CS6348 Course Project:
Secure File Vault for Cloud Storage with GPU Acceleration

Jihye Choi Wenqing Jiang Guihong Wan Husheng Zhou*

1 Introduction

Cloud storage applications have been widely used by enterprises and individuals for its convenience of automatic file synchronization and backup. However, this convenience brought potential data privacy concerns. For example, once the password of the cloud storage application is leaked, then all files involved are exposed to the attacker, especially when all files stored in the cloud storage are in plain text. Thus an additional layer of protection that automatically encrypting the files before storing it in the cloud and decrypting them when users access them are urgently needed, to further protect the files stored while not negativing the convenience it brought to us. However, introducing additional encryption and decryption operations may bring obvious drawbacks in performance. Thus we adopted Graphics processing units (GPUs) to provide speedups for the encryption and decryption operations. GPUs are being widely used as co-processors for performance acceleration due to its highly multi-threaded architecture in many application domains including big data analytics and machine learning.

In this project, we designed and developed a secure file vault software for cloud storage to provide extra protection for files stored in cloud storage. Considering the drawbacks of introducing additional encryption and decryption operations, we realized GPU-accelerated methods to dramatically improve the performance, including GPU-accelerated DES, AES, and RSA. In the following sections, we first illustrate the adversarial scenario and challenges we faced in our project (Section 2), then highlight the design and implementation details (Section 3), and show the application usage and performance evaluations (Section 4).

*Names are in alphabetical order

2 Threat Model and Challenges

2.1 Threat Model

We assume that the password of the victim's cloud storage application (e.g., Google drive, One drive, or Dropbox) has been leaked to an attacker. The attacker can login to the victim's cloud applications anytime and anywhere. The simple scenario is that victim and attacker use different physical machine. The advance scenario is that attacker and victim resident at the same physical machine as normal users without root privilege. For example, attacker and victim are all UTD student using the cs1 server with their own account, the attacker has got the victim's Dropbox password, and wants to reveal the content of files stored in Dropbox.

As a normal user, the attacker can access all folders and intermediate files that defaultly set to be publicly shared. Also, the attacker can access the underlying GPU hardware using CUDA [5] APIs. As a consequence, the attacker can access the GPU hardware through legal APIs to read the residual data left by victim on GPU. We note that our secure file vault only prevent direct plain text leakage incurred by using GPU acceleration, other types of threat such as side-channel attacks [3, 2, 1, 4], are out of our scope of this project.

2.2 Challenges

Identifying trustful and untrustful objects. In our project the first challenge we faced is how to identify which objects are potentially to be exposed to attackers.

Protecting security key. Users need to use his password to login to the security vault, and each vault may be encrypted with different encryption keys.

Using GPU to accelerate. Implementing GPU encryption algorithms are not easy, need to maximize the parallelism while guaranteeing the correctness.

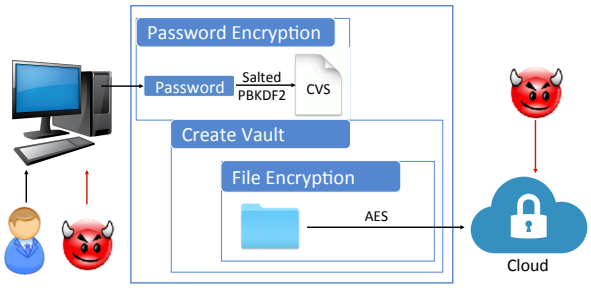
Protecting from GPU-incurred leakage. Current GPU hardware and systems software including GPU device drivers, compilers, and operating systems do not implement proper memory protection mechanisms due to performance and proprietary reasons, which has consequently caused severe security vulnerabilities such as information leakage.

3 Design and Implementation

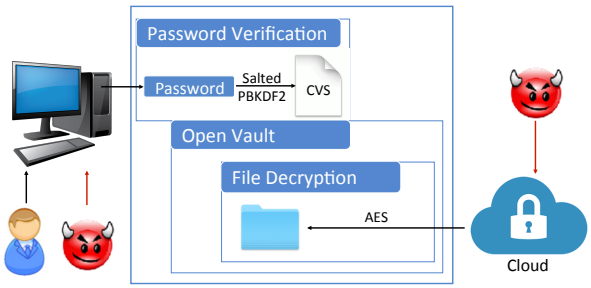
The framework of our software is shown as Figure 1, where users create secure file vault to automatically encrypt files before uploading it to the cloud storage (Figure 1 (a)), and automatically decrypt the encrypted files at usage. Each file vault is protected by a password, files under this vault is encrypted with the same secure key. At decryption phase, users need to input his password for a specific secure vault to pass the verification. Once verification passed, the files in the secure vault can be automatically decrypted.

The strength of password is enforced in our software to contain at least eight characters including lower and upper case and special characters. The hash value of the secure key is used to verify the user input password. Only when the verification is passed, user can decrypt the files in cloud storage. All his further operations for encryption and decryption does not need to re-input password for convenience.

Timeout of two minutes is set to reduce the attack surface, that is, a watch service is running to check



(a)



(b)

Figure 1: Overview of secure file vault: (a) encryption before uploading file to cloud storage (b) decryption at downloading

the activity of the software, if user does not operate on the file vault for more than two minutes then re-verification is enforced.

Considering the attacker can be another normal user residents on the same physical machine with victim, the file vault and intermediate plain text should be enforced with operating system's privilege isolation, which prevent other users to access his files.

Also considering the potential information leakage introduced by GPU, the GPU code is enforced with data cleaning after its usage. For example, before deallocating the memory objects on GPU, each object is zero-filled. Within each GPU kernel execution, the variables stored in GPU shared memory are also cleaned by assigning zero to the array elements.

The project is realized using Java language with 2500+ lines of code, and 5000+ line of CUDA code. The major framework is written with Java code.

The GPU acceleration parts are written with CUDA and compiled into dynamic library. Then the Java framework call the functionalities of GPU acceleration by using JNI (Java native interface).

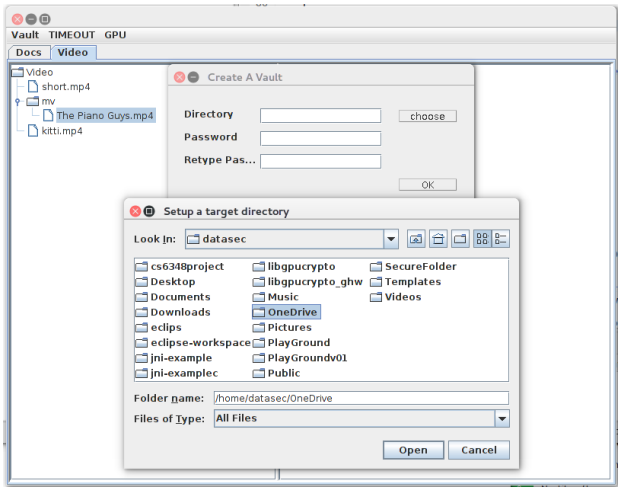
4 Application Usage

As illustrated in Figure 2, (a) shows the screen of creating a secure vault, where user can select the local folder that he wants to protect (here we use One Drive cloud storage), and need to setup the password for this vault. Figure 2 (b) shows user need to enter his password for current vault to decrypt files. Once verification passed, he can drag the files from left panel to the right panel to access the plaintext. The decrypted files are put in a secure folder that can only be accessed by current user and will be destroyed and cleaned after software termination. Figure 2 (c) illustrates the files in the left panel as ciphertext, and the decrypted plaintext on the right panel.

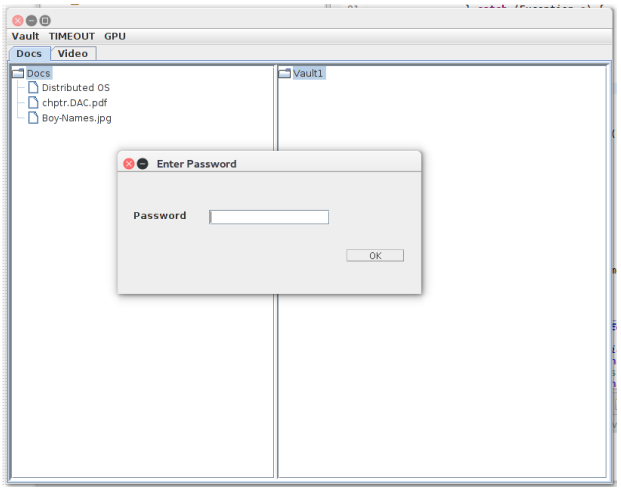
Table 1 illustrate the performance gain of our GPU acceleration using DES encryption. We use three different type of files including text, image, and video. We observe significant performance improvement using GPU for up to 20x speedup.

5 Division of Labor

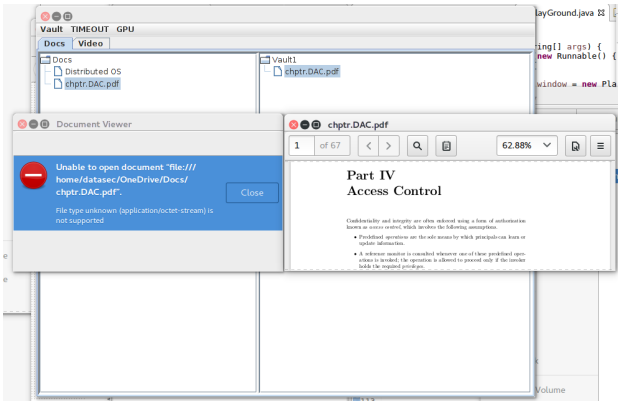
Jihye Choi: Documentation writing and integration testing.
Wenqing Jiang: Design the major framework and developed the Version 1.0 with CPU encryption/decryption.
Guihong Wang: Developed the UI and improved the functionalities, did unit testing and integration testing.
Husheng Zhou: Implemented the GPU acceleration for DES, AES, and RSA algorithms using CUDA, and provided java interfaces by loading the dynamic library with JNI. And documentation revision.



(a)



(b)



(c)

Figure 2: Screenshots of software

References

[1] JIANG, Z. H., FEI, Y., AND KAELI, D. A complete key recovery timing attack on a gpu. In *High Performance Computer Architecture (HPCA), 2016 IEEE International Symposium on* (2016), IEEE, pp. 394–405.

Table 1: Performance improvement of GPU DES encryption

Name	Size	Type	CPU	GPU	Speedup
README.txt	2.7KB	Text	10ms	6.15ms	1.63x
cat.jpg	96.1KB	Image	220ms	12.5ms	17.6x
KITTI.mp4	56.7MB	Video	28209ms	1379ms	20.46x

[2] JIANG, Z. H., FEI, Y., AND KAELI, D. A novel side-channel timing attack on gpus. In *Proceedings of the on Great Lakes Symposium on VLSI 2017* (2017), ACM, pp. 167–172.

[3] KADAM, G., ZHANG, D., AND JOG, A. Rcoal: Mitigating gpu timing attack via subwarp-based randomized coalescing techniques. In *High Performance Computer Architecture (HPCA), 2018 IEEE International Symposium on* (2018), IEEE, pp. 156–167.

[4] LUO, C., FEI, Y., LUO, P., MUKHERJEE, S., AND KAELI, D. Side-channel power analysis of a gpu aes implementation. In *Computer Design (ICCD), 2015 33rd IEEE International Conference on* (2015), IEEE, pp. 281–288.

[5] NVIDIA. CUDA 4.2. `https://developer.nvidia.com/cuda-toolkit-42-archive`, 2012.