CUDA GPU 상의 PDF 1.4-1.6 해독 최적 구현

김현준[1†], 엄시우[2], 서화정[2‡]
[1]한성대학교 정보컴퓨터 공학과
[2]한성대학교 IT융합 공학과
khj93072004@gmail.com,

PDF 1.4-1.6 Passward Cracking Optimal Implementation on CUDA GPU

Hyun-Jun Kim[1†], Si-Uoo Eum[2], Hwa-Jeong Seo[2‡]
[1]Dept. of Information and Computer Engineering, Han-sung University
[2]Dept. of IT Convergence Engineering, Han-sung University

요 약

PDF (Portable Document Format)는 1992년 Adobe 에서 개발한 파일 형식으로 ISO 32000 으로 표준화 되어 전세계적으로 사용되고 있다. PDF와 같이 주로 사용되는 파일은 암호 해독(Password Cracking)의 대상이 될 수 있다. 본 논문에서는 PDF 1.4-1.6 암호 해독을 위해 CUDA GPU 상의 최적 구현하였다. 암호 해독에 사용되는 MD5와 RC4 알고리즘의 최적화와 CUDA GPU의 요소를 사용하였으며 RTX 3060 환경에서 크래킹 도구 해시캣과 비교하여 22.5%의 성능 향상을 달성하였다.

1. 서론

PDF(Portable Document Format)[1]는 텍스트 서식 및 이미지를 포함한 문서를 응용 프로그램 소프트웨어, 하드웨어 및 운영 체제 와 독립적인 방식으로 표시 하기 위해 1992년 Adobe 에서 개발한 파일 형식이다. ISO 32000 으로 표준화되었으며 전세계적으로 DOC파일과 함께 표준(de facto standard) 문서로 자리 잡았다. PDF 파일 형식은 매우 유연한 장점을 갖는다. 모든 PDF 버전은 모두 표준이며 이전 버전과 호환된다. PDF와 같이 주로 사용되는 파일은 암호 해독의 대상이 될 수 있다. 암호 해독(Password Cracking)은 컴퓨터 시스템 에 저장되거나 전송된 데이터에서 암호를 복구하는 작업을 말한다. 사용자가 잊어버린 암호를 복구하도록 돕거나, 무단으로 시스템의 접근 권한을 얻거나, 판사가 접근을 허용한 디지털 증거에 액세스하기 위해 사용한다. 일반적으로 암호화된 파일의 내용을 복호화하기 전에 비밀번호의 정확성을 검사하기 위해 해시를 포함 한다. 전체 파일의 복호화를 시도할 필요 없이 사용 가능한 암호의 해시와 암호 추측을 비교하여 반복적으로 무차별 대입 공격을 시도한다. 유명 암호 해독 도구로 Hashcat, John the ripper가 있다. 본 논문은 PDF 1.4 - 1.6 버전 암호 해독의 알고리즘을 CUDA GPU 상에서 최적화하였으며 RTX 3060 환경에서 크래킹 도구 해시캣과 비교하여 22.5%의 성능 향상을 달성하였다.

2. 관련 연구

2.1 MD5

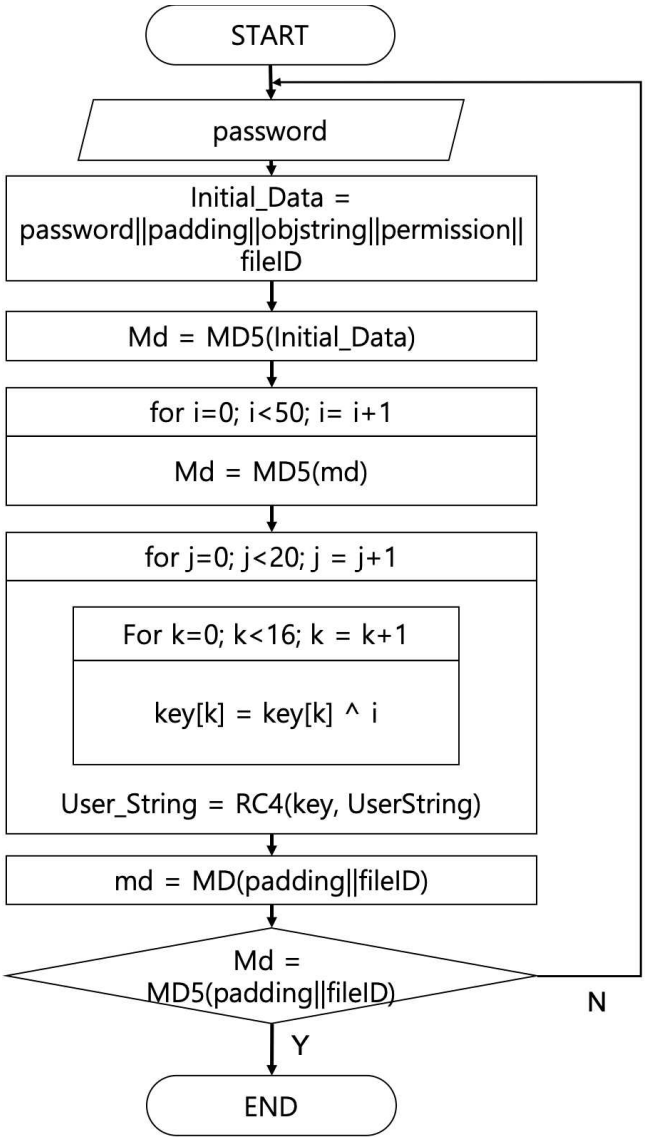
PDF 1.4 - 1.6 버전의 암호 해독 알고리즘은 MD5 알고리즘[2]과 RC4 알고리즘[3]을 반복적으로 사용한다. MD5 알고리즘은 임의의 길이의 메시지를 입력받아 128 비트의 값 을 생성하는 암호화 해시 함수이다. 입력 메시지는 512-비트 블록들로 나뉘어 차례로 수행된다. 32-비트 워드 4개로 이루어진 128 비트 state로 동작 한다.

2.2 RC4

RC4 알고리즘은 스트림 암호로 키 스케줄링 알고리즘(Key-Scheduling Algorithm, KSA)과 유사 난수 생성 알고리즘(Pseudo-Random Generation Algorithm, PRGA)로 구성된다. 256 바이트 의 순열과 2개의 8비트 인덱스 포인터로 KSA를 사용하여 40~2048 비트 사이 의 가변 길이 키로 초기화 된다. 이후 PRGA로 비트 스트림이 생성된다.

2.3 PDF 암호 해독 알고리즘

PDF 1.4 - 1.6 버전의 암호 해독 알고리즘은 그림 1과 같으며 크게 3단계로 나눌 수 있다. 첫 번째 단계에서 추측 패스워드 값은 패딩 값과 파일의 메타 데이터(objstring, permisssion, fileID)와 결합되어 MD5 알고리즘으로 해싱된다. 다음으로 해싱된 128비트의 값이 MD5 알고리즘으로 50번 반복적으로 해싱된다. 두 번째 단계는 50회의 RC4 암호화 과정이 수행된다. 이전 과정에서 생성된 128비트의 해싱된 값에 상수 값을 XOR한 값을 RC4 암호화의 키 값으로 사용한다. 그리고 파일의 User_String값을 RC4를 사용하여 20회 반복적으로 암호화한다. 3단계 과정에서는 파일의 fileID값이 MD5 알고리즘으로 해싱된 값과 이전 과정에서 생성된 암호문을 비교하여 패스워드 검증한다. 올바른 패스워드 값이 나올 때 까지 이전 단계를 반복한다.



(그림 1) PDF 1.4 - 1.6 버전의 암호 해독 알고리즘

2.4 CUDA 프로그래밍

CUDA(Compute Unified Device Architecture)는 GPU에서 수행하는 병렬 처리 알고리즘을 C 프로그래밍 언어를 비롯한 산업 표준 언어를 사용하여 작성할 수 있도록 하는 GPGPU 기술이다. CUDA는 엔비디아가 개발해오고 있으며 이 아키텍처를 사용하려면 엔비디아 GPU와 특별한 스트림 처리 드라이버가 필요하다. CUDA GPU 구조로는 GPU에서 실행되는 함수 커널, 스레드 그룹 블록, 블록의 그룹 그리드, 32개의 스레드 묶음 워프, 1개의 워프가 실행되어 스레드를 동시에 실행하는 SM (Streaming Multi-processor)이 있다. 암호해독을 위해서는 하나의 스레드에서 하나의 패스워드를 해독하며 여러 개의 암호해독이 병렬로 이루어진다. 성능을 위해 블록당 스레드 수, 그리드당 블록 수의 적절한 값 선택 필요하며 데이터 전송 지연을 고려하여 충분한 스레드와 블록 사용이 필요하다. CUDA의 큰 장점으로 스레드 간에 공유할 수 있는 빠른 공유 메모리 영역에 접근 할 수 있다. 제안 기법에서는 공유 메모리를 사용하여 높은 성능 향상을 보일 수 있었다.

3. 제안기법

제안 기법은 PDF 1.4 - 1.6 버전 암호 해독 알고리즘에 주로 사용되는 MD5 알고리즘과 RC4 알고리즘에 대한 최적화와 성능향상의 영향을 미치는 블록, 스레드 수 조정을 고려하였다.

3.1 MD5 최적화

32-비트 워드 a, b, c, d로 이루어진 128 비트 state로 동작하며 첫 는 상수 값으로 초기화 후 사용된다. 따라서 첫 라운드에서 일부 연산이 사전 연산 가능하다. 아래는 첫 라운드 연산 과정의 의사코드이다.

```
a += K;
a  = a +x + f (b, c, d);
a  = rotl32 (a, s);
a += b;
```

여기서 x를 제외한 a, b, c, d, K, s의 값은 상수값이므로 아래와 같이 변경 될 수 있다.

```
a = rotl32(x + 0xd76aa477, s) + 0xefcdab89;
```

다음으로는 50번의 MD5 반복 연산에 대한 최적화를 적용하였다. 128-비트에 대한 해시 연산이 반복적을 수행 되기 때문에 512-비트 입력 메시지 M ($M\_0, M\_1, \cdots, M\_30, M\_31$)에서 M_0, M_1, M_2, M_3 만 변경된다. M_0, M_1, M_2, M_3를 제외한 나머지 값들은 변경되지 않으므로 해당 메시지 워드의 덧셈 연산

생략할 수 있다.

3.2 RC4 최적화

RC4는 8비트 프로세서에 최적화되어있기 때문에 일반적으로 8비트 워드 연산으로 구현된다. 첫 번째 RC4 최적화 제안 기법은 RC4 알고리즘을 32비트의 워드로 통합하여 연산한다.

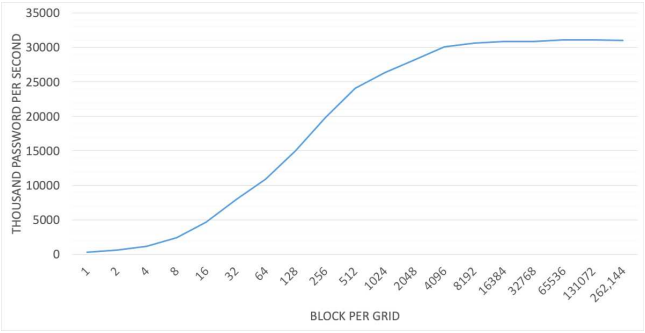
RC4 알고리즘 과정에서는 MD5연산의 결과로 생성된 128비트를 키 값으로 사용한다. MD5연산은 32비트 워드로 연산이 진행되어 GPU의 32비트 프로세서와 일치하며 128비트를 키 값은 4개의 32비트 워드 상태로 저장된다. 따라서 8비트 워드로 변환 없이 RC4 이전 MD5 과정의 출력 값을 32비트 워드 그대로 사용 가능하다. 또한 결합된 만큼의 덧셈 및 XOR 연산 횟수가 감소된다.

RC4 알고리즘은 KSA와 PRGA에서 반복적으로 상태 배열 S를 조회한다. 메모리에서 값을 불러오고 저장하는 과정은 많은 시간을 소비한다. 이를 개선하기 위한 두 번째 RC4 최적화 제안 기법은 상태 배열 S를 공유메모리에 저장하여 상태 배열 S 조회 및 저장에 생기는 지연 최소화이다. 각 스레드에서 하나의 암호 해독 알고리즘이 동작하므로 하나의 상태 배열 S 256 바이트가 필요하다. 따라서 공유 메모리 또한 각각의 스레드마다 상태 배열 S를 저장할 256 바이트가 필요하다. 이러한 점 때문에 커널에서 사용 가능한 공유 메모리의 크기를 넘지 않도록 블록당 스레드의 수를 조절 해야한다.

이때 공유 메모리는 정적 혹은 동적으로 할당 가능하다. GPU 아키텍처마다 사용 가능한 공유메모리 용량이 다르기 때문에 사용 가능한 스레드도 GPU 아키텍처마다 다르다. 이러한 점을 고려하여 제안기법에서는 공유 메모리를 동적 할당하여 구현하였다.

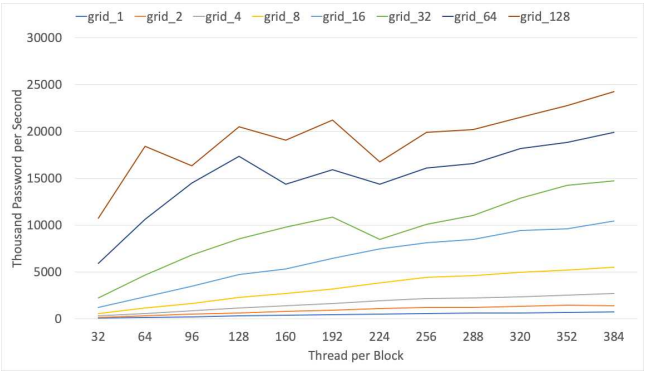
3.3 블록, 스레드 수 조정

블록 당 스레드 수와 그리드당 블록 수는 성능에 영향을 미치기 때문에 최적 성능을 달성하기 위해서는 적절한 선택 필요하다. 먼저 이를 확인하기 위해 블록 당 스레드 수와 그리드당 블록 수에 따른 성능을 변화를 확인 하였다. RTX 3060 환경에서 제안기법을 적용한 PDF 암호해독 구현을 동작하여 변화에 초당 패스워드 연산 횟수를 측정하였다. 그림 2은 블록 수에 따른 초당 패스워드 연산 횟수 결과이다.

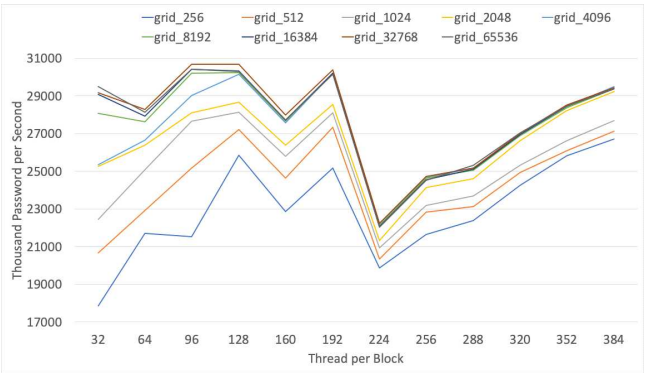


(그림 2) RTX 3060 환경에서 PDF 크래킹 시, 그리드당 블록 수에 따른 초당 패스워드 연산 횟수(kp/s)

블록수가 증가할수록 블록 수에 따라 성능이 높아지다 특정 지점에 수렴하였다. 스레드 수에 따른 초당 패스워드 연산 횟수의 경우 블록의 수가 적을 때는 그림 3와 같이 일정하게 증가하였으나 블록의 수가 많아지면 그림 4와 같이 불규칙 적으로 변화하였다.



(그림 3) RTX 3060 환경에서 PDF 크래킹 시, 각각의 128개 이하의 그리드당 블록 수에서 블록당 스레드 수에 따른 초당 패스워드 연산 횟수(kp/s)



(그림 4) RTX 3060 환경에서 PDF 크래킹 시, 각각의 128개 이상의 그리드당 블록 수에서 블록당 스레드 수에 따른 초당 패스워드 연산 횟수(kp/s)

이러한 불규칙성을 고려하여 최적의 파라메터 선택을 위해 [4]의 Autotune을 사용하였다. 성능에 영향을

미치는 요소로는 레지스터 사용, 스레드 블록 크기, 루프 등과 같은 다양한 요인이 있다. [4]는 장치와 알고리즘에 따라 최적의 파라미터 값을 자동으로 탐색하는 Autotune를 제안하였다. [4]에서는 레지스터 사용, 스레드 블록 크기, 루프 unroll과 같은 요인 탐색한다. Hashcat 또한 v3부터 Autotune 기능 추가하였다. 제안 기법에서는 스레드 수 및 그리드 수 두 가지 요소에 대하여 탐색하였다. 본 논문의 구현 기법은 RTX 3060 환경에서 그리드 당 블록 수는 65536개, 블록 당 스레드 수는 96개에서 가장 좋은 성능을 보였다.

4. 성능비교

RTX 3060환경에서 측정하였으며 Hashcat-6.2.5과 비교하였다. Hashcat은 제공되는 밴치마크 모드(hashcat -b -m10500 -w 4 -O)로 동작한 초당 계산 횟수(를 측정하였다. 제안기법 구현은 Hashcat과 동일하게 GPU로 데이터를 전송 이전 부터 커널 완료까지 시간측정하여 초당 계산 횟수를 측정하였다. 결과적으로 표 1과 같이 제안기법이 31460kp/s, hashcat에서 25693kp/s로 22.5%의 향상된 성능을 보였다.

< 표 1> RTX 3060환경에서 Hashca과 제안 기법의 초당 계산 횟수 비교

	RTX 3060
hashcat 6.2.5	25693 kp/s
제안기법	31460 kp/s
개선율	22.5%

5. 결론

본 논문은 PDF 1.4 -1.6 버전 암호 해독의 알고리즘을 CUDA GPU 상에서 최적화하였다. MD5와 RC4의 최적구현기법 적용과 Autotune을 사용하여 최적의 성능에 가까운 블록당 스레드 수, 그리드당 블록 수의 값을 사용하여 높은 성능 향상을 보였다. 결과적으로 RTX 3060 환경에서 크래킹 도구 해시캣과 비교하여 22.5%의 성능 향상을 달성하였다. 추후 연구에는 더 다양한 환경에서의 성능 비교와 다른 PDF 버전의 암호 해독 알고리즘의 최적화를 구현 하고자 한다.

6. Acknowledgement

이 논문은 2022년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2021-0-00540, GPU/ASIC 기반 암호알고리즘 고속화 설계 및 구현 기술개발, 100%).

참고문헌

[1] The application/pdf Media Type“, 2017, RFC 8118

[2] B. den. Boer, A. Bosselaers. Collisions for the compression function of MD5, Advances in Cryptology, Eurocrypt'93 Proceedings, Springer-Verlag, 1994.

[3] Knudsen, Meier, Preneel, Rijmen, and Verdoolaege. Analysis methods for (alleged) RC4. In ASIACRYPT: Advances in Cryptology-ASIACRYPT: International Conference on the Theory and Application of Cryptology. LNCS, Springer-Verlag, 1998.

[4] Khalid, A., Paul, G., & Chattopadhyay, A. “New speed records for Salsa20 stream cipher using an autotuning framework on GPUs”. In International Conference on Cryptology in Africa (pp. 189-207). Springer, Berlin, Heidelberg. 2013.