파일 부분 암호화 지원을 위한 시스템 호출에 관한 연구

서혜인* · 성정기* · 김은기*

*한밭대학교

A study of a System Call Interface for Supporting File Partial Encryption

Hye-in Seo* · Jeong-gi Seong* · Eun-gi Kim*

*Hanbat National University

E-mail : iloveu2727@naver.com

요 약

현재 디스크에 파일을 암호화하여 저장하기 위한 다양한 파일 암호화 시스템 및 응용 프로그램들이 존재한다. 하지만 기존의 파일 암호화 솔루션은 암호화 및 복호화를 파일 혹은 디렉터리 단위로 일괄되게 처리한다. 본 연구에서는 파일의 부분적 암호화 기능을 지원하는 시스템 호출을 제안한다. 사용자가 시스템 호출 인터페이스를 사용하여 파일의 부분적 암호화 기능을 설정한 후, 파일의 내용을 쓰면 디스크에 암호화되어 저장된다. 또한 복호화 기능을 설정한 후 파일의 내용을 읽어오면 설정된 내용이 적용되어 필요한 부분만을 복호화 한다. 사용자 설정에 따라 필요한 부분만을 암호화하여 저장매체에 저장함으로써 비밀 수준의 정보들을 효율적이고 안전하게 보관할 수 있다.

ABSTRACT

There are currently various file encryption systems and applications for encryption and storage of file on disk. However, the existing file encryption solutions handle encryption and decryption all at once by file or directory. In this study, we propose a system call supporting partial encryption function of the file. The user sets the partial encryption of the file by using system call interface, and writes the contents. And then the data is encrypted and stored on the disk. Also if the user sets the decryption and reads the data, the necessary part of data is decrypted by applying the user setting. According to the user setting, only the necessary part is encrypted and stored on a storage medium. As a result, the information in a secret level can be saved efficiently and securely.

키워드

시스템 호출, 리눅스 커널, 파일, 암호화, 암호 키

Ⅰ. 서 론

정보화 시대의 급속한 발전으로 인해 현재는 개인 정보와 같이 중요한 데이터를 암호화하여 디스크에 저장하기 위한 다양한 암호화 파일 솔루션이 존재한다. 기존의 암호화 솔루션들은 데이터의 기밀성을 지원하지만 암호화의 단위가 파일 또는 폴더 전체로 한정된다[1-3]. 이와 같은 방법은 파일 전체를 일괄적으로 암호화하기 때문에 파일의 내용을 부분적으로 읽을 때, 필요한 부분만이 아닌 전체를 모두 복호화해야 한다는 성능상의 오버헤드가 따른다. 이러한 기존의 파일 암호화 솔루션의 문제를 보완하고자, 본 연구에서는 시스템 호출을 이용하여 파일을 부분적으로 암호화하고, 복호화 할 수 있는 방안을 제안한다. 이 방법은 시스템 호출을 이용하여 사용자가 직접 암호화 구간을 설정할 수 있으며, 암호화 알고리즘, 암호 키 또한 유동적으로 설정할 수 있다. 제안된 시스템 호출을 이용하여 암호화가 필요 시되는 부분만을 암호화하기 때문에 기밀성이 불필요한 부분까지 암호화하는 오버헤드를 줄일 수 있다.

Ⅱ. 본 론

2.1 파일 부분 암호화 지원을 위한 시스템 호출 실행 과정

본 연구에서는 기존의 시스템 호출 fcntl, write, read를 수정하여 파일의 부분 암호화를 지원한다. 이 시스템 호출들은 내부적으로 검사 모듈, 관리 모듈, 암호화 모듈, 복호화 모듈, HMAC 모듈을 필요한 경우에 따라 거치며 부분 암호화 및 복호화를 수행한다.

사용자는 파일 부분 암호화 설정을 위해 fcntl 시스템 호출을 사용한다. 부분 암호화를 원하는 부분에서 fcntl 시스템 호출로 암호화를 설정한 후, write 시스템 호출로 파일의 내용을 입력하면 설정에 따라 파일 데이터가 암호화되어 저장된다. 또한 fcntl로 복호화를 설정한 후, read 시스템 호출을 이용하여 복호화 된 데이터를 읽어온다. 아래 그림 1에서 각 시스템 호출의 모듈들을 커널 내부 계층 구조에 나타낸다.

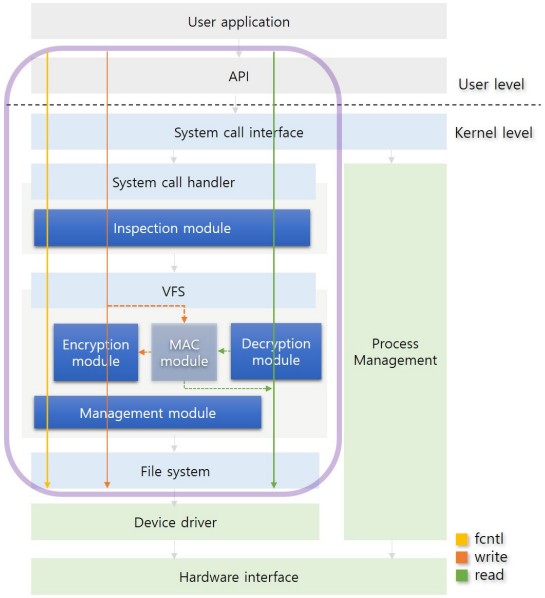

그림 1. 파일 부분 암호화를 위한 커널 계층 내 모듈 구조

2.2 부분 암호화 설정을 위한 기존의 시스템 호출 수정 사항

사용자는 원하는 부분의 암호화를 설정하기 위해 fcntl 시스템 호출을 사용한다.

2.2.1 fcntl 시스템 호출

파일의 부분 암호화를 지원하기 위해 기존의 fcntl 시스템 호출의 flock 구조체와 커맨드를 수정하였으며, 아래 표 1,2는 그 수정 사항을 나타낸다.

표 1. flock 구조체 수정 사항

prototype	int fcntl (int fd, int cmd, struct flock *lock)	
flock structure	original	modified

	struct flock{
	short l_type;
	short l_whence;
	long l_start;
	long l_len;
	int l_pid;
	}
		struct flock{
		short l_type;
		short l_whence;
		long l_start;
		long l_len;
		int l_pid;
		u8 algo;
		char *key;
		}

표 2. fcntl 시스템 호출 cmd 수정 사항

	name	description	value
original	F_GETLK	get record lock information	5
	F_SETLK	set record lock	6
	F_SETLKW	set record lock (with blocking)	7
added	F_GETSEC	get encryption or decryption status	18
	F_SETSEC	set encryption or decryption	19

수정된 fcntl 시스템 호출로 사용자가 원하는 파일의 구간에서 암호화 알고리즘, 암호 키를 설정할 수 있다. 또한 HMAC을 이용한 메시지 무결성 검증[4] 기능 사용 유무도 설정할 수 있으며, 복호화 설정 또한 복호화에 사용될 알고리즘, 키를 설정해 줌으로써 이루어진다. 다음 표 3은 커널 내부에서 유지 및 관리하고 있는 파일 부분 암호화에 대한 설정 값이다.

표 3. fcntl 설정에 따른 커널 내부 상태 값

status	value	description
SNONE	0x00	general write or read
SENC	0x01	encrypt and write
SENCM	0x02	encrypt and write (with HMAC)
SDEC	0x03	read with decryption

다음 그림 2는 커널 내부에서 유지 및 관리하고 있는 파일 부분 암호화 설정에 대한 상태도이다.

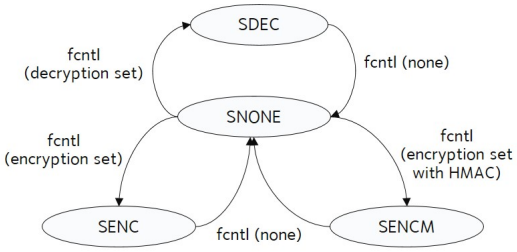

그림 2. fcntl 설정에 따른 커널 내부 상태도

2.3 동작 정의

제안된 시스템 호출의 암호화 및 복호화 설정은 상태가 SNONE으로 돌아오기 전까지 한번 설정된 내용이 계속 적용된다. 본 연구에서는 제안된 시스템 호출의 시나리오를 정의하여 그 동작 결과를 예상하였다. 아래 그림 3, 4는 파일 부분 암호화를 위한 시스템 호출 시나리오와 그 결과를 나타낸다.

```
char *data1 = “abcde”;                                  // 5bytes
char *data2 = “123456789012345”;                        // 15bytes
char *data3 = “ABCDE”;                                  // 5bytes
char *data4 = “123456789012345678901234567890”; // 25bytes

① write (data1);
② fcntl (F_SETSEC, F_ENC with aes-256-ecb / key1);
③ write (data2);
④ fcntl (F_SETSEC, F_NONE);
⑤ write (data3);
⑥ fcntl (F_SETSEC, F_ENCM with aes-128-cbc / key2);
⑦ write (data 4);
   …
```

그림 3. 파일 부분 암호화를 위한 시스템 호출 시나리오

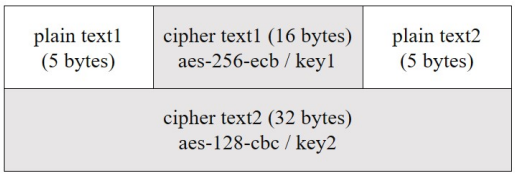

그림 4. 파일 부분 암호화 시나리오에 따른 예상 결과

그림 3의 시나리오에서 ①, ⑤의 write는 데이터가 평문 그대로 쓰여 지고, ②, ⑥의 설정에 따라 ③, ⑦의 write는 파일의 부분별로 다른 암호화 알고리즘과 키가 적용되어 암호화 된 결과를 예상할 수 있다.

Ⅲ. 결론

본 연구에서는 기존의 파일 암호화 솔루션에서 파일 또는 폴더 전체를 일괄적으로 암호화함에 따라 발생하는 성능상의 오버헤드를 보완하고자, 사용자가 암호화를 원하는 파일의 부분만을 암호화할 수 있는 시스템 호출을 제안했다.

제안된 시스템 호출은 fcntl 시스템 호출로 암호화 알고리즘, 암호 키, HMAC 사용 유무를 설정한 후, write 시스템 호출을 통해 파일 데이터를 암호화하여 쓴다. 마찬가지로 fcntl을 통한 복호화 설정 후, read 시스템 호출을 이용하여 파일 데이터를 필요한 만큼 복호화하여 읽어온다.

본 연구의 파일 부분 암호화 방안은 파일의 부분적인 암호화 지원을 통해 불필요한 부분을 암호화할 필요가 없을뿐더러 암호화에 적용될 설정이 사용자에 의해 제어되기에 리눅스 환경에서 유동적인 파일 암호화가 가능할 것으로 예상된다.

본 연구의 시스템 호출은 모듈별로 리눅스 커널 내부에 구현될 예정이며, 그 성능을 개발 보드에서 검증할 예정이다.

ACKNOWLEDGMENTS

This research was supported by the research fund of Hanbat National University in 2017.

참고문헌

[1] J. H. Kim, T. K. Part, and G. H. Cho, “User Transparent File Encryption Mechanisms at Kernel Level,” The Journal of Korea Institute of Information Security And Cryptology, vol. 16, no. 3, pp.3-16, June. 2006.

[2] J. Y. Heo, J. M. Park, and Y. K. Cho, “An Efficient Encryption/Decryption Approach to Improve the Performance of Cryptographic File System in Embedded System,” The Journal of Korean Institute of Information Scientists and Engineers, vol. 35, no. 2, pp.66-74, Feb. 2008.

[3] The Linux Documentation Project. Cryptographic File System under Linux HOW-TO LINUX SECURITY FAQ [Internet]. Available: http://www.tldp.org/pub/Linux/docs/faqs-archived/security/Cryptographic-File-System

[4] IETF Std. RFC 2104, HMAC: Keyed-Hashing for Message Authentication, IETF, 1997.