GPU를 활용한 Argon2 고속화 구현

엄시우*, 김현준*, 송민호*, 서화정**

*한성대학교 (대학원생)

**한성대학교 (교수)

Fast Implementation of Argon2 using GPU

Si-Woo Eum*, Hyun-Jun Kim*, Min-Ho Song*, Hwa-Jeong Seo**

*Hansung University(Graduate student)

**Hansung University(Professor)

요 약

본 논문은 GPU를 이용한 Argon2 해시 함수의 고속화 구현을 진행하였다. Argon2는 비밀번호 저장 및 전송에 사용되는 안전한 해시 함수로, 본 논문은 GPU를 활용하여 Argon2의 성능을 향상시키는 방법을 제시합니다. 기존 Argon2의 연산을 진행하기 위해 첫 블록의 연산을 CPU와 GPU로 나누어 연산을 구현함으로써 CPU에서 첫 블록의 연산을 모두 연산하였을 때보다 메모리 관점에서 이점이 있었으며, 결과적으로 메모리 복사 시간이 1.7배 빨라진 것을 확인하였다. 또한 해당 기법이 메모리 관점에서의 이점을 가져오면서 Argon2의 크래킹 구현에서도 이점으로 작용함으로써 효율적인 크래킹 구현이 가능함을 설명한다.

I. 서론

현대의 정보 시스템에서는 개인 정보와 보안 데이터의 안전한 저장과 전송이 핵심 과제로 인식되고 있다. 이를 위해 안전하고 강력한 암호화 기술이 필요하다. Argon2는 비교적 최근에 개발된 비밀번호 해시 함수로, PHC (Password Hashing Competition)에서 우승한 알고리즘이다. Argon2[1]는 GPU를 활용한 크래킹 공격으로부터 안전한 기법을 지원한다.

GPU는 병렬 처리 능력이 뛰어나고 대용량 데이터에 대한 연산을 가속화할 수 있는 그래픽 처리 장치입니다. 이 특징을 활용하여 Argon2의 고속화 구현이 가능하다면 Argon2 해시 함수의 성능을 향상시키는 것뿐만 아니라 크래킹 공격에 대한 분석이 가능하다.

본 논문은 GPU를 활용한 Argon2 해시 함수의 고속화 구현 방법에 대해 다루며, 이를 통해 속도와 메모리 관점에서의 성능 향상을 목표로 한다. GPU를 사용하여 Argon2의 병렬 처리를 최적화하고, 속도와 메모리 측면에서 기존 구현과 비교하여 결과를 분석한다.

II. 관련 논문

2.1 Graphic Processing Uint

그래픽 처리 장치(GPU)는 전용 하드웨어로, 주로 그래픽 및 영상 처리 작업에 특화된 장치이다. CPU와는 구조와 목적이 다르며, 대량의 데이터를 동시에 처리가 가능하다. 일반적으로 CPU는 다양한 작업을 처리하는 범용 프로세서로 사용하지만 GPU는 대량의 데이터를 병렬로 처리하는 데에 특화되어 있다.

최근에는 GPU가 일반 컴퓨팅 작업에도 널리 활용되고 있습니다. GPU는 수천 개 이상의 코어를 가지고 있어 동시에 많은 연산을 처리할 수 있으며, 병렬 처리 능력을 활용하여 과학, 엔지니어링, 딥 러닝, 데이터 마이닝 등의 계산 집약적인 작업에서 성능을 향상시킬 수

있습니다. CUDA나 OpenCL과 같은 프로그래밍 모델을 사용하여 GPU를 프로그래밍할 수 있다[2,3].

2.2 GPU를 활용한 크래킹

GPU는 병렬 연산 능력이 뛰어나기 때문에 패스워드 크래킹과 같은 암호 해독 작업에 매우 유용하게 사용된다. 크래킹은 암호화된 데이터를 해독하여 암호화에 사용된 패스워드를 찾는 과정이다.

패스워드는 해시 함수를 통해 암호화되어 저장되는 경우가 많다. 일반적으로 해시 함수는 일방향 함수로, 암호화된 값을 다시 원래 값으로 역추적하기 어렵게 만든다. GPU는 많은 수의 코어를 가지고 있으며, 이를 활용하여 동시에 다수의 해시 값을 계산할 수 있다.. 이를 통해 더 빠른 속도로 크래킹 작업을 수행 가능하다[4].

2.3 비밀번호 해시 함수

비밀번호 해시 함수는 일반적인 해시 함수와 다른 특징을 가지고 있다. 일반적으로 해시 함수는 임의의 데이터를 받아 고정된 길이의 해시 값으로 변환하는 알고리즘이다. 이러한 해시 함수는 데이터의 무결성을 확인하거나 데이터의 일치 여부를 빠르게 확인하는 데 사용된다.

비밀번호 해시 함수는 일반적인 해시 함수와 다르게 비밀번호와 같은 암호화된 데이터를 해시 값으로 변환하는 알고리즘이다. 주요 차이점은 추가적인 보안 기능을 제공하는 것이다. 추가적인 보안 기능을 제공하기 위해서 입력 값에 솔트 값이 추가되거나 연산의 반복 횟수를 조절하는 등의 기법이 적용된다.

대표적인 비밀번호 해시 함수로는 PBKDF2, bcrypt, scrypt 등이 있으며 본 논문에서 다루는 Argon2도 비밀번호 해시 함수이다[5,6].

2.4 Argon2

Password Hashing Competition(PHC)는 비밀번호 해시 함수의 안전성과 보안성을 향상시키기 위해서 2013년에 개최된 대회이다. 2015년에 최종 선정 알고리즘이 결정되었으며, 여기서 최종 선정된 알고리즘이 Argon2 알고리즘이다[1].

Argon2는 비밀번호를 안전하게 저장하기 위해 사용되는데, 일반적으로 비밀번호를 평문으로 저장하는 것은 취약하며, 해시 함수를 사용하여 비밀번호를 암호화하고 저장한다. 여기서 비밀 번호를 암호화할 때 사용되는 알고리즘이다.

Argon2은 부채널 공격과 GPU를 활용한 크래킹 공격으로부터 내성이 강한 특징이 있다. 이를 위해서 Argon2d, Argon2i, Argon2id 총 세 가지 버전을 지원하고 있다. Argon2는 메모리 접근 패턴을 통해서 각 공격으로부터 안전한 알고리즘을 설계하였다. Argon2i의 경우 데이터 독립성 메모리 접근을 통해 부채널 공격으로부터 안전하게 설계되었다. 데이터 독립성은 입력된 데이터와 상관없이 실행 결로나 메모리 접근 패턴이 동일한 것을 의미한다. Argon2d는 데이터 종속성 메모리 접근을 통해 GPU 크래킹 공격으로부터 안전하게 설계되었다. 데이터 종속성은 입력 데이터에 따라서 실행경로, 메모리 접근 패턴이 달라지기 때문에 알고리즘의 동작을 예측하기 어렵고, 이로 인해 크래킹 공격으로부터 강한 내성을 갖는다. 마지막으로 Argon2id는 두 구현의 장점을 혼합하여 구현된 하이브리드 방법이다. Argon2는 이외에도 솔트, 스트레칭 파라미터 기법 등을 통해서 여러 공격으로부터 안전한 알고리즘이다.

Argon2의 알고리즘은 (그림 1)과 같다.

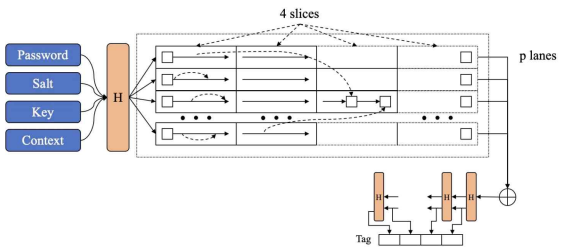

Fig. 1. Algorithm of Argon2

III. 구현 기법

본 논문에서는 GPU를 활용하여 Argon2 해시 함수의 고속화 구현을 진행한다. 기존 GPU 구현[7]을 참고하여 구현을 진행하였다.

[7]구현의 경우 Argon2 내부에서 활용하는 Blake2B 해시 함수(이하 H)가 CPU에서 연산되고 연산된 결과값을 GPU로 복사한 후에 Argon2의 핵심 연산이 GPU에서 동작하도록 구현되어 있다. 본 논문에서 CPU에서 연산되는 H 연산 일부를 GPU에서 연산되도록 구현하였다. 본 장에서는 H 연산을 GPU에서 연산하도록 함으로써 오는 장점에 대해서 메모리와 크래킹 관점에서 설명한다.

3.1 메모리 관점

GPU를 활용한 구현은 보통 대용량의 데이터를 병렬 처리하기 위해서 Host(CPU)에서 Device(GPU)로 대용량의 메모리를 복사하여 연산을 진행한다. 즉, GPU를 활용한 구현에서는 메모리 복사 비용이 성능에 영향을 미친다. 기존[]구현에서도 메모리 복사의 비용을 줄이기 위해서 Stream 방식을 활용하여 메모리 복사를 구현하였다. 또한 Cuda에서도 메모리 복사를 좀 더 효율적으로 할 수 있도록 여러 기능을 지원하고 있다.

먼저 H 연산을 Device 즉, GPU에서 연산함으로써 CPU에서 GPU로 값을 복사하는 비용이 줄어든다. 기존에는 CPU에서 여러 입력값을 토해서 blake2B 해시값을 계산하고 계산된 해시값을 Argon2 코어 연산을 위해서 여러 lane에 맞춰서 복사해주어야 한다. 이로 인한 값의 크기는 8,192-Byte의 크기를 갖는다. 여기에서 만약 Batchsize가 커지게 되면 복사하는 비용은 더 크게 증가하게 된다. 즉 기존 구현의 경우 연산을 위해 복사해야 하는 데이터의 크기는 8,192 * Batchsize Byte이다.

하지만 (그림 2)와 같이 첫 블록 연산의 일부를 GPU에서 동작하게 되면 복하해야 하는 메모리 비용이 감소한다. 결론적으로 64-byte로 감소하게 된다. 마찬가지로 연산된 해시 값을 통해서 다음 연산이 진행될 수 있도록 해시 값이 복사되어야 하는데, 이 과정이 GPU에서 연산하게 되면 GPU 내부에서 값을 복사하는 것이기 때문에 CPU에서 GPU로 복사하는 비용보다 작다.

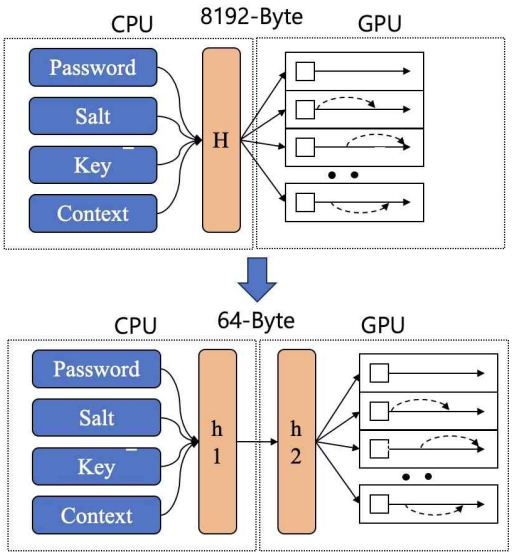

Fig. 2. Decomposition of H-functions for better performance from a memory perspective

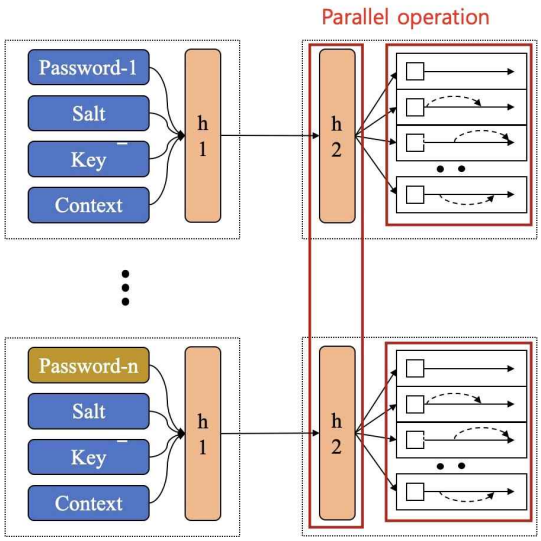

Fig. 3. Working structure from a cracking point of view

3.2 크래킹 관점

Craking을 목적으로 하는 관점에서 보게 된다면 메모리 관점에서의 장점이 그대로 크래킹에서도 장점으로 작용한다. 크래킹을 하기 위한 특성상 무수히 많은 입력 값의 해시 값을 계산해야 한다. 크래킹에서는 입력값의 차이만 존재하기 때문에 GPU를 활용하여 병렬 연산하기에 좋은 특징을 가지고 있다. 하지만 H 연산이 CPU에서

동작하고 큰 메모리의 값을 복사해야 한다면 무수히 많은 입력 값을 계산하기에는 비효율적이다. 하지만 본 논문에서 제안하는 H 함수의 분할은 무수히 많은 입력 값 계산을 병렬로 처리하는 것이 가능하다.

(그림 3)과 같이 GPU에 올라온 h1 함수의 연산 결과 값들은 모두 같은 연산을 진행하기 때문에 GPU에서 병렬 구현이 가능하다.

하지만 이때 고려해야 하는 점은 Argon2을 GPU에서 구현하기 위해서는 많은 메모리와 Thread를 사용한다. 많은 패스워드를 한 번에 병렬 처리하기 위해서는 더 많은 메모리와 Thread를 요구하기 때문에 파라미터 조절을 통해서 Thread의 적절한 배분과 메모리 조절을 통해 구현되어야 한다.

IV. 성능 평가

성능 평가를 위해 Nivida GTX 3060 Laptop 상에서 구현 및 성능 측정을 진행하였다. Visual Studio상에서 구현을 진행하였으며, Cuda 11.7 runtime 버전을 사용하였다. 프로젝트 빌드 시 Release 방식으로 빌드하여 동작한 성능을 측정하였다.

성능 측정은 10번 동작하였을 때의 평균 시간을 측정하였으며, CPU에서 GPU로 메모리 복사하는 시간, GPU에서의 연산 시간, GPU에서 CPU로 메모리를 복사하는 시간을 계산하였다.

	[7]	Ours
CPU->GPU	9.6ms	5.4ms
Computeation	56.3ms	75ms
GPU->CPU	0.75ms	1.01ms

Table 1. Performance result

측정 결과 CPU에서 GPU로 메모리를 복사하는 시간이 4ms 감소한 것을 확인할 수 있다. 하지만 H함수 연산이 GPU에 포함됨으로써 GPU에서 연산하는 시간이 20ms 증가하였다.

결과적으로 본 논문에서 제안하는 기법을 통해서 메모리 복사 비용이 1.7배 정도의 빨라졌지만, 연산 비용은 0.7배 정도 느려졌다.

V. 결론

본 논문에서는 GPU를 활용한 Argon2 구현을 좀 더 고속화 구현을 하기 위해 H 연산을 GPU에서 연산하는 것을 제안하였다. H 연산이 GPU에서 동작함으로써 CPU에서 GPU로 데이터를 복사하는 비용을 줄이는 것이 가능하며, 크래킹 구현에 효율적인 구현이 가능해진다. 추후에는 고속화 구현을 활용한 Argon2의 크래킹 최적화 구현을 제안한다.

VI. Acknowledgement

This work was 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, 100%).

[참고문헌]

[1] Biryukov, Alex, Daniel Dinu, and Dmitry Khovratovich. "Argon2: new generation of memory-hard functions for password hashing and other applications." 2016 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2016.

[2] An, SangWoo, et al. "Parallel implementations of ARX-based block ciphers on graphic processing units." Mathematics 8.11 (2020): 1894.

[3] Owens, John D., et al. "GPU computing." Proceedings of the IEEE 96.5 (2008): 879-899.

[4] Sprengers, Martijn. "GPU-based password cracking." On the Security of Password Hashing Schemes regarding Advances in Graphics Processing Units,

Radboud University Nijmegen (2011).

[5]Provos, Niels, and David Mazieres. "Bcrypt algorithm." USENIX. 1999.

[6] Ertaul, Levent, Manpreet Kaur, and Venkata Arun Kumar R. Gudise. "Implementation and performance analysis of pbkdf2, bcrypt, scrypt algorithms." Proceedings of the International Conference on Wireless Networks (ICWN). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2016.

[7] Alexandru Ionut Budisteanu, “GPU is unfriendly for WebDollar - argon2-gpu for WebDollar,” [Github Source Code], https://github.com/WebDollar/argon2-gpu