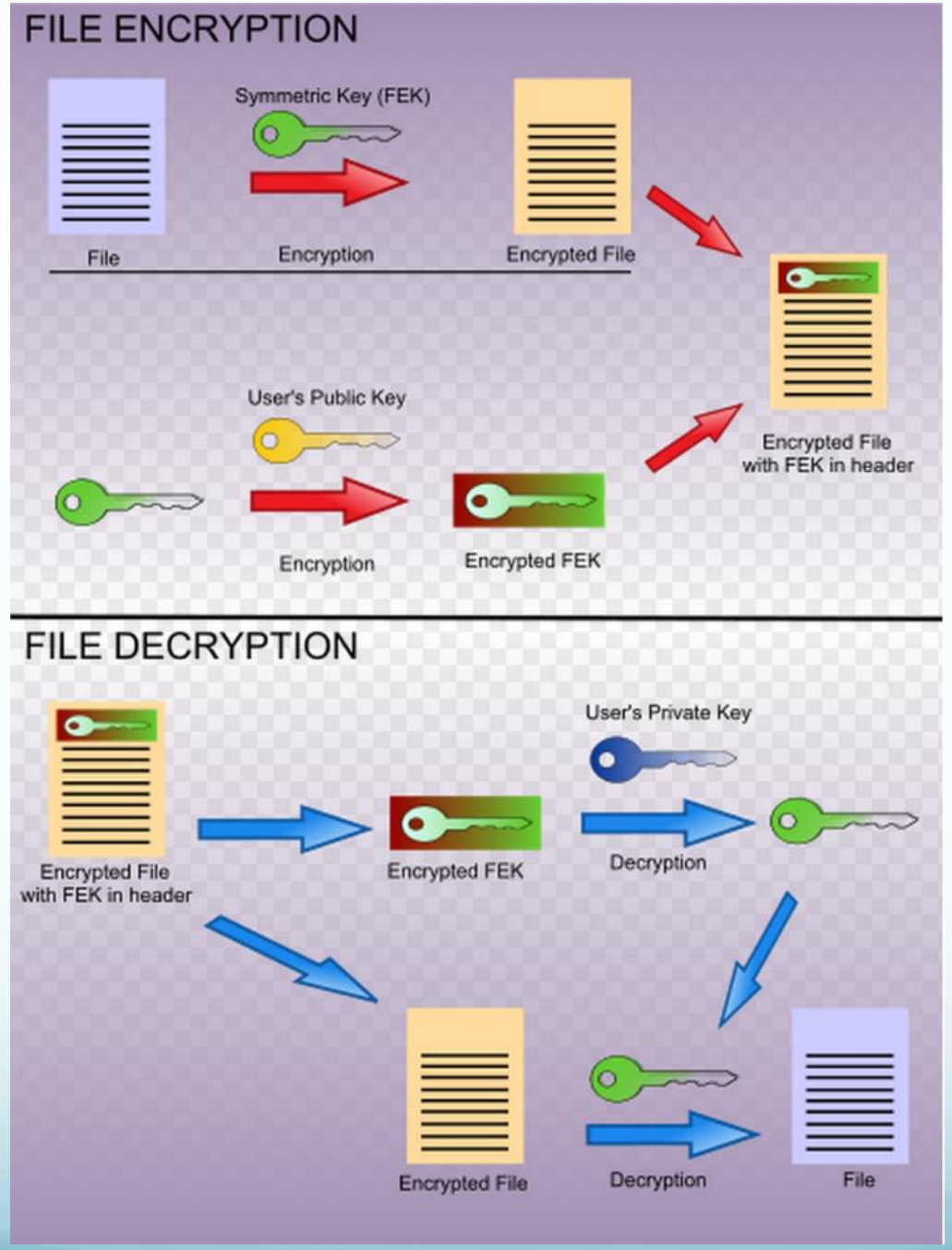
Project 1
Encrypted File System

TA: Yu-Yen Chung

Overview

- Traditional File System
  - Unencrypted
  - When the disks are stolen by someone, contents of those files can be easily recovered
- Encrypted File System (EFS)
  - Prevent such leakages
  - Files on disks are all encrypted
  - Nobody can decrypt the files without knowing the required secret

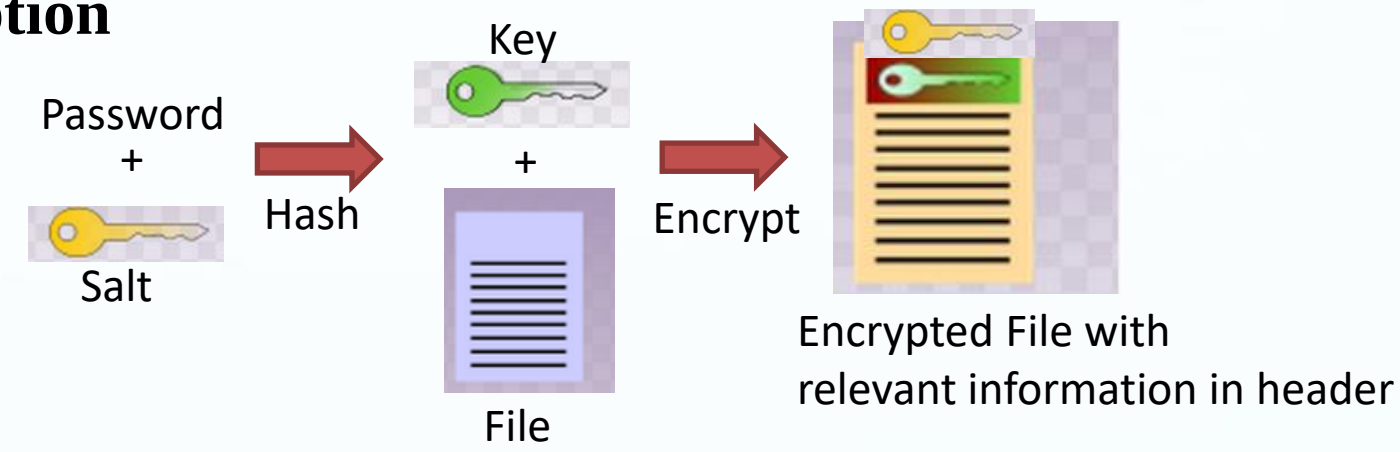
EFS Design



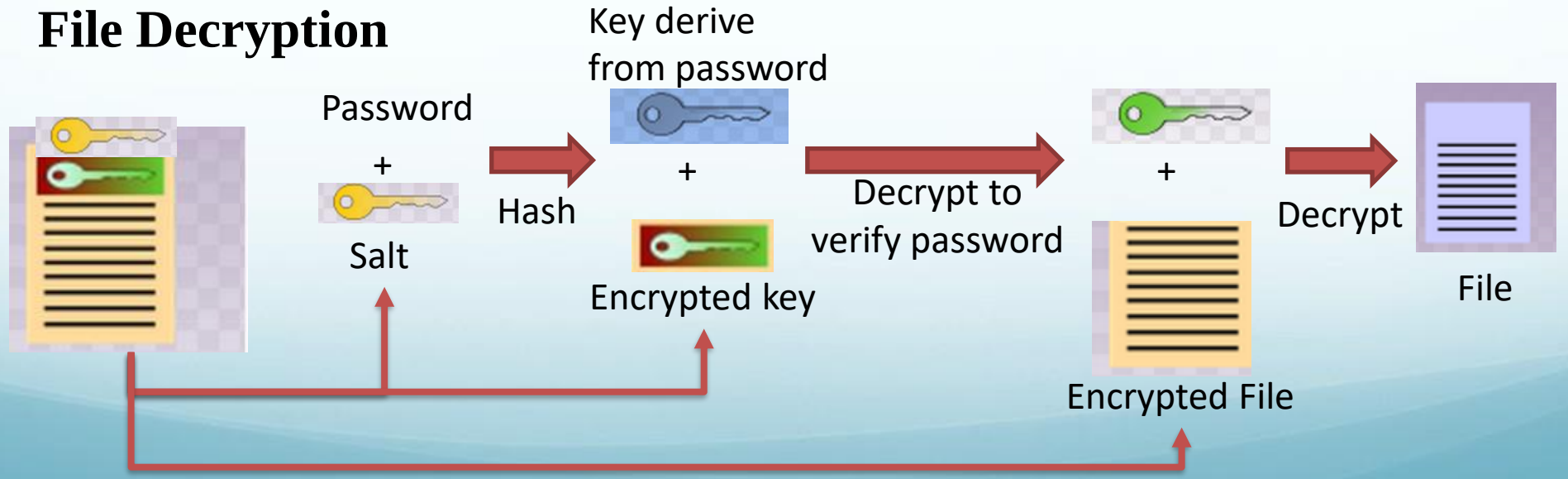
https://en.wikipedia.org/wiki/Encrypting_File_System

Example of Simplified Design

File Encryption

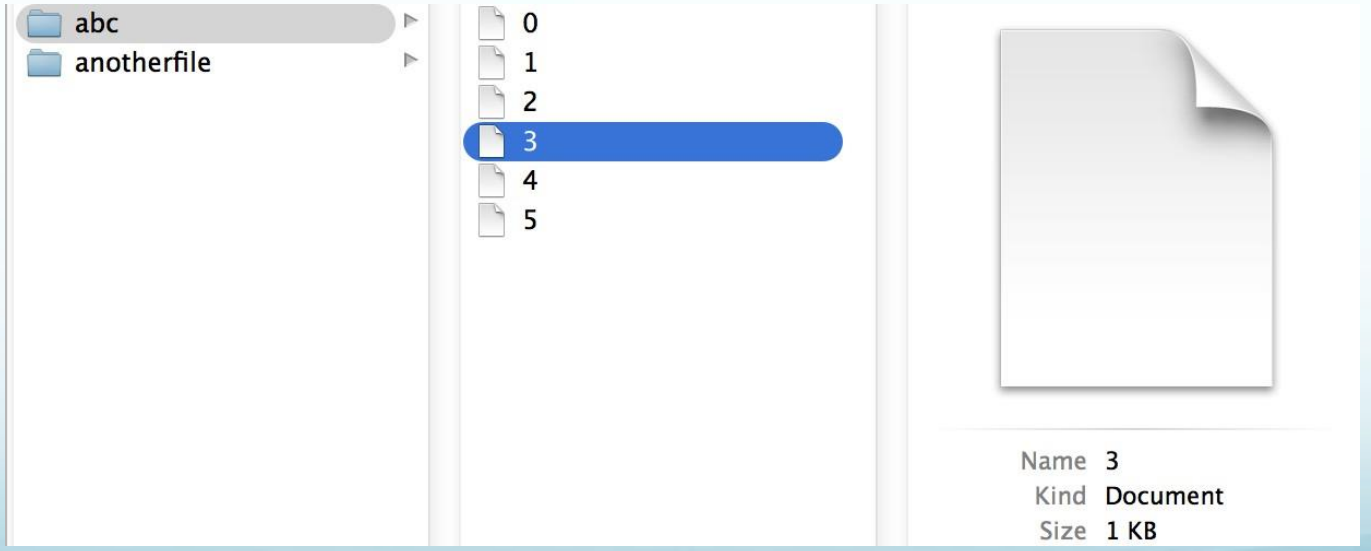


File Decryption



Overview

- Implement a simulated version of EFS in Java
  - To simulate a file system, files are stored in blocks of fixed size.



Overview

- You will need to implement several library functions, which simulates the functionalities of EFS

Functions to Implement

- void create(String file_name, String user_name, String password)
  - Create a file that can be opened by someone who knows both user_name and password. Both user_name and password are ASCII strings of at most 128 bytes.

Functions to Implement

- String findUser(String file_name)
  - Return the user_name associated with the file.
- int length(String file_name, String password)
  - Return the length of the file, provided that the given password matches the one specified in creation. If it does not match, your code should throw an exception.

Functions to Implement

- byte[] read(String file_name, int starting_position, int length, String password)
  - Return the content of the file for the specified segment, provided that the given password matches. If it does not match, your code should throw an exception.

Functions to Implement

- void write(String file_name, int starting_position, byte[] content, String password)
  - Write content into the file at the specified position, provided that the password matches. If it does not match, the file should not be changed and your code should throw an exception.

Functions to Implement

- void cut(String file_name, int length, String password)
  - Cut the file to be the specified length, provided that the password matches. If it does not match, no change should occur and your code should throw an exception.

Functions to Implement

- boolean check_integrity(String file_name, String password)
  - Check that the file has not been modified outside this interface. If someone has modified the file content or meta-data without using the write call, you should throw an exception.

Security Requirements

- Meta-data storage
  - Part of physical file
  - You will need to decide where to put such data
- User Authentication
  - If the password does not match, reading and writing is not allowed
  - Store something that is derived from the password

Security Requirements

- Encryption keys
  - Encrypt each file using a different key
    - reduce the amount of data encrypted under one key
  - Do NOT store encryption keys in plaintext
- Algorithm Choice
  - AES(128 bits = 16 bytes), CTR
  - An adversary may read the content of the file stored on disk from time to time
  - The adversary may observed the content on disk between modifications.

Security Requirements

- Message Authentication
  - Detect unauthorized modification to the encrypted files
  - MAC
  - How to combine encryption and MAC

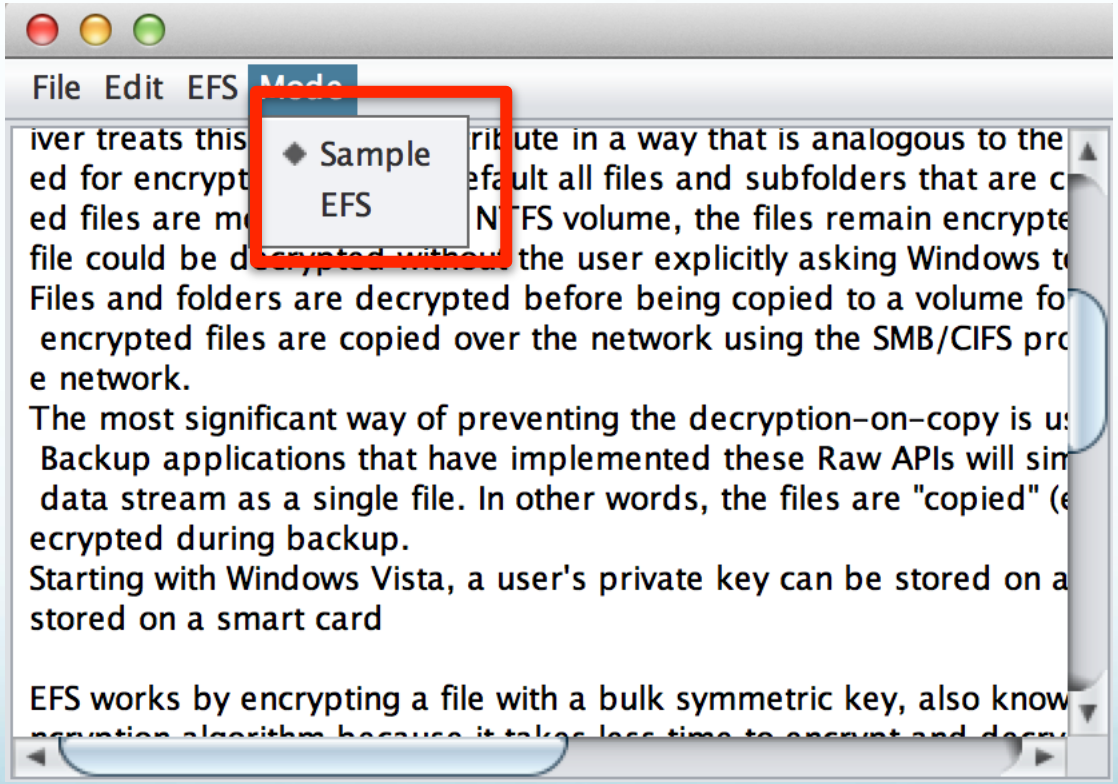
Efficiency Requirements

- Storage
  - Minimize the number of physical files used for each file.
- Speed
  - Minimize the number of physical files accessed for each read or write operation

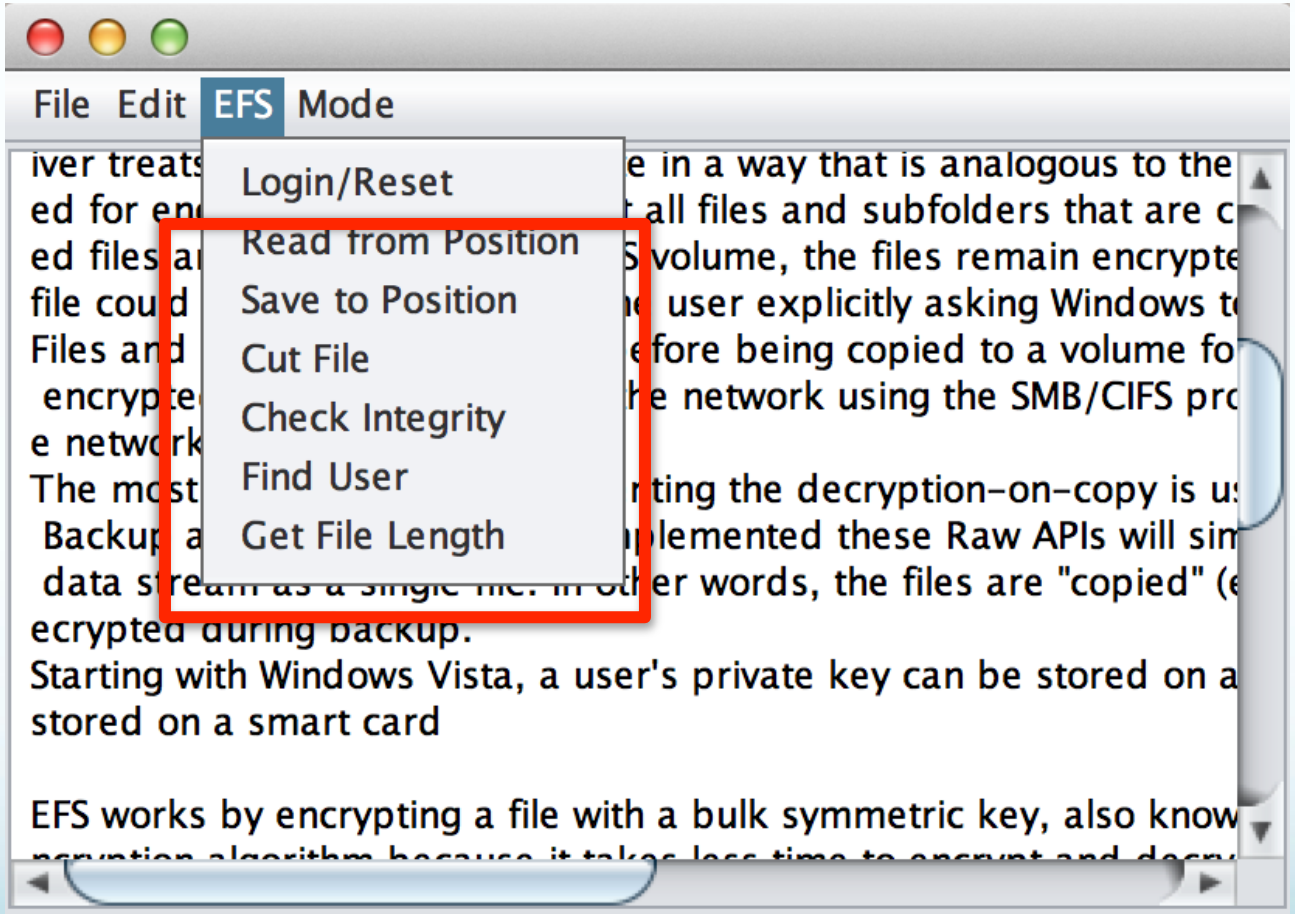
Note: In this project, size of each physical file is fixed to simulate the block on disk.

Instruction

- We provide an Editor to help you to verify your implementation



Instruction



Instruction

- byte[] encrypt_AES(byte[] plainText, byte[] key)
- byte[] decript_AES(byte[] cipherText, byte[] key)
- byte[] hash_SHA256(byte[] message)
- byte[] hash_SHA384(byte[] message)
- byte[] hash_SHA512(byte[] message)
- byte[] secureRandomNumber(int length)

Instruction

- Try NOT to modify the files other than EFS.java
  - You are supposed to provide library functions
  - If there is indeed a need, please explain your modification in detail
- Do NOT use the Encryption/Hash/MAC functions(from Java library or other open source script) other than what we provided

Submission

- EFS.java (and the other source files you created)
  - If you have modified the other provided scripts, please submit all the source files
- Report
  - Description of your design and implementation
  - Answers for Design variation and Paper Reading

Questions?

- Check instruction sheet carefully
- Try to use Questions channel on MS Team to discuss if possible