the data.

(3) PROGRAM EXECUTION

In the process of application execution on the cloud platform, dynamic data protection and encryption are similar to the protection of process memory space, as shown in Fig. 6. During process execution, the occupied memory process cannot be accessed by other processes and operating systems. The VMM serves as the bridge of data exchange between the operating system and the user process. When the OS copies the data from the user memory space, the VMM, not the operating system, performs the copying operation, because the operating system lacks read and write privileges. When the data are copied into the private memory space of the process, the VMM decrypts the data using the corresponding AES symmetric key. Thus, the data can be computed normally. Conversely, when the data in the private memory space of the process are copied to the outside, the VMM encrypts the data using the corresponding AES symmetric key. Hence, the user data stored on disk is in ciphertext form. In a word, in the private space of a user process, the security plug-in decrypts PRE data from the big data platform and the VMM decrypts data from the cloud

user (SESP). The generated data are encrypted when the user process is completed, and then the data is destroyed according to the terms of the lease. Therefore, the private space of the user process acts as a balance point of the security mechanism between the data owner and user, benefiting both while preventing sensitive information leakage.

5. CONCLUSIONS

In summary, we proposed a systematic framework of secure sharing of sensitive data on big data platform, which ensures secure submission and storage of sensitive data based on the heterogeneous proxy re-encryption algorithm, and guarantees secure use of clear text in the cloud platform by the private space of user process based on the VMM. The proposed framework well protects the security of users' sensitive data. At the same time the data owners have the complete control of their own data, which is a feasible solution to balance the benefits of involved parties under the semi-trusted conditions. In the future, we will optimize the heterogeneous proxy re-encryption algorithm, and further improve the efficiency of encryption. In addition, reducing the overhead of the interaction among involved parties is also an important future work.

REFERENCES

[1] https://en.wikipedia.org/wiki/Twofish

[2] J. Li, G. Zhao, X. Chen, D. Xie, C. Rong, W. Li, L. Tang, and Y. Tang, Fine-grained data access control systems with user accountability in cloud computing, in Proc. 2nd Int. Conf. on Cloud Computing, Indianapolis, USA, 2010, pp. 89–96.

[3] S. Ananthi, M.S. Sendil, and S. Karthik, Privacy preserving keyword search over encrypted cloud data, in Proc. 1st Advances in Computing and Communications, Kochi, India, 2011, pp. 480–487.

[4] H. Hu, J. Xu, C. Ren, and B. Choi, Processing private queries over untrusted data cloud through privacy homomorphism, in Proc. 27th IEEE Int. Conf. on Data Engineering, Hannover, Germany, 2011, pp. 601–612.

[5] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, Privacy-preserving multi-keyword ranked search over encrypted cloud data, in Proc. 30th IEEE INFOCOM, Shanghai, China, 2011, pp. 829–837.

[6] C. Hong, M. Zhang, and D. Feng, AB-ACCS: A cryptographic access control scheme for cloud storage, (in Chinese), Journal of Computer Research and Development, vol. 47, no. 1, pp. 259–265, 2010.

[7] X. Chen, T. Garfinkel, E. C. Lewis, and B. Spasojevic, Overshadow: A virtualization-based approach to retrofitting protection in commodity operating systems, in Proc. 13th Int. Conf. on Architectural Support for Programming Languages and Operating Systems, Seattle, USA, 2008, pp. 2–13.

[8] J. Yang and K. G. Shin, Using hypervisor to provide data secrecy for user applications on a per-page basis, in Proc. 4th Int. Conf. on Virtual Execution Environments, Seattle, USA, 2008, pp. 71–80.

[9] H. Chen, F. Zhang, C. Chen, Z. Yang, R. Chen, B. Zang, W. Mao, H. Chen, F. Zhang, C. Chen, et al., Tamperresistant execution in an untrusted operating system using a virtual machine monitor, Technical Report, Parallel Processing Institute, Fudan University, FDUPPITR-2007-0801, 2007.

[10] P. Dewan, D. Durham, H. Khosravi, M. Long, and G. Nagabhushan, A hypervisor-based system for protecting software runtime memory and persistent storage, in Proc. the 2008 Spring Simulation Multiconference, Ottawa, Canada, 2008, pp. 828–835.