The Design and Implementation of User Autonomous Encryption Cloud Storage System Based on Dokan

Jiawei Yin, Junxiang Yang, Yang Chen
College of Software, Jilin University
JLU
Changchun, China
Yangjx5513@hotmail.com

***Abstract*—To solve the problem of low security in cloud storage system under the background of rapid prevailing cloud technology, this paper studies the design and implementation of User Autonomous Encryption Cloud Storage System based on Dokan. With the help of Dokan's file development model, a specific file system intended for curtain group of users can be easily customized. In order to overcome the limitations existing in traditional cloud storage systems, this system manages to establish the environment with embedded AES encryption algorithms on local clients and cloud storage servers based on OpenStack. The experimental results show that the User Autonomous Encryption Cloud Storage System based on Dokan is of higher security when compared with the traditional cloud storage systems.**

Keywords—Dokan, OpenStack, AES, Cloud Storage System

I. INTRODUCTION

Nowadays, along with the rapid popularization and development of the Internet, cloud storage has gradually become an essential part of network life. An increasing amount of user data is now stored in all kinds of network servers, which causes a great deal of sensitive information being transmitted among network nodes either in wired or wireless ways [1]. The private data is rather valuable that it will seriously threaten personal privacy and economic interests once divulged. Extensive researches and studies of cloud storage have been carried out (e.g., Research of cloud storage service system based on HDFS by Xiaoyun Huang [2], Research and implementation of the secure cloud storage system based on Hadoop by Fubin Pan [3] and the analysis of storage based on the open source platform OpenStack by Bin Zhi [4]). These studies show that a variety of defects exist in current cloud storage system. Most of them do not support the encryption of data, while the few who are capable of encryption execute the encryption algorithms on the cloud and store the generated keys in the servers. In both cases, users are not able to manage the keys autonomously. As can be found on the report of the cloud storage application investigation by Twinstrata in 2012, only 20 percent of people are willing to store their private data in cloud storage and about 50 percent of people would like to use cloud storage for data backup, filing and disaster recovery, etc.

In consideration of the privacy protection of cloud data, the design and implementation of User Autonomous Encryption Cloud Storage System based on Dokan is proposed. The system encrypt data with a key entered by its user so that the key is not stored in any kind of physical medium.

II. RELATED TECHNOLOGIES ANALYSIS

A. OpenSatck

OpenStatck is an open-source software platform for cloud computing. The software platform consists of interrelated components (including Nova, Glance, Keystone, Cinder, Swift, Horizon, etc.), which control hardware pools of processing, storage, and networking resources throughout a data center. OpenStack supports almost all types of cloud environment and aims to offer the abilities of convenient, abundant and massively-extensible, unified standard cloud computing management.

OpenStack has provided two ways of cloud storage. One of them, Cinder, is an online storage service that keeps the data even when the virtual machine is shutdown. The other one is Swift, an off-line storage service for large scale data storage [5]. Cinder is a new component separated from the persistent block storage function (Nova-Volume) of the Nova in the F version of OpenStack. Through integrating various storage methods on the back end, it provides block storage service for the external users and programs with uniform API, the core of which is the management of volume that allows to handle the volumes as well as the types and the snapshots of them.

B. File System Development Model: Dokan

Dokan is a file system development model, a lightweight package that consists of a library and a kernel mode system driver that designed to provide developers with a new way of creating file systems in Windows environment. Being similar to FUSE, the user mode file system for Linux, Dokan allows the creation of file systems without writing the device driver, this being provided by its engine directly. [6]

Dokan runs in both user mode and kernel mode. On one hand, its kernel driver mainly works in kernel mode, which is responsible for coordination of the communication between

Windows I/O subsystems and Dokan user mode library. On the other hand, file system program and Dokan user mode library mainly work in user mode. They will handle the file request from application program and return the results of handling to the source program.

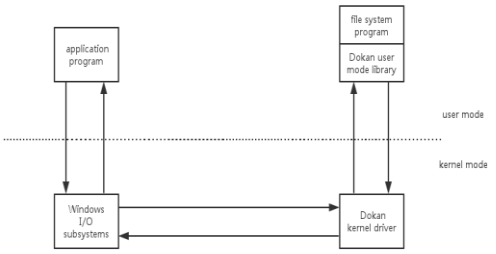

Fig.1. Working principle diagram of Dokan

C. AES (Advanced Encryption Standard)

The Advanced Encryption Standard (AES), also known as Rijndael, is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001. For AES, each cipher block has a fixed size of 128 bits, but the key lengths vary from 128, 192 and 256 bits (the length will be completed when inadequate). AES contains a lot of repetitions and transformation. [7]

The general steps are:

- Key Expansion: round keys are derived from the cipher key using Rijndael's key schedule. AES requires a separate 128-bit round key block for each round plus one more.
- Initial Round: AddRoundKey—each byte of the state is combined with a block of the round keys using bitwise xor.
- Rounds:
    1. SubBytes—a non-linear substitution step where each byte is replaced with another according to a lookup table.
    2. ShiftRows—a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.
    3. MixColumns—a mixing operation which operates on the columns of the state, combining the four bytes in each column.
    4. AddRoundKey.
- Final Round (no MixColumns):
    1. SubBytes.
    2. ShiftRows.
    3. AddRoundKey.

III. The Architecture Design of User Autonomous Encryption Cloud Storage System

Several access methods based on a local client are provided for the external user in the system. In other words, users can manipulate the intended files through a local client, including storage, upload, download and dynamic fuzzy search of files. When uploading, the system calls for a Dokan writing method embedded with AES encryption algorithm to encrypt the file to be uploaded through a hook function. The uploaded encrypted files are stored in a private cloud server constructed with OpenStack. When downloading, the system calls for a Dokan writing method embedded with AES decryption algorithm to decrypt the ciphertexts. During the whole process, the disk space used is a virtual disk created by Dokan, the space of which can be controlled through setting parameters. That can make users develop a proper space as needed.

In the system, the client is not only just the operation panel to users, but also a hub between local end and server end. It transfers the encrypted files through network transmission protocol to store them in the server and delivers the downloaded ciphertexts to the local user mode file system for decryption. The system architecture diagram of the User Autonomous Encryption Cloud Storage System is shown below:

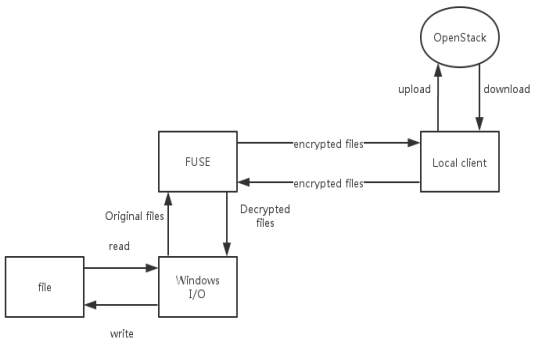

Fig.2. System architecture diagram.

IV. The Implementation of User Autonomous Encryption Cloud Storage System

A. The implementation of Dokan file system

Based on the original Dokan file system, a mirror. C file is created. The DokanOperation structure contains function pointers (including CreateFile, ReadFile and WriteFile, etc.) that represent the corresponding file operation functions. It is also the structure that performs the one-to-one allocations of pointers, some of which would perform different allocations with different user's operation instruction given. Eventually the DokanOperation variable will be transferred to the file system through the DokanMain interface to execute the file operation function implemented in mirror.c.

To meet the requirement that files should be encrypted when uploaded and decrypted when downloaded, the WriteFile function pointer should be implemented in two concrete ways: MirrorUploadWriterFile and MirroDownload-WriteFile, which perform different assignment with different user instruction given. There are embedded file encryption and decryption operations along with common file writing operations.

The MirrorUploadWriteFile function calls for an AES encryption function, to which a key generated from user instruction and sent to Dokan file system will be transferred. The key will be extended and compressed into a 16-bits AES encryption key under certain principles, and it will then be used as the standard for the file stream encryption. The encrypted files will be written in the virtual disk created by Dokan.

MirrorDownloadWriteFile function is somehow alike, the only difference is that it calls for an AES decryption function.

B. The implementation of client

The client, consists of user login module, file browser module, download/upload module, deletes module and file retrieve module, is developed with Qt Creator. A series of dynamic link libraries (including QtCored, QtGuild, QtWidget, etc.) has been used during the development process.

- User Login module: User should first enter their account name and password to validate access, then enter an encryption code for twice. The encryption code will first be checked, next to which the account name and password will be sent to the server through TCP protocol to check for validity.
- File Browser module: The module is mainly responsible for the maintenance and update of the ListWidget in the user interface. After user login, the client will handle the corresponding file information sent from the server through SSH protocol. A temporary file will be created at the temporary file path in the system with the target filename extension. The client will request the icon of the file in the system from QIcon class, then a corresponding QListWidgetItem object will be instantiated and added to the ListWidget. The operations are the same that when updates a ListWidget. The files in the temporary folder will be deleted when the system is shutdown.
- Upload/Download/Delete Module: By clicking on certain areas in the user interface, a variety of slot functions will be called. During the uploading process, the system receives the path of a file from a file selector, which is implemented with QFileDialog, and then creates a thread to convey the upload command that request for virtual disk initialization. The file at the specific path will then be transferred to the virtual disk in file stream and gets encrypted there. Next, the client will convey the upload command to the server and transfer the ciphertexts in the virtual disk to the server. The file browser will be refreshed when upload is done. Eventually, the related process will be shut down and the virtual disk will be empty. The download process is basically the same as upload, except some minor differences in specific instructions. The delete operation will first send the delete command to the server, then transfer the filename and refresh the file browser afterwards..
- File Retrieve module: The "textEdited" signal in the search box is connected with the "searchListByName" slot function so that every time a search box is edited the file browser will update dynamically.

C. The implementation of the server

Establish the Nova, Cinder, Horizon, Keystone nodes of OpenStack system using Fuel and develop the server side on the complete OpenStack environment. Keystone validates the user information when server receives a login request. Nova creates a corresponding instance that represents the identity of the user, which contains a volume of Cinder node that carries the files of the user. When the user requests a file, the file is transferred in and out through FTP protocol.

V. THE IMPLEMENTATION OF USER AUTONOMOUS ENCRYPTION CLOUD STORAGE SYSTEM

The design and implementation of user autonomous encryption cloud storage system based on Dokan is suggested, which provides reference on building a cloud storage system with great security and reliability using Dokan. This system focus on Dokan framework embedded with AES encryption algorithm, adopts private cloud server built with OpenStack as file storage space and achieve the goal of files encryption and decryption during upload and download process, which separates the users from specific encryption and decryption operations. Through combining with the Dokan framework embedded with AES encryption algorithms and OpenStack, a user autonomous cloud storage system with great security, reliability and flexibility is built, which not only solves the problems of user private data upload and storage security but also provides persistent competitive advantages for enterprises that are developing a cloud storage system in an age where users cares a great deal about the security of data.

REFERENCES

[1] Tianliang Lu, Yunwei Zhou and Yinqing Liu. “Review on problem of internet user privacy leakage”, Computer Science, vol. 41, pp. 62-67, November 2014.

[2] Xiaoyun Huang. “Research of cloud storage service system based on HDFS”, Da Lian: Dalian Maritime University, June, 2010.

[3] Fubin Pan, “Research and implementation of the secure cloud storage system based on hadoop”, Cheng Du: University of Electronic Science and Technology of China, Februry 2010.

[4] Bin Zhi, “Design and implementation of a private cloud storage system”, Beijing: Beijing University of Posts and Telecommunications, May, 2011.

[5] Hong Deng, Dongxing Wang, “The analysis of storage based on the open source platform OpenStack”, Heilongjiang Science and Technology Information, vol 32, pp 134, November 2012.

[6] Yimeng Tian, “Design and implementation of windows client for blueocean mass storage system”, Shanghai: Shanghai Jiao Tong University, January 2013.

[7] https://en.wikipedia.org/wiki/Advanced_Encryption_Standard