PDF 2.0 CRACKING on CUDA

1st HyunJun Kim
dept. of Information Computer Engineering
Hansung University
Seoul, South Korea
khj930704@gmail.com

2nd SiWoo Eum
dept. of IT Convergence Engineering
Hansung University
Seoul, South Korea
shuraatum@gmail.com

3rd Hwajeong Seo
dept. of IT Convergence Engineering
Hansung University
Seoul, South Korea
hwajeong84@gmail.com

***Abstract*—The latest graphic processor provides high processing power, and research using the graphic processor for cryptographic processing is being actively conducted through the groundbreaking performance of the GPU. PDF (Portable Document Format) is a file format developed by Adobe in 1992, standardized as ISO 32000 and used worldwide. Mainly used files such as PDFs can be subject to password cracking. In this paper, an optimal implementation on CUDA GPU is implemented for PDF 2.0 cracking. Optimization of the AES and RSA-2 algorithms used for cracking was optimized. In addition, CUDA GPU shared memory and command functions were used. As a result, we achieved the performance of calculating approximately 76,405 passwords per second in the RTX 3060 environment.**

***Index Terms*—CUDA implementation, PDF 2.0 cracking, , AES, SHA-2**

I. Introduction

The latest graphic processor provides high processing power, and research using the graphic processor for cryptographic processing is being actively conducted through the groundbreaking performance of the GPU. Password cracking refers to the operation of recovering passwords from data stored or transmitted in a computer system. It is used to help users recover forgotten passwords, gain unauthorized access to systems, or access digital evidence that judges have granted access to. In general, before decrypting the contents of an encrypted file, a hash value is included to check the correctness of the password. Therefore, the attacker compares the guessed password with the hash value inside the file by repeatedly attempting a brute force attack without having to attempt to decrypt the entire file. Popular cryptocurrency tools include Hashcat and John the ripper. In this paper, the algorithm of PDF 2.0 version decryption was optimized on CUDA GPU. PDF (Portable Document Format) is a file format developed by Adobe in 1992 to display documents, including text formats and images, in a way that is independent of application software, hardware, and operating systems. It has been standardized as ISO 32000 and has established itself as a de facto standard document along with the DOC file worldwide. The PDF file format has the advantage of being very flexible. All PDF versions are all standard and backward compatible. Mainly used files such as PDFs can be subject to decryption. In this paper, PDF 2.0 cracking algorithm is optimized for CUDA environment. The PDF 2.0 cracking algorithm uses the same algorithm from to. The AES algorithm and SHA-2 algorithm used in the algorithm were optimized and applied, and the elements of the GPU were optimized to make the most of it. As a result, we achieved the performance of calculating about 76,405 passwords per second in the RTX3060 environment.

Identify applicable funding agency here. If none, delete this.

II. Related Works

A. Password Cracking Algorithm

There are SHA-256, SHA-384, SHA-512, and AES-128 CBC mode encryption algorithms used in PDF 2.0 version decryption algorithm. PDF 2.0 The process of the version of the decryption algorithm is shown in Fig. 1.

- 48 byte data U is extracted from the PDF document. 32 bytes of U is a hash value for password verification. This value is used when verifying the password in the last step of the decryption algorithm. Among the remaining data of U, 8 bytes are used as salt values and entered as input of SHA-256 along with UTF-8 encoded password P.
- In the next step, round 64 is repeated. One of the AES-128 CBC and SHA-2 algorithms is used per round. First, CBC mode AES-128 operates. 16 bytes of K are used as key values and the remaining 16 bytes are used as IV values. The value with 64 $P\|K$ appended is entered as plain text. Next, the algorithm of SHA-2 operates. For the selection of the SHA-2 algorithm, the value of the first block of the ciphertext is modularly calculated as 3, and when it is 0, SHA-256, when it is 1, SHA-384, and when it is 2, SHA-512 is selected.
- The round proceeds until the last byte of the ciphertext E generated during the round is greater than round i - 32.
- Finally, compared with the verification value, if it is the same, the password is returned. If not, it returns to the first step of the algorithm and enters a different password.

The algorithm characteristically varies in execution time depending on the input password. The longer the password, the greater the number of AES-128 operations. And depending on the ciphertext generated from AES128-CBC, different SHA-2 algorithms are used and additional rounds can be performed.

B. AES

The Advanced Encryption Standard (AES) is an encryption method established by the National Institute of Standards and

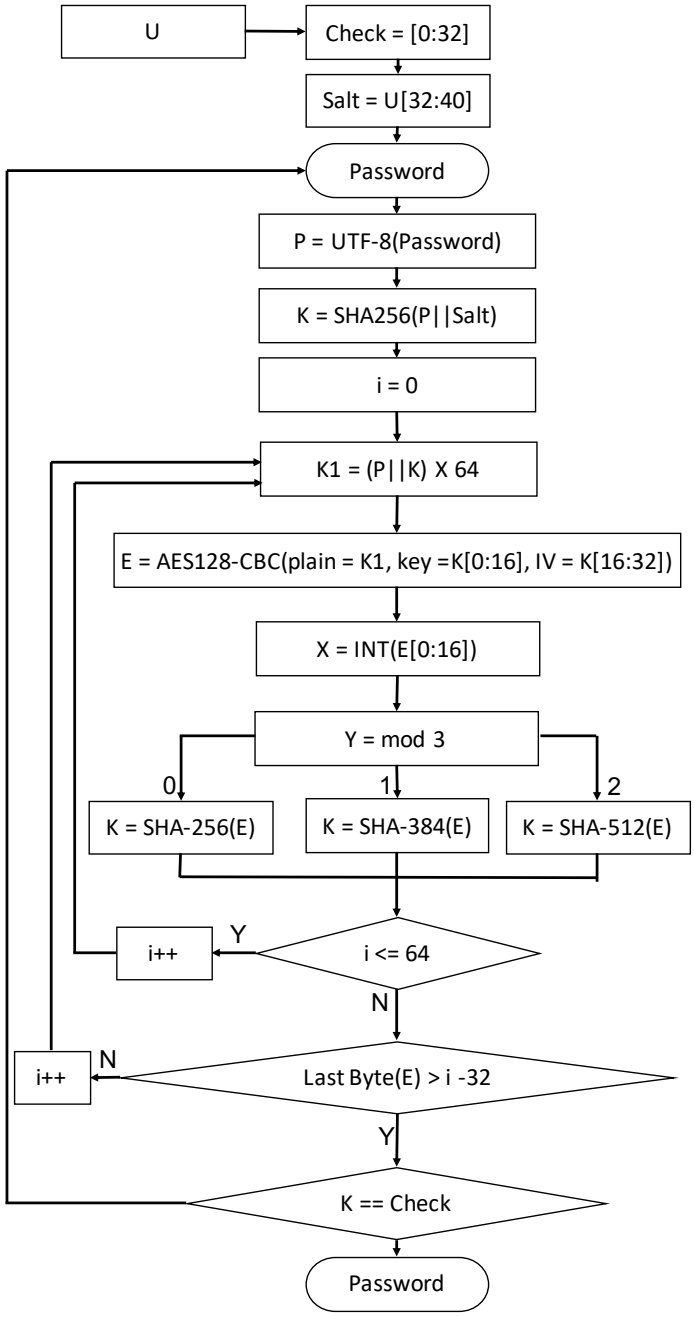

Fig. 1. Password cracking algorithm in PDF 2.0 version

Technology (NIST) in 2001 [1]. There are 128, 192 and 256-bit key versions and encrypts blocks of size 128 bits. One round of AES consists of SubBytes, ShiftRows, MixColumns and AddRoundKey operations.

- SubBytes(SB): Blocks are exchanged in byte unit format. An 8-bit S-Box is applied 16 times in parallel to each byte of state. If you have enough memory, you can create an S-Box, which is a table that calculates operations for all inputs, and use it to substitute a lot.
- ShiftRows(SR): Performs row-wise operation. The ith row is rotated left by i bytes.
- MixColumns(MC): Performs column-wise operation. Each column is multiplied by a constant 4 × 4 inverse matrix M.
- AddRoundKey(AK): 128-bit internal state is XORed with a 128-bit round key.

C. SHA-2

SHA-2 (Secure Hash Algorithm 2) is a set of cryptographic hash functions designed by the US National Security Agency (NSA) [2]. Consists of 6 hash functions with digests (hash values) of 224, 256, 384 and 512 bits: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512 /256. SHA-256 and SHA-512 are hash functions that use 32-byte and 64-byte words, respectively, and although some constants are different, their structures are identical except for the number of rounds. One compression function of SHA-256 is as follows. For SHA-512, bit rotation uses a different constant.

$$\mathrm{Ch}(E,F,G) = (E \wedge F) \oplus (\neg E \wedge G)$$
$$\mathrm{Ma}(A,B,C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$$
$$\mathrm{Ma}(A,B,C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$$
$$\Sigma_0(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$$
$$\Sigma_0(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$$
$$\Sigma_1(E) = (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)$$
$$\Sigma_1(E) = (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)$$

D. CUDA Programming

CUDA (Compute Unified Device Architecture) is a GPGPU technology that enables parallel processing algorithms performed on GPUs to be written using industry standard languages including C programming language. CUDA has been developed by Nvidia, and this architecture requires an Nvidia GPU and special stream processing drivers. The CUDA GPU structure includes a function kernel, thread group block, block group grid, 32 thread bundle warps, and SM (Streaming Multi-processor) that executes one warp and executes threads simultaneously. For decryption, one password is decrypted in one thread, and several decryption is performed in parallel. For performance, it is necessary to select appropriate values for the number of threads per block and the number of blocks per grid, and it is necessary to use sufficient threads and blocks in consideration of data transmission delay. A big advantage of CUDA is that it can access fast shared memory areas that can be shared between threads. The proposed method could show high performance improvement by using shared memory.

III. PROPOSED

The decryption algorithm uses AES, SHA-2. These algorithms are widely used algorithms, and many optimization studies have been conducted. The optimization technique of these studies was used. A recent study on cracking PDF 2.0 showed a 47x faster performance boost than CPUs in [3].In particular, Hashket provides the PDF 2.0 hacking algorithm as an open source. The proposed technique referred to the implementation of hashket. This chapter describes the applied optimization technique.

A. Shared Memory

The main optimal implementation techniques available for GPU implementation of AES are table-based, bit-slicing. In the proposed technique, table implementation was used. The fastest implementation of AES-128 is described by Hajihassani et al, achieving 1,478 Gbps on V100. It is an implementation of [4]. However, the bit slice technique is suitable for parallel processing modes such as ECB and CTR. Since the decryption algorithm uses the CBC mode, a table implementation was used. Many existing table-based implementations use the technique of storing T-tables in shared memory. Tezcan [5] achieved 878.6 Gbps throughput for AES-128 on the RTX 2070 Super GPU. [5] shows performance improvement by eliminating bank conflicts in the T-table. [6] completely eliminates bank conflicts that can occur in [5]. [6] proposed a warp-based technique for writing T-boxes to shared memory without bank conflicts. [6] uses more shared memory to eliminate bank conflicts and performs additional byte rotation on the output of the S-box. Hashcat uses an implementation technique without bank conflict resolution. Table 1 shows the implementation using shared memory for the proposed PDF 2.0 decryption algorithm. When shared memory was used, performance was improved by 23% compared to when not used. The code used is as follows.

```
__shared__ u32 s_te0[256];
__shared__ u32 s_te1[256];
__shared__ u32 s_te2[256];
__shared__ u32 s_te3[256];
__shared__ u32 s_te4[256];

  for (int i = threadIdx.x; i ¡ 256; i += blockDim.x)
{
s_te0[i] = te0[i];
s_te1[i] = te1[i];
s_te2[i] = te2[i];
s_te3[i] = te3[i];
s_te4[i] = te4[i];
}
__syncthreads()
```

B. __syncthreads() CUDA FUNCTION

__syncthreads() is a barrier synchronization function. Barrier synchronization is a simple and widely used method for coordinating parallel tasks. Waits for all global and shared memory accesses performed by a thread to be visible to all threads in the block. The thread that called __syncthreads() is stopped at the calling position, until all threads in the block reach that position. Additional __syncthreads() calls are often required at the end of the loop if there are reads and writes from different threads to the shared memory location. The proposed technique stores the T-tlable in shared memory. Therefore, __syncthreads() was added after saving T-tlable in shared memory. In addition to this point, the proposed technique improved performance by adding __syncthreads() to the iteration section in the decryption algorithm.

The PDF 2.0 version of the cracking algorithm selects the SHA-2 algorithm or performs additional iterations depending on the input and intermediate state values. This causes control divergence and consequently reduces the performance of the program. We wanted to reduce the control divergence, but we couldn't control it because the direction of the divergence that occurred would not be known until it was computed.

In branching, the hardware is computed as shown in Fig. 2. Because it has to go through both A and B, it causes control divergence and leads to poor performance. Here, the same operation must be performed in Warp's thread before and after branching. We wanted to clarify before and after branching. Added __syncthreads() to the iteration section in the decryption algorithm. Not sure if __syncthreads() reduced control divergence, but it did improve performance.

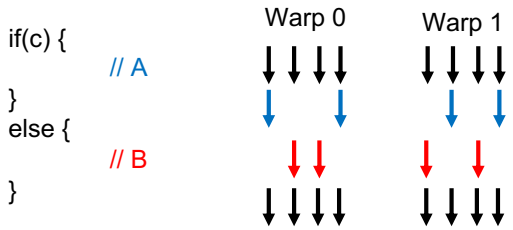

Fig. 2. Thread flow in warp when branching on hardware.

C. Block Size, Thread Size

The number of threads per block and the number of blocks per grid have an impact on performance. Therefore, proper selection is necessary to achieve optimal performance. First, to confirm this, we checked the performance change according to the number of threads per block and the number of blocks per grid. In the RTX 3060 environment, the number of password calculations per second was measured by operating the decryption implementation applying the proposed technique. Since the execution time of the algorithm may vary depending on the input, the same password value was input for accurate comparison. The number of operations per second was measured by fixing the block size and increasing the thread size by 64 from 64 to 512. As shown in Fig. 3, 256 showed the best performance, and it was confirmed that the performance was irregular depending on the thread size. Next, the number of operations per second was measured by fixing the thread size and increasing the block size by 2 times from 256. As shown in Fig. 4, the higher the block size, the higher the performance. And it converges when the value increases to some extent.

RESULTS

It was implemented in the Visual Studio environment in RTX 3060, and the performance was measured by adjusting the number of threads per block and the number of blocks per grid. As a result, the higher the number of threads per

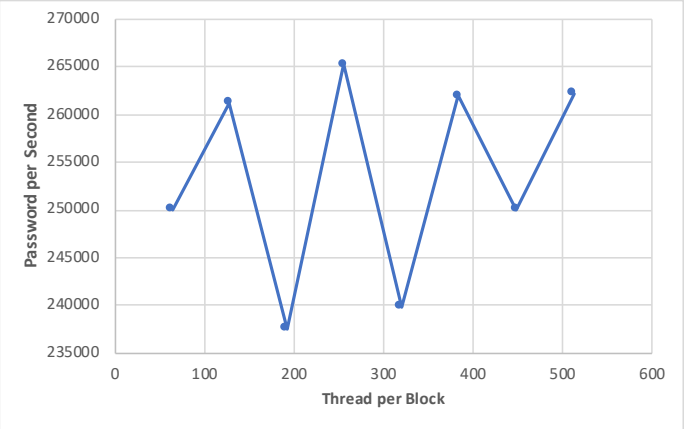

Fig. 3. When cracking PDF in RTX 3060 environment, the number of password operations per second according to the number of threads per block (kp/s)

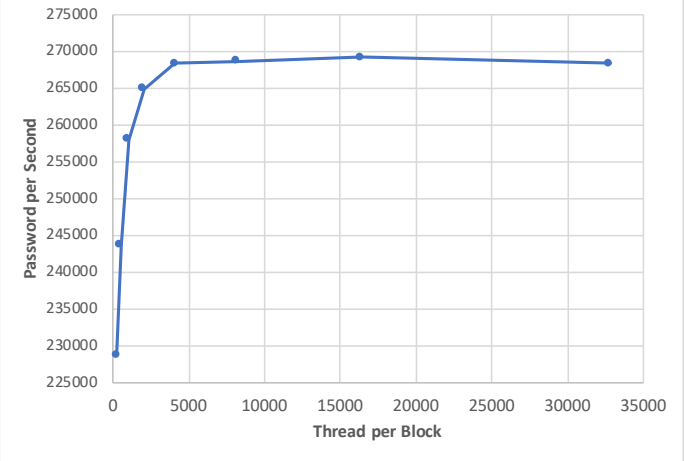

Fig. 4. When cracking PDF in RTX 3060 environment, the number of password operations per second according to the number of blocks per grid (kp/s).

block and the higher the number of blocks per grid, the higher the performance. As a result, the RTX 3060 showed the best performance, processing approximately 76,405 passwoards per second with 4048 blocks per grid and 512 threads per block. And the performance improvement when using shared memory and __sysncthreads(), which is the main optimization technique, was compared together. The results are shown in Table I. Performance improvement of 23% was shown when shared memory was used. When __sysncthreads() was used, the performance improvement of 29% was shown. And when shared memory and __sysncthreads() were used, 49% performance was improved.

CONCLUSION

This paper optimizes the PDF 2.0 version decryption algorithm on CUDA GPU. The optimal implementation technique using shared memory and __sysncthreads() was applied, and the GPU optimization factor was used as much as possible.

TABLE I
THE NUMBER OF PASSWORD CRACKING PER SECOND AND PERFORMANCE DIFFERENCE ACCORDING TO OPTIMIZATION

Optimization	Speed	Improvement
None	51,241 p/s	0%
shared memory	63,030 p/s	23%
__sysncthreads()	66,377 p/s	28%
shared memory +__sysncthreads()	76,405 p/s	49%

High performance improvement was shown by using the values of the number of threads per block and the number of blocks per grid close to the optimal performance. As a result, the RTX 3060 achieved about 76,405 Performance improvement of 49% was achieved compared to before optimization. In future research, we intend to implement performance comparison in more diverse environments and optimization of other decryption algorithms.

IV. ACKNOWLEDGEMENT

This work was partly supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2018-0-00264, Research on Blockchain Security Technology for IoT Services, 25%) and this work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2021-0-00540, Development of Fast Design and Implementation of Cryptographic Algorithms based on GPU/ASIC, 50%) and this work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2022-0-00627, Development of Lightweight BIoT technology for Highly Constrained Devices, 25%).

REFERENCES

[1] J. NECHVATAL,, et al. Report on the development of the Advanced Encryption Standard (AES). Journal of research of the National Institute of Standards and Technology, 2001, 106.3: 511.
[2] National Institute of Standards and Technology. FIPS PUB 180-2: Secure Hash Standard.
[3] F. YU, H. YIN, Password Cracking of PDF 2.0 documents on GPU. In: 2021 IEEE 6th International Conference on Computer and Communication Systems (ICCCS). IEEE, 2021. p. 721-725.
[4] O. HAJIHASSANI, et al. Fast AES implementation: A high-throughput bitsliced approach. IEEE Transactions on parallel and distributed systems, 2019, 30.10: 2211-2222.
[5] C. TEZCAN. Optimization of advanced encryption standard on graphics processing units. IEEE Access, 2021, 9: 67315-67326.
[6] W. LEE , et al. Efficient Implementation of AES-CTR and AES-ECB on GPUs With Applications for High-Speed FrodoKEM and Exhaustive Key Search. IEEE Transactions on Circuits and Systems II: Express Briefs, 2022, 69.6: 2962-2966.