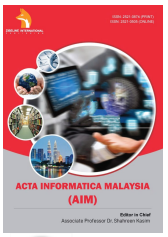
ZIBELINE INTERNATIONAL PUBLISHING

***ISSN: 2521-0874 (Print)***
***ISSN: 2521-0505(Online)***
***CODEN: AIMCCO***

**Acta Informatica Malaysia (AIM)**

*DOI: http://doi.org/10.26480/aim.02.2020.19.21*



CrossMark

***REVIEW ARTICLE***

# INTEGRATED CRYPTOGRAPHICAL ACCESS CONTROL OVER NETWORK PROJECT

**Yakubu Ajiji Makeri**

***Kampala International University Uganda, School of Computing and Information Technology***
****Corresponding Author Email: yakubu.makeri@kiu.ac.ug***



**ARTICLE DETAILS**

***Article History:***

Received 03 April 2020
Accepted 05 May 2020
Available online 18 May 2020

**ABSTRACT**

Cryptanalysis is a new ID-based encryption scheme proposed by Meshram. I found a method for factor N, where N is the parameter proposed by Meshram. We also provide a method for retrieving the Secret Master key for Mayshram's ID-based encryption scheme. Identity-based (ID-based) cryptography is very useful because it simplifies certificate management in public-key cryptocurrency. For the design of the Integrated File Level Cryptographic Access Control (IFLCAC) system, it makes file security much easier for the end-user. This system combines the advantages of traditional file-level cryptography and full-disc cryptography systems, making it safe and easy to use. We first look at existing file cryptography systems, compare them to two, and then describe the interactions between components and components of the integrated file-level cryptographic access control system. Because its defense relies on the difficulty of discrete logarithmic and integer factor problems, it proves that his scheme is safe against favorable select-plain invasion. We show that this new ID-based encryption scheme is not secure by introducing a method to retrieve the secret master key.

**KEYWORDS**

google script analysis is a new ID-based encryption

## 1. INTRODUCTION

The value of data stored on digital platforms is growing much faster than existing assignment schemes, showing that our scheme's high-security class can effectively retrieve any secret key of its successor and not change any keys issued when new security is available. The class is added to the hierarchy (Bell and Lapadula, 1973). File-level cryptographic access control is important to ensure that sensitive data is stored and accessed securely. File cryptography systems today follow three basic models: file-level cryptography, virtual partition cryptography, and file system cryptography. Each design brings its own strengths and weaknesses when implemented. Lastly, most access control schemes are very stable, which describes the procedure applicable during normal operation but requires exceptional allocation by the exceptional authority when exceptional access is required (Nartional Bureau of standard, 1977). We will present a system for accountable cryptographic access control to address these issues.

File-level cryptography is the traditional method. This design provides special control over which files are encrypted; Each file is manually encrypted and decrypted by the user. Unfortunately, this method is also very difficult for the end-user (Denning, 1976). There are so many file-level cryptography applications today. One such application is Expcriptory for Windows. AxCrypt enables the user to encrypt files using the AES 128 bit standard encryption algorithm.

File System Cryptography protects the entire file system. It uses a separate file system that encrypts all the data that goes into the file system and decrypts all data from the file system. All files written to disk are stored securely (Axcrypt, 2007). This design is easy for the user to use, but consumes a lot of overhead. Additionally, this method depends on the file system, which means that when the user changes their file system (eg, from FAT32 to NTFS), the entire file system needs to be rebuilt. PGP whole disk encryption is the implementation of such file system cryptography. The application has the option to encrypt the entire contents of the drive attached to the computer, including the boot sector and swap files. Virtual partition cryptography tries to find a balance between file-level and file system cryptography (Kher and Kem, 2005). A separate partition of the disk is created to store sensitive files securely, but non-sensitive files are stored on a regular partition. This design forces all secure files to a single central location, but allows greater granularity than file system cryptography. Virtual partition cryptography assigns the required SP

## 2. TROUBLESHOOTING

File systems and file-level cryptography, when used in combination, create a highly secure system. Such a design allows all files to be protected, with an extra layer on the most sensitive files. The overhead of file system cryptography is offset by the increasing speed of personal computing, and the lack of granularity is overcome by the use of file-level cryptography. The solution leaves the user the same as the previous one (Lee et al., 2004). This means that the user still has to manually encrypt and decrypt the most sensitive files in the system. This encryption scheme, the overhead required by the user, introduces a dangerous element to the user. Once the

| Quick Response Code | Access this article online | |
|---|---|---|
| | **Website:** www.actainformaticamalaysia.com | **DOI:** 10.26480/aim.01.2020.19.21 |

file is accessed, it is easy for the user to forget to encrypt it. In addition, this user overhead limits the total number of files required for additional encryption; This limit varies on a per-user basis. The limit is the amount of overhead the user is willing to incur (Ludwig and Kalfa, 2001). To ensure that all necessary files are properly protected, this user element in the system should be reduced as much as possible. A user-friendly, efficient and secure file-level cryptography method is needed to fill the void left in file cryptography. The new method must meet certain criteria to completely fill the void:

(A). The method should not introduce a new weak link in the security system.

(B). This method should be natural enough to work for end-users and simple enough for non-computer literate users to operate and maintain.

(C). The method must be independent of the file system; This ensures portability between all computers on the system. Independence is very different, meaning that the program can be run on different operating systems such as Windows and Solaris. Playtime freedom is virtually impossible for platform-level cryptography systems.)

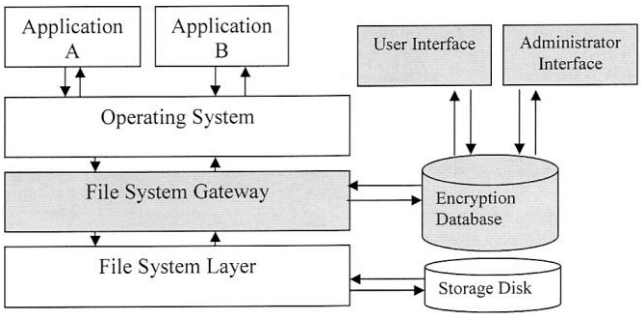
(D). The method is easy to update; This ensures that the method evolves with the changing security situation.

## 3. Proposed Solution

We first introduce the structure and various components of the new system, and then in Section 3.2, the model interactions between components are discussed to illustrate how a particular operation is performed in the new system.

### 3.1 Unified File Level Secure Access Control

As shown in Figure 1, the design of the new file-level cryptography method consists of four main components: the file system gateway, the encryption database, the user program, and the administrator program. The file system gateway sits just above the file system, blocking calls made to the file system (Naor et al., 2005). The encryption database stores the required information about encrypted files. The user program provides a simple interface for the end-user. The administrator provides access to the program's configuration options. All encryption and hash algorithms are stored in the 110 exporting dynamic link library.




Pre-defined interface. Storing encryption and hash algorithms in this way allows for easy insertion and updating of algorithms.

File system gateway is a very important part of the new cryptography method. This element causes all data encryption and decryption. The gateway controls all communication between the top application layers and the underlying file system. The database file system has all the configuration options for the gateway element (PGP Whole Disk Enoyption, 2007). The gateway handles all file request calls from the application, queries the database to check if the file is securely stored. The unique name from the file is used as an index in the encryption database. If cryptographic operations are required, the appropriate dynamic link library is loaded. Cryptographic operations are performed and the resulting data is returned to the request.

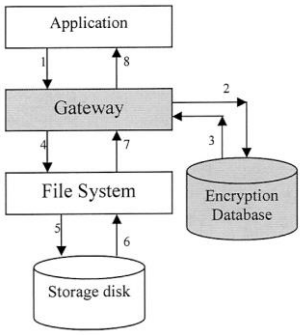
An important aspect of the entrance needs to be pointed out. The gateway is implemented as a layer on top of the file system; In this way, it should be completely independent of the file system. It depends on the specific file pointer information of any file system. In addition, an additional dynamic link library can be used to implement some special functions related to applying specific names to a given file system (Truecrypt, 2007). The only place for file system dependencies is in this library. The gateway calculates the encryption key based on the user's password after logging into the system. Additional passwords must be retrieved directly from the user to open encrypted files with a different key. This adds to the need for the gateway to reside on the client machine when accessing a distributed file system. The functionality of cryptography in the user machine allows minimal fidelity between the local computer and the remote file server, and eliminates the need to distort the asymmetric key cryptography before communication between the remote file server and the local computer.

An encryption database is used to store all information related to encrypted files. Additionally, it can also be used to store configuration options for other elements. It needs to be designed to access the database effectively, because every call to open the file queries the database. The encryption database consists of two key elements, the file system database and the configuration options. The file system database stores 111 information about which files are encrypted and what algorithm is used to encrypt the file. The other key aspect of the database is the configuration option; This element contains information on the available Dynamic Link Library, the current file system in use and other common options such as password rules and administrative settings (Boneh and Boyen, 2011). The encryption database contains all the information needed to allow proper access to the file system. Unfortunately, this provides only one failure with the integrated file-level encryption method. At the very least, the administrator can access the program to manually decrypt the files if the database fails. If communication with the encryption database is not available, the gateway enters an invisible state and returns the data directly while it is on disk. This status allows normal operations on unencrypted files, but encrypted files are accessed by the admin program. The user program allows the end-user to interact with the integrated file-level encryption system. The program displays the path for encrypted files and path modifiers (Boneh et al., 2004). The program allows the user to add a file to the encryption list. The user program also allows basic management, such as changing their password. For most users, this program is the only interface on the system. The admin program allows advanced access to the system's configuration and database. The program allows you to manually encrypt files.

### 3.2 Element Interaction

Four elements come together to create a new cryptography method. Both the file system gateway and the encryption database are constantly running; Both must be executed for a user to retrieve the encrypted file on the system. The other two components, the user program, and the administrative program, cannot be implemented unless they need to change the current system configuration (Boneh et al., 2004). The system can also be used to ensure file integrity. Encryption can digest the message of any file in the database. When the file is accessed, the new message digest is calculated compared to the stored value. If the message is not digested, the file is edited outside the integrated file-level cryptography system and notified to the user.

The most common interaction for opening a file is a simple single-user decryption. As shown in Figure 2, the file's unique name is searched through the database of the modified files (Step 2) and the information is returned to the gateway (Step 3). Gateway loads the encryption algorithm and, if necessary, the message-digest algorithm from the appropriate Dynamic Link Library. The file stream is opened by a simple file layer call (Step 4). All the required data is logged into the memory list 112




Steps:
1) Application makes a file request
2) Gateway queries database.
3) Gateway performs necessary crypto graphical functions.
3) Database returns file information.
4) File request continues to File System.
5) File system handles request.
6) File system returns data to Gateway.
7) Gateway makes necessary crypto graphical functions.
8) Decrypted information is returned to the calling application.

Support multiple files being opened simultaneously. The open file pointer then returns to the calling application . For a file read request, the open file is decrypted to fill the request from the calling application. File writing can be a bit complicated depending on the encryption algorithm used. Many symmetric key algorithms encrypt data in blocks of a set size. Before data can be officially written to disk, a complete block or file shutdown is required. Finally closing the file completes any abnormal data encryption. Only the symmetric key algorithm should be used for single-user file access.

Figure 2 - Simple Operation of Retrieving Encrypted File The integrated file-level cryptography method allows easy modification of encryption algorithms used in the system. As the stronger and more efficient algorithms are published, new encryption algorithms can be incorporated through dynamic link library systems. When updating the new encryption algorithm, all currently encrypted files must be changed. This is an inevitable and costly operation. Encryption databases can be used to perform these tasks in a very timely fashion. When accessing an outdated file, the original encryption scheme is used to decrypt the file, while a new encryption scheme is used to decrypt the file. If the old encryption algorithm is damaged, it is possible for the administrator to immediately convert all the files.

For files with multiple user access and user/administrator access, a major escrow system should be implemented. When a file is flagged for multi-user access, a random symmetric key for the file is generated. The copy of the symmetric key is encrypted with each user's public asymmetric key (Denning, 1976). Symmetric keys are secured by the asymmetric encryption algorithm currently in use. The encrypted key is then stored in the user database for retrieval and decryption. This method of key escrow allows the administrator to easily update group access files and perform file encryption using stronger and more efficient symmetric key encryption.

## 4. Applying For Classroom Teaching

The IFLCAC system can be used to teach various courses such as Operating Systems, DataBase, Advanced Operating Systems, Computer Security and Capstone Project. There are many possibilities. In this section, we explore some approaches. In the Advanced Operating Systems or Capstone Projects course, for example, students may be asked to create a detailed design of the IFLCAC system, followed by system implementation and evaluation. One way to use the system in a computer security class is to provide students with a fully developed system that asks them to implement and evaluate various cryptographic algorithms in the system (DES, AES, blowfish, etc.). For beginner operating system courses, students will need to implement a file system gateway component, while the other three components (encryption database, user interface and administrator interface) are provided by the instructor. On the other hand, is a database course, the instructor can provide students with file system gateway and user interface components, while students must design and implement encryption database and administrator interface components. At the discretion of the instructor, other combinations of available components are possible for available-developed parts.

## 5. Conclusion

In our protocol, the decryption authorities do not know which Dk they are capturing the identity key for. However, some or all authorities may be empowered to determine which records are decrypted. Our protocol can be easily generalized to cover this case - this endorsement leaves only the blinding mechanism from the IBE extraction protocol for ambiguous authorities, while using it for vague auto

## References

AxCrypt. 2007. Quantum Software AB. October 31, 2007, http://www.axantum.com/AxCrypt/

Bell, D.E., Lapadula, L.J., 1973. Secure computer systems: Mathematical foundations and model. Rep. M74-244, The MITRE Corp., Bedford, Mass.

Boneh, D., Boyen, X., 2011. Efficient selective identity-based encryption without random oracles. J. Cryptology, 24(4), 659–693.

Boneh, D., Canetti, R., Halevi, S., Katz, J., 2007. Chosen-ciphertext security from identity-based encryption. SIAM J. Comput. 36(5), Pp. 1301–1328.

Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G., 2004. Public key encryption with a keyword search. In: Advances in Cryptology-Eurocrypt, Pp. 506–522.

Denning, D.E. 1976. A lattice model of secure information flow. Commun. ACM 19, 5, 236-243.

Kher and Kim, 2005. Securing Distributed Storage: Challenges, Techniques, and Systems. Proceedings of Workshop on Storage Security and Survivability. ACM Press.

Lee, Boyd, Dawson, Kim, Yang, Yoo, 2004. Secure Key Issuing in ID-based Cryptography. Australasian Information Security Workshop. Australian Computer Society.

Ludwig, Kalfa, 2001. File System Encryption with Integrated User Management. ACM SIGOPS Operating Systems Review. ACM Press.

Naor, Shenhav, and WoolToward, 2005. Securing Untrusted Storage without Public-Key Operations. Workshop on Storage Security and Survivability. ACM.

National Bureau Of Standards. 1977. Data Encryption Standard, Federal Information Processing Standards (FIPS), Publication 46. National Bureau of Standards, Wash., D.C.

PGP Whole Disk Encryption. 2007. PGP Corporation. October 31, 2007. http://www.pgp.com,products,wholediskencryption

TrueCrypt, 2007. TrueCrypt Foundation. October 31, 2007, http://www.truecrypt.org.