2020 19th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)

Research on the Application of Transparent Encryption in Distributed File System HDFS

Fei Jiang, Zhenggao Pan, Qingzhao Li
School of Information Engineering Suzhou
Suzhou University
Suzhou City Anhui province, Peoples R China,234000
E-mail:feiwuhan@126.com
68080507@qq.com
1279494538@qq.com

Linsheng Huang*, Dongyan Zhang
National Engineering Research Center for Agro-Ecological Big Data Analysis & Application, Anhui University
Anhui University, Hefei, Peoples R China, 230601
E-mail: linsheng0808@163.com
zhangdy@ahu.edu.cn

***Abstract*—More and more companies are turning to Hadoop to store and deal with their most valuable data, the protection and management of data begin to face increasing security risks. As a data protection technology closely related to big data platform, HDFS transparent encryption technology has certain practical value. In this paper, the HDFS transparent encryption technology is studied, the implementation principle of HDFS transparent encryption technology is introduced, the security and performance of the HDFS transparent encryption technology are analyzed, and the encryption technologies at different levels of the big data platform are compared. Research shows that HDFS transparent encryption technology has the advantages of high performance, transparent application and easy deployment, but there are still potential security problems in the application process.**

Keywords-transparent encryption; big data security; distributed file system

I. INTRODUCTION

With the rapid development of the Internet and the continuous in-depth exploration of technology, a large amount of data is being generated. Traditional solutions cannot meet the storage and processing needs of massive data, so Hadoop large data processing technology emerged as the times require. Hadoop [1] is a Map at Google by Doug Cutting, the founder of Apache's Lucene Full Text Retrieval Project, and his team Hadoop technology, created on the basis of components such as Reduce [2] and GoogleFS [3], has been widely used by many Internet companies such as Yahoo, Facebook, Baidu, Ali, Tencent, and has become a generally accepted framework for large data analysis and storage.

The Hadoop platform was initially developed without security considerations, when Hadoop's use cases were all about how to manage a large amount of public Web data without considering data confidentiality and complex internal rights management. As originally conceived by Hadoop, it assumes that clusters are always in a trusted environment composed of trusted, collaborative computers used by trusted users. However, as more and more data, users, and applications access large data platform clusters, and more and more enterprises store private information on the Hadoop cluster, Hadoop security issues become particularly prominent. In 2009, the Yahoo team published a paper [4] choosing Kerberos as the authentication scheme for Hadoop platform, which provides a solid foundation for the security management scheme of Hadoop large data platform. In addition to Yahoo and other companies, many network security researchers at home and abroad are doing Hadoop security research: methods such as a trustworthy equalization [5-7] ,mixed encryption algorithm for Hadoop environment[8, 9], triple data encryption algorithm [10], parallel encryption [11] and other schemes are proposed. Hadoop also has five security projects: Apache Sentry[12], Apache RecordService[13], Apache Ranger[14], Apache Knox Gateway[15], and Apache Rhino[16].Among them, Apache Sentry is an open source Hadoop component released by Cloudera that provides granularity, role-based authorization, and a multi-tenant management model; RecordService is a new Hadoop security layer designed to provide secure access to data and analysis engines running on Hadoop, although the Sentry security components previously provided by Cloudera support access to the defined control rights. RecordService supplements Sentry by controlling access to row and column levels; Apache Ranger is an open source Hadoop component released by Hortonworks, which solves the current decentralized security management of each service on the Hadoop platform , and creates a centralized and unified management interface for all services to provide access management, log auditing, etc. Apache Knox Gateway is a cluster boundary security management scheme proposed by Hortonworks. For the operation and maintenance team, only the internal cluster needs to be concerned, and no deployment details need to be disclosed. For the development team, centralized access to Hadoop-related services through the boundary network management greatly simplifies the complexity of development; Apache Rhino, an open source project led by Intel, is dedicated to providing enhanced security for Hadoop's ecological components and data.

Encryption is the most direct and effective means of data protection and the last defense line for data protection. The Apache Rhino project proposes HDFS transparent encryption technology that can be used to secure Hadoop data. Transparent encryption [17, 18] means that for users, the encryption and decryption of data is automatic and will not affect their normal operation habits. With this technology, when users save data, the system will automatically encrypt the data. When users open or edit specified files, the system will automatically decrypt the encrypted files. The encrypted files are stored on the hard disk, and the clear text is stored in memory. Once they leave the environment, the data cannot be automatically

2473-3636/20/$31.00 ©2020 IEEE
DOI 10.1109/DCABES50732.2020.00011

decrypted, thus protecting the data. The effect of content back to HDFS transparent encryption has led to the concept of Encryption Zone, an encrypted area that is a special directory on HDFS, where written content is encrypted transparently while reading the contents in the encrypted area space is decrypted transparently. Encryption and decryption do not require changes to user application code. This encryption is also end-to-end, meaning that the data can only be encrypted and decrypted by Client. At present, foreign data platforms such as Cloudera and IBM started to apply this encryption technology; however, China has not yet achieved enterprise application in the mainstream large data platforms, and the application of HDFS transparent encryption technology needs to be further strengthened.

II. How HDFS Transparent Encryption works

A. Introduction to HDFS

Distributed File System HDFS is the core component of Hadoop Platform to implement massive data storage at the bottom of the platform. HDFS clusters involve three important roles: NameNode, DataNode, and Client. NameNode is the controller of the entire cluster, which manages the HDFS namespace, cluster information, block replication, and so on. NameNode stores HDFS metadata in memory, mainly including file information, block information corresponding to the file, and DataNode where each block is located. DataNode is the data storage node of the cluster and periodically reports all block information to NameNode. Client is an application that needs to obtain distributed file system files and is responsible for reading and writing cluster data.

File Writing: First Client sends a request for a file writer to NameNode; then NameNode returns information about DataNode available to Client based on the size of the file and the block configuration; and finally Client divides the file into blocks and writes the blocks into DataNode sequentially based on the received DataNode information.

File reading: Client first initiates a file read request to NameNode; NameNode then returns the information of the DataNode stored in the file; and Client finally reads the file information.

B. HDFS Transparent Encryption works

HDFS Transparent Encryption uses the Key Management Server KMS to generate and manage the following keys:

Encryption Zone Key (EZK): Each encrypted zone has one encrypted zone key s stored on the KMS server; NameNode does not know the encrypted zone key;

* Data Encryption Key (DEK): one data encryption key for each file; Client uses it to decommission data on HDFS; exposes KMS and Client, and keeps NameNode and DataNode private;

* Encrypted Data Encryption Key (EDEK): KMS uses EZK to encrypt DEK generated and stored on NameNode; KMS uses EZK to decrypt EDEK to obtain DEK.

Write data flow:

The system administrator first creates a directory as the encrypted zone EZ and an encrypted zone key EZK.

1. When Client writes a file to the encrypted area, Client first communicates with NameNode to confirm that the file is written; 2. After NameNode receives Client's request, it requests the encryption data encryption key EDEK from KMS;

3. After KMS receives the NameNode request, it retrieves the master key, chooses a numeric value as the DEK, encrypts the DEK via EZK, and sends the EDEK to NameNode in cipher.

4. NameNode saves the EDEK and sends the EDEK and available DataNode information to the Client;

5. Client sends EDEK to KMS after receiving EDEK and requests decryption of EDEK;

6. After KMS receives the request to decrypt the EDEK, it uses the encrypted area key EZK to decrypt the EDEK, generates the data encryption key DEK, and returns the DEK to Client;

7. Client uses DEK to encrypt data and stores it in an encrypted area (Client passes file ciphertext block by block to the corresponding DataNode in order, and DataNode, which receives the block, copies the block to other DataNode).

Reading data flow:

1. Client sends the file path to be read to NameNode;

2. NameNode gets the meta-information of the file (mainly the block's stored location information, EDEK) and returns it to Client;

3. Client requests KMS to decrypt EDEK;

4. KMS decrypts EDEK and returns DEK to Client;

5. Client finds the blocks of corresponding DataNode files one by one based on the returned information and performs data append and merge locally on the Client to decrypt the encrypted data using DEK to get the entire file.

III. Analysis of HDFS Transparent Encryption Technology

A. Security Analysis

HDFS transparent encryption technology involves three entities: KMS server, HDFS cluster (NameNode and DataNode), and Client. The security of encrypted zone data is analyzed when they are under attack.

1. Attacking KMS Server

Attackers attack KMS servers and can obtain EZK, EDEK, DEK. In a secure environment, the KMS/keystore administrator and HDFS administrator roles should be separated so that they do not have access to all encrypted data and all encrypted keys at the same time. An attacker cannot obtain HDFS ciphertext because he does not have HDFS access, which prevents the attacker from decrypting the ciphertext. Secondly, an attacker attacking the KMS server will destroy or delete the key, and Client will not be able to access the encrypted zone data properly.

2. Attacking HDFS clusters

The NameNode of the HDFS cluster has an EDEK and the DataNode node has text data. Although EDEK and cryptographic data can be obtained by attacking HDFS clusters, it is not possible to decrypt cryptographic data because DEK cannot be recovered without EZK.

3. Attack Client

An attacker can impersonate a user to obtain a password file allowed by HDFS access rights and an EDEK allowed by KMSACL access rules. An attacker can then obtain a DEK by interacting with KMS to recover the

ciphertext data that intersects the two. In security settings, this access to HDFS and KMS keys should be strictly controlled.

4. Network eavesdropping

1) Network eavesdropping between Client and KMS

Attackers can obtain EDEK, DEK by eavesdropping on the network between Client and KMS. If an attacker can decrypt part of the ciphertext in the encrypted area by obtaining the ciphertext data, there is a risk of network eavesdropping. It is recommended to configure TLS/SSL to enhance the security of network transmission.

2) Network eavesdropping between KMS and HDFS

Attackers eavesdropping on the network between KMS and HDFS can only obtain EDEK because there is no EZK and ciphertext data to recover encrypted zone data.

3) Network eavesdropping between Client and HDFS

Attackers eavesdropping on the network between Client and HDFS can only obtain encrypted data and EDEK, because DEK cannot be obtained without EZK, thus decrypting encrypted data.

In summary, HDFS transparent encryption has the following security features and risks.

HDFS transparent encryption has the following security features:

A) Both encryption and decryption are in Client, which meets two typical encryption requirements: static encryption (data stored in HDFS encrypted area is encrypted data) and transmission encryption (data is transmitted from Client to minefield in encrypted text), which protects data when an attacker steals a hard disk or intercepts a network.

B) Attacking HDFS clusters does not leak encrypted zone data.

C) The potential harm to a malicious user is limited. Malicious users can only access password files that they have HDFS access to, and can only decrypt EDEKs that are allowed by KMS ACL access rules. Its ability to access plaintext is limited to the intersection of the two.

There are several security risks to HDFS transparent encryption:

A) Risk of KMS being attacked: Client will read encrypted zone data normally by meta-method once key loss or corruption means data disaster. KMS key storage must be an authoritative storage key at the enterprise level to ensure that the key is not lost. KMS/keystore administrator and HDFS administrator roles should also be separated to avoid data disclosure due to KMS attacks.

B) Network sniffing risk: There is a risk of network eavesdropping when keys are transmitted in clear text between Client and KMS via http. The security of network transmission should be enhanced by configuring TLS/SSL.

C) The risk of impersonating a user: an attacker can impersonate a user to decrypt some files in the encrypted zone. HDFS and KMS users should be configured with authentication and permission rules in a safe production environment, while KMS ACL rules should be used to control user access to and operation of keys. minimum permissions for users can be set to ensure that a legitimate user can only get DEK of his own encrypted zone file in KMS, while other users without relevant permissions can be blocked.

B. Performance Analysis

HDFS transparent encryption uses symmetric encryption algorithm-AES, an advanced encryption standard algorithm, to encrypt data in CTR mode. By default, 128-bit length key encryption is used. HDFS transparent encryption is supported in Hadoop version 2.6.5 and above. This paper chooses to test the performance of HDFS transparency technology on the open source Apache Hadoop-2.7.1 platform: 1 NameNode and 3 DataNodes in a cluster; each computer with 2 CPUs and 64GB memory are adopted; 10 files are tested with 100GB file size; TestDFSIO tool tests showed that when HDFS transparent encryption was not applied, the rate of reading files was 153.123 MB/sec and writing files was 33.206 MB/sec; when HDFS transparent encryption was used, the rate of reading files was 142.726 MB/sec, which decreased by approximately 6.8%, The rate of writing files is 32.657MB/sec, which decreases by about 1.7%. It can be seen that HDFS transparent encryption technology has good performance.

C. Comparison of encryption at different levels

The Hadoop Big Data Platform implements several different forms of encryption, which can be divided from top to bottom into application layer encryption, database layer encryption, HDFS transparent encryption, file system layer encryption and disk encryption. The following chart compares different layers of encryption for the platform. You can see that encryption at the upper level is more flexible and meets the fine-grained security needs of users; encryption at the lower level is easier to deploy and operate. The lowest level of encryption encrypts all node data, effectively protecting the data, but lacks finer-grained encryption. HDFS transparent encryption is between the database house and the file system, and can achieve finer-grained encryption than file system.

TABLE I. COMPARISON OF DIFFERENT LEVELS OF ENCRYPTION ON HADOOP PLATFORM

Encryption Level	Advantages	Disadvantages
Application Layer Encryption	The safest and most flexible way to accurately respond to user security needs.	Difficult and requires changes in user application code
Database Layer Encryption	Many database vendors provide some form of encryption	Performance issues, indexes cannot be encrypted
HDFS Transparent Encryption	Provides high performance, transparent application, easy deployment, more flexible than file system encryption.	Number of keys makes key management difficult
File System Layer Encryption (FSL)	easy to deploy, high performance	the unflexible requirements of application security, database column encryption, end user data encryption for multi-tenant applications.
Disk encryption	easy to deploy, high performance	inflexible and prevents only physical data theft.

IV. CONCLUDING REMARKS

Data has become a national basic strategy of plundering and social basic production elements. Large data platforms carry a large amount of data, and effective measures need to be taken to resist the security risks of large data platforms. Research shows that HDFS transparent encryption technology has high performance, transparent application, easy deployment, and is more flexible than file system encryption. It is an effective technology to protect data security of large data platforms. However, in the application process, great importance should be attached to the security risks of KMS attack, network sniffing, and impersonating users.. By default, Hadoop uses file-based Java KeyStore for key storage, which does not meet the security needs of large enterprises (Client will not be able to properly read encrypted zone data once the key is lost or corrupted), which require a more robust and secure key management solution. Cloudera has developed Cloudera Navigator Key Trustee Server to provide platform key storage and management. However, due to the lack of dedicated key management products, the enterprise application of HDFS transparent encryption technology has not been achieved in the mainstream large data platforms in China. The application of HDFS transparent encryption technology needs to be further strengthened.

ACKNOWLEDGMENT

This work was supported in part by The Open Research Fund of National Engineering Research Center for Agro-Ecological Big Data Analysis & Application, Anhui University (No. AE2019011, No. AE2018010), the Key Natural Science Project of Anhui Provincial Education Department under Grant (No. KJ2019A0668), Supported by the Open Research Fund of AnHui Key Laboratory of Detection Technology and Energy Saving Devices, AnHui Polytechnic University (No. DTESD2020B05) and The Key Research and Technology Development Projects of Anhui Province (No. 202004a06020045)

REFERENCES

[1] Apache. Hadoop. [EB/OL]. http://hadoop.apache.org/.

[2] Ghemawat S , Gobioff H. and Leung S. T. The google file system[C]//Proceeding of the 19th ACM symposium on Operating systems principles, New York, USA,2003: 29-43.

[3] Dean J. and Ghemawat S. Map Reduce: simplified data processing on large clusters[J]. Communications of the ACM,2008,51(1): 107-113.

[4] Malley O, Zhang K, Radia S, et al. Hadoop security design. Sunnyvale, CA, USA: Yahoo Inc. , 2009.

[5] Khalil I, Dou Zuochao, Khreish A. TPA-based authentication mechanism of Apache Hadoop[C]//Proc of Int Conf on Security and Privacy in Commu-nication Networks. Berlin: Springer, 2015, 152: 105-122.

[6] Leicher A,Kuntze N ,Schmidt A. Implementation of a trusted ticket system [C]//Emerging Chanllege for Securiy,Privacy and Trust. Berlin:Springer, 2009, 152-163.

[7] Ruan A,Martin A. Towards a trusted map reduce infrastructure[C]//Proc of the 8th IEEE World Congrss on Services(SERVICES). Piscataway,NJ: IEEE,2012: 141-148.

[8] Shehzad, Danish &. Khan, Zakir &. Dag, Hasan &. Bozkus, Zeki. (2016). A Novel Hybrid Encryption Scheme to Ensure Hadoop Based Cloud Data Security[J]. International Journal of Computer Science and In-formation Security 194 7-5500. 14. 480-484.

[9] H. Lin, S. Shen, W. Tzeng and B. P. Lin, Toward Data Confidentiality via Integrating Hybrid Encryption Schemes and Hadoop Distributed File System [C]// 2012 IEEE 26th International Conference on Advanced In-formation Networking and Applications, Fukuoka, 2012, pp. 740-747. doi: 10. 1109/ AINA. 2012. 28.

[10] Chao Yang, Lin Weiwei, Mingqi Liu. A Novel Triple Encryption Scheme for Hadoop-Based Cloud Data Security [C]// Emerging Intelligent Data and Web Technologies (EIDWT) 2013 Fourth International Conference on, pp. 437-442, 2013.

[11] Wang Feng , Kohler M, Schaad A. Initial encryption of large searhable data sets using Hadoop [C]//Proc of 20th ACM Symp on Access Control Models and technologies,New Work:ACM,2015: 165-168.

[12] Cloudera. Sentry. [EB/OL]. https://www.cloudera.com/products/ open-source/apache-hadoop/ apache-sentry. html

[13] Cloudera. Recordservice.[EB/OL].(2015).http://blog.cloudera.com/blog/2015 /09recordservice-for-fine-grained-security-enforcement-across-the-hadoop ecosystem/.

[14] Hortonworks.Ranger. [EB/OL].(2016).https://hortonworks. com/a-pache/ranger I.

[15] Hortonworks. KnoxGateway.[EB/OL].(2016).https://horton-works. com/apache/ knox-gateway/.

[16] Intel.Project Rhino.[EB/OL].(2015).http://github.com/intel-ha-doop/projecthino/

[17] ZHOU Daoming, QIAN Lufeng, WANG Lulu. Transparent encryption technology research [J]. NetInfo Security, 2011, 12 (14): 54-56. (in Chinese)

[18] WANG Quanmin, ZHOU Qing, LIU Yuming, ZHU Erfu. Research on file system transparent encryption techniques [J]. Computer Technology and Development, 2010, 3(20): 147-150. (in Chinese)