UsiFe File System (User Space File System with Support for Intra File Encryption)

UsiFe is an intra-file based encryption file system. It lies over the top of existing ext2 file system, and uses FUSE library to intercept file system calls. This document describes a little bit about the file system. For an extended introduction and more details refer to the paper here.

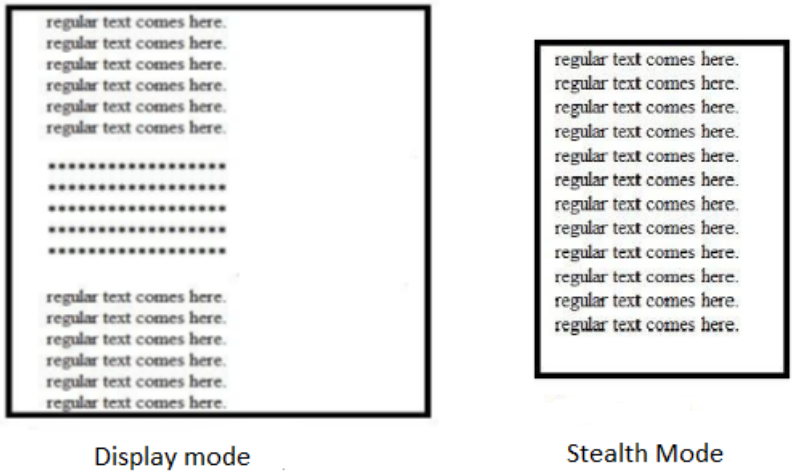
Introduction

Traditionally, cryptographic file systems have mainly focused on encrypting entire files or directories. But UsiFe takes a different approach and allows for encryption at a finer granularity, i.e. encrypting parts of files. This file system could be useful for protecting parts of large XML files, scientific data or any other use for protecting sensitive portions of a file.

How it works

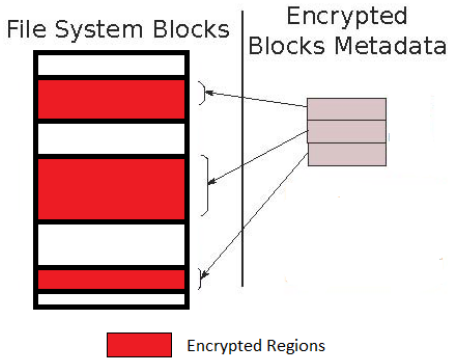
UsiFe has two modes of operation

- **Stealth Mode:** In stealth mode, encrypted data is kept hidden from user. All encrypted data is considered as non-existent. File effectively looks shortened to the user.
- **Display Mode:** In the display mode, encrypted regions in the file are visible to user. Encrypted regions appear as garbage and can be decrypted by supplying valid key.



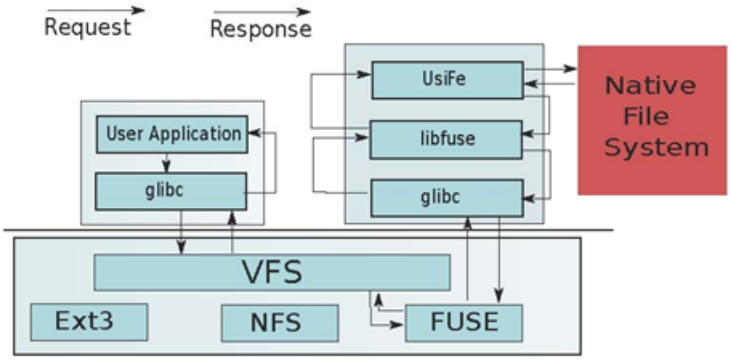
Interfacing with the Ext2 File System

UsiFe uses open source ext2fs library to interface with ext2 file system. UsiFe currently works on file system image. Ext2 file system can be created using 'mke2fs' command on Linux. Ext2 file system supports extended attributes for a file. In the inode structure of ext2 file system, there is a predefined attribute data pointer, 'i_file_acl'. This is meant to be used to implement sophisticated access control lists. We use this functionality to save encryption and decryption data instead of access control lists. We make 'i_file_acl' to point to metadata. Initially 'i_file_acl' points to only one 4KB block. If one 4kB block is not sufficient, then we put a pointer to another 4kB block at the end of previous block. In this manner, we chain the UsiFe metadata blocks together.



File system operations

UsiFe uses FUSE to intercept file system calls and performs basic operations on file system using ext2fs library.



The user application, does not perceive the fact that it is using a different file system. It makes regular system calls through the standard C libraries (glibc). The standard C libraries pass on the request to the kernel VFS layer in the Linux kernel. The VFS (Virtual File System) layer abstracts the functionality of a file system. It provides generic calls like open, read, write, and create to work with files.

Metadata Format

For every encrypted region of file we store logical block number and also start and end of the region.

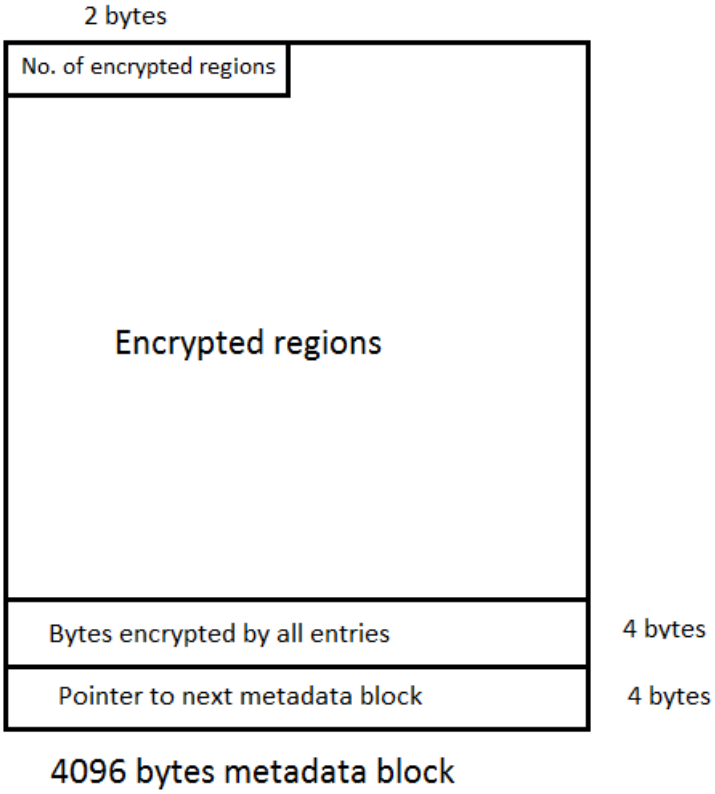
This structure

```
struct region{
    int block;
    short start;
    short end;
   };
```

defines an encrypted region.

Since size of structure is 8 bytes, we can have maximum of 4096/8=512 encrypted regions in a single block. Also first two bytes of block store number of encrypted regions contained in the block, and so as to chain metadata blocks, we store 510 encrypted regions in a single block.

Also to speed up things we store no of bytes encrypted by all entries in the current block at near end of the block, just before the pointer to the next metadata block.



Installation

Dependencies:

- **FUSE**: 2.6 or newer.
- OpenSSL: Preferably latest version.
- **E2fsprogs**: Ext2 File System Utilities, old versions might also work.
- Judy library.
- gtk+-2.0

Make sure that you have installed the dependencies before compiling. After downloading the project folder enter the Code folder. You will see two folders inside.

- FileSystem: Contains file system code.
- lib: Contains code for library , a few test codes and copy script.

1. Enter “FileSystem” folder and run make. This would compile all the files and finally make an executable.
2. Enter “lib” folder and run make. This would compile all the files and finally make a library.
3. Now enter “FileSystem” folder. You will see an executable named “usife”. But we require an ext2 file system image before running it. (Make sure you don’t disturb the directory structure. There should be a directory named “Mount” inside the “FileSystem” folder)
4. To create an ext2 file system image run the following commands in sequence.
    - dd if=/dev/zero of=CentOS bs=10485760 count=1
    - mkfs -t ext3 -b 4096 CentOS
5. Run “make mount” and a new drive will be mounted. Press Ctrl-C in the terminal anytime you want to unmount the file system.
6. Now enter “lib” folder and run “make test”. Then run “make run” to run the test. It will create a small file with few encrypted characters and then show you output for different read modes.
7. There is small script in the “lib” folder “usife_mv.py”. It can be used to copy data in and out of the file system so that the metadata about encrypted regions is preserved. To test it run
   ./usife_mv.py ../FileSystem/Mount/test .
   in the “lib” folder.
   You will see a new file “test.zip.usife”. This is a zipped file containing two files
    - **data**: Contains file data.
    - **enc**: Contains information about encrypted regions.