# Designing a File System with Secure Deletion

Ahmed Hassan, Faculty Mentor Dr.Xiaowen Zhang, Department of Computer Science, CSI

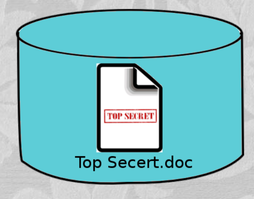
## Introduction

Most common file systems shipped with the operating systems do not support secure deletion. When a file is deleted, it just simply releases the allocated hard drive blocks taken by that file. It does not securely erase/overwrite the actual data from the hard drive. If a deleted file contains confidential information, it's very likely that the file can be recovered by some undelete programs. This is a big vulnerability from the security point of view.

A lot of research has been done on the similar topics. Ours is focused on a so called zero key management encryption file system, which eliminates the administration overhead. The file system decrypts/encrypts data when it opens/creates a file with cryptographic symmetric key algorithm. A master-key is stored on the extended attribute. When the file system is to read/write a file, the system gets the key from the extended attribute first and decrypts/encrypts the file on the fly. When a file is deleted, the key in the extended attribute is overwritten, and then the hard drive data blocks occupied by the file are released. It is infeasible for a undelete program to recover the deleted file, because the key is erased completely. Without the key to decrypt, the recovered data is illegible random bytes. With our file system, the aforementioned security vulnerability is prevented.

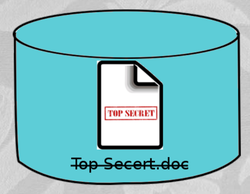
## Example.

A lot of people do not understand the risk of not securely deleting their data. For instance, when a user deletes a data, most of them think that the data has been deleted. However, that is not true. When the user requests to delete a data from the hard drive, most of the regular file system just release the allocated blocks from the hard drive. That can lead to a risk of the users' assets. An adversary can use recovering programs to restore deleted data.


## Deleted Files

There a lot of commercial and free open source programs having the capability to restore deleted data. In this paper we will explain how file recovering algorithms work. We will discuss the approaches that the previous works mentioned. We will talk about theirs and our algorithms efficiency.




The file stays there even after deletion

## recovering algorithms

There a lot of softeware that can recover the deleted data. All the software has to do is just look for all sectors in the file system, and try to recover recognized file types.

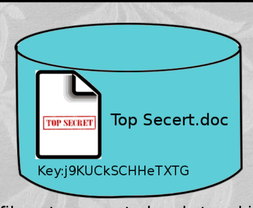

## Problem Statement

The problem with the current file system encryption is that users need to memorize key to encrypt and decrypt the hard drive. The users can be at risk from adversaries in the cases such as government agency can request a court warrant to reveal the key. Also, there is attack on the memory that can reveal master-key. Therefore, encrypting the whole hard drive with only one key is not efficient for secure deletion attack.

## Solution

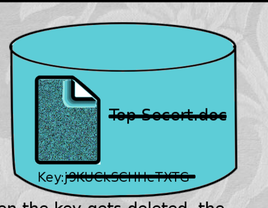
We used FUSE example fusexmp.c to add our changes. That file has a lot of default APIs already written. For instance, when open function gets called on FUSE file system, it overrides the original file system call and redirect to modified open function. We edited functions open( ), read( ) and write() to add encryption and automatic key management. Keys are generated automatically. When open() gets called, it will check if there is any key stored in extended attribute. If not, it will generate one and store on it. When write() or read() get called, they will read encryption keys from extended attributes, and encrypt or decrypt data from the hard drive.

1



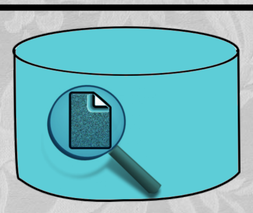

Each file gets encrypted and stored in a database with a diffrent key.

2




When the key gets deleted, the undelete programs cannot recover the file. That is because the file was encrypted with the deleted key.

3



When recovering software tries to recover the deleted files, all they will see is just random data.

This poster was made by: Ahmed Hassan, and Dr.Xiaowen Zhang
Ahmed@Linuxism.com and Xiaowen.Zhang@csi.cuny.edu

Sponsored by

STEAM

CSI CU NY COLLEGE OF STATEN ISLAND
The City University of New York

NSF