情 報 保 護 學 會 誌
第20卷 第6號, 2010. 12

GPU에서의 SEED암호 알고리즘 수행을 통한 공인인증서 패스워드 공격 위협과 대응

김 종 회*, 안 지 민**, 김 민 재*, 주 용 식***

요 약

병렬처리를 이용한 GPU(그래픽 프로세싱 유닛)의 연산 능력이 날이 갈수록 고속화됨에 따라 GPU에 대한 관심이 높아지고 있다. GPU는 다중 쓰레드 처리가 가능하도록 CPU보다 수십 배 많은 멀티코어로 구성되어 있으며 이 각각의 코어는 병렬 프로그래밍이 가능하도록 처리 결과를 공유할 수 있다. 최근 해외에서 이러한 GPU의 연산 능력을 이용한 해쉬 인증 공격의 효과가 다수 입증되었으며 패스워드 기반의 인증 방식이 보편화 되어있는 국내에서도 GPU를 이용한 인증 공격이 시도되고 있다. 본 논문에서는 국내 금융권에서 사용되고 있는 공인인증서의 개인키 복호화 과정을 GPU내에서 고속 수행이 가능하도록 개선하고, 이를 바탕으로 패스워드 무차별 대입 공격을 시도하여 공인 인증서에 사용되는 패스워드가 보안의 안전지대만이 아님을 보인다. 또한 날로 발전하는 하드웨어의 연산속도에 맞추어 공인인증서 등에 보편적으로 사용되는 패스워드 정책의 개선 방안을 제시한다.

Ⅰ. 서 론

1992년 2월 국내에서 공개키 암호화 기술에 기반한 전자서명에 대한 법적인 효력을 부여하는 전자서명법이 제정됨에 따라 이를 근간으로 국내 전자서명 인증체계가 구축되었다. 공인인증서를 이용한 전자서명 인증 체계는 전자상거래를 포함한 인터넷 뱅킹, 사이버 증권거래, 민원발급에 이르기까지 다양한 분야에서 사용자 식별, 메시지 무결성 보장, 부인방지와 같은 기능을 제공하며 널리 사용되고 있다.

공개키 기반(PKI)의 전자서명시스템에서 사용되는 공인인증서와 개인키 저장 파일은 X.509 v3를 기준으로 작성되며, 특히 개인키 파일은 KISA(한국 인터넷 진흥원)에서 개발한 SEED 블록 암호화 알고리즘에 의해 사용자가 입력한 패스워드로 보호된다[16].

또한 안전한 사용자 식별을 위해 OTP (One - Time - Password) 및 HSM (Hardware Security Module)과 같은 부가적인 개인키 보호 수단이 도입 되고 있다. 그러나 현실적으로 대부분의 공인인증서 사용자는 패스워드만으로 보호된 인증서를 사용하고 있으며 이에 따라 사용자의 개인키 암호화 패스워드 노출 시 개인정보 유출 및 신분위장 공격 위험이 따르게 된다. 현재 국내 공인인증서 패스워드 정책은 보편적으로 영문자와 숫자가 혼합된 최소 8자리 이상의 패스워드를 기준으로 시행되고 있다.

하지만 이러한 패스워드 정책 기준은 현재의 공격용 하드웨어 성능 발달이 고려되지 않은 채 몇 년간 동일

〔표 1〕 LockDown Password Recovery Speed 요약

패스워드		사양	
길이	경우의 수	워크스테이션	슈퍼컴퓨터
2	3,844	Instant	Instant
3	238,328	Instant	Instant
4	15 Million	Instant	Instant
5	916 Million	9 Secs	Instant
6	57 Billion	9½ Mins	56 Secs
7	3.5 Trillion	10 Hours	58 Mins
8	218 Trillion	25¼ Days	60½ Hours

* 단국대학교 정보보안연구실 (gonan@dankook.ac.kr, iscooking@hanmail.net)
** 제주대학교 (cumico0312@nate.com)
*** 성균관대학교 정보통신대학원(yongsigi00@nate.com)

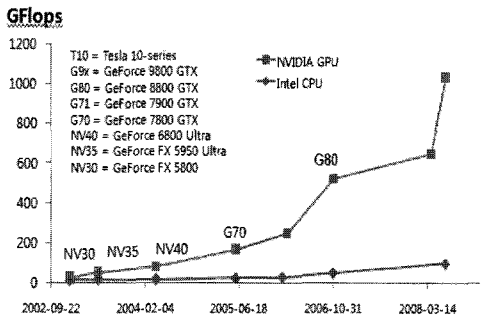
하게 유지되고 있다. [표 1]은 2007년 1월 영국의 웹사이트 'LockDown'은 'Password Recovery Speed'라는 보고서의 일부로 영문자와 숫자를 조합한 8자리 패스워드의 경우 워크스테이션은 약 25일, 슈퍼 컴퓨터의 경우는 약 60시간이 소용되는 것으로 패스워드 공격 전수조사 결과를 나타낸다. 또한 영어대소문자와 숫자를 포함한 총 62개의 문자열 조합에 대한 패스워드 공격이 듀얼 코어 PC 기준 260일이 넘지 않음을 보인다. 이는 공인인증서의 경우 인증서 유효기간이 1년인 점을 감안할 때 매우 위협적인 내용이 아닐 수 없다.

현재 하드웨어의 성능은 비약적으로 발달 하고 있으며, 특히 연산처리 과정에 있어서의 GPU를 이용한 병렬처리 속도는 CPU를 능가하여 매년 약 2배 정도 증가하고 있다. CPU의 발전 방향이 연산속도의 증대에서 멀티코어 탑재로 전환되고 있어 CPU의 속도 증대는 다소 주춤한 상황이다. 그러나 그래픽 관련 부동 소수점 연산을 주로 하는 GPU는 연산능력에 있어서 CPU의 능력을 압도한지 오래이다. 또한 NVIDIA社의 경우 그래픽 출력 인터페이스가 포함되지 않은 연산처리 전용 GPU인 Tesla를 내놓으면서 개인용 분산 컴퓨팅 환경 시대를 열었다.

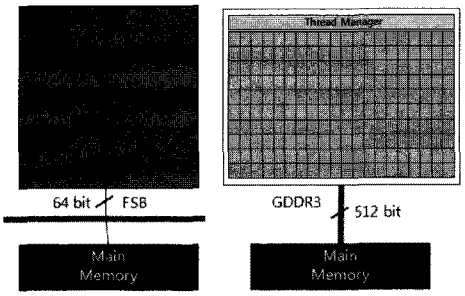
[그림 1]은 CPU와 GPU의 성능 발전 추이를 비교한 그래프로, 2004년 이후 성능 발전 격차가 벌어지기 시작하여 현재에는 매우 큰 차이를 보이고 있다.

GPU는 CPU와는 달리 수 백개에 달하는 코어를 내장하고 있어 다수의 연산 작업에 대한 병렬처리에 효과적이며 이러한 결과는 이미 수치적으로 입증되었다.

또한 병렬처리 과정에서 발생할 수 있는 I/O 병목현상은 PCI Express 지원으로 해소되었고 NVIDIA社에서 제공하는 CUDA와 같은 GPU 프로그래밍 라이브러



[그림 1] CPU와 GPU의 성능 발전 추이[8]



[그림 2] CPU와 GPU의 구조적 차이

리는 일반 사용자에게 GPU를 활용한 슈퍼컴퓨팅 환경을 제공하게 되었다. [그림 2]는 CPU와 GPU의 구조적 차이를 보이는 것으로, 수 백개에 이르는 코어와 큰 대역폭의 버스를 통하여 CPU에 비해 GPU가 더 빠른 연산이 가능함을 나타낸다.

최근 해외에서는 이러한 GPU의 고속 연산처리 능력을 이용하여 해쉬 알고리즘 공격도구와 장비가 소개되고 있으며 이러한 시도는 공인인증서 개인키 암호화에 사용되는 패스워드 정책 역시 언제든 공격대상이 될 수 있음을 의미한다.

Ⅱ. 공인 인증서 개인키 암호화 과정

국내 금융권에서 사용되고 있는 공인인증서의 경우 BER/DER 방식으로 인코딩 되어 있으며[6], 이 방식은 인증서 폴더 내에 der확장자를 가지는 인증서 파일과 key확장자를 가지는 개인키 파일을 가지게 된다. 개인키 저장파일은 PKCS #5(Public Key Cryptography Standards #5)에서 정의한 PEBS1(Password Based Encryption Scheme, 패스워드 기반의 키 암호화[14]) 기준을 따르며 사용자가 입력한 패스워드를 비밀 키로 하여 KISA에서 개발한 SEED블록 암호 알고리즘으로 암호화 한다[16].

암호화 된 개인키 파일은 ASN.1으로 인코딩 되어있으며 이 파일에 저장되어 있는 OID(객체 식별자)는 개인키 파일을 암호화하기 위한 서명 알고리즘을 나타낸다. 국내에서는 '1.2.410.2000041.15'의 OID를 가지는 seedCBCwithSHA1 서명 알고리즘이 주로 사용되는데 이는 SHA-1 해쉬로 유도된 초기화 벡터와 키를 이용하여 SEED블록 암호 알고리즘으로 암호화 한다는 것을

의미하고 KISA의 암호 알고리즘 규격(KISA.TS.ENC)에서 다음과 같이 정의 하고 있다.

EncryptedPrivateKeyInfo ::= SEQUENCE{
encryptedAlgorithm
EncryptedAlgorithmIdentifer
encryptedDate EncryptedData
}

EncryptedAlgorithmIdentifer는 PBKDF1 함수에 암호화 알고리즘의 객체 식별자(OID)와 함께 8바이트 크기의 솔트(S) 값과 반복횟수(C)를 입력 값으로 하여 20바이트의 추출키(DK)를 생성한다[2].

$$DK = PBKDF1(P, S, c, 20)$$

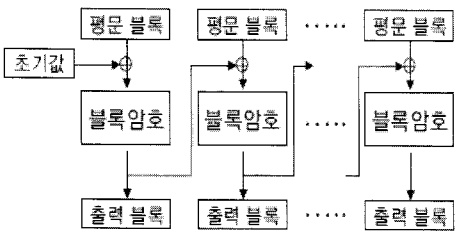
이 과정에서 추출된 DK의 처음 16바이트는 암호화 키로 정의된다[2].

$$K = DK<0 .. 15>$$

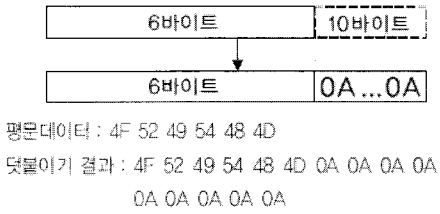
초기화 벡터(IV)의 생성은 DK에서 키(K)를 제외한 나머지 4바이트를 SHA-1으로 해쉬화 하여 20바이트 값 (DIV)를 생성하고, 처음 16바이트를 초기화 벡터로 정의하고 있다[2].

$$DIV = Hash(DK<16 .. 19>)$$
$$IV = DIV <0 .. 15>$$

이렇게 유도된 비밀키와 초기화 벡터는 SEED 암호 알고리즘에서 개인키 메시지 블록을 암호화 하기 위해



〔그림 3〕 SEED 알고리즘의 CBC 운영 모드[2]



〔그림 4〕 seedCBCwithSHA1의 패딩처리 방식[2]

사용된다. SEED블록 암호화 알고리즘 과정에서 CBC 운영 모드는 평문데이터의 128bit 블록으로 구분하여 이전 128bit 출력(암호문) 블록 값과 128bit 평문 블록 값의 베타로직 논리합을 수행한 후, 블록암호의 입력 값으로 처리하는 운영모드를 의미하는데, 이는 [그림 3]과 같다.

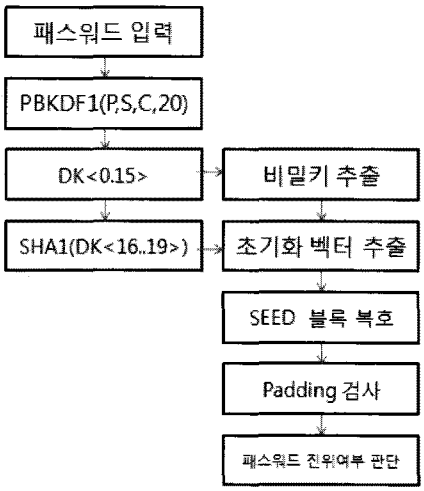
또한 운영모드를 이용하여 입력 값을 처리하기 위해서는 평문데이터의 길이 128bit의 양의 정수배가 되도록 패딩처리가 이루어져야 한다. SeedCBCwithSHA1에서의 패딩처리 방식은 평문 데이터를 바이트 단위로 처리하며, 평문데이터의 크기가 16바이트의 정수배가 아닐 경우, 16의 배수가 되기 위해 필요한 바이트의 개수를 한 바이트로 표현하여 필요한 바이트 수만큼 최하위 바이트에 패딩 처리한다. [그림 4]에서 와 같이 16의 배수가 되기위해 10개의 바이트 패딩이 필요하며, 패딩처리에는 10에 대한 Hex값인 '0A'를 10번 반복하여 덧

```
02 dd d2 53 01 1f d5 22 98 20 68 99 9a 55 54 81
26 d2 3e aa 7f d4 57 38 2b b6 7a e0 ab 20 bc 6a
9f 5a a8 b7 5f 45 5f 02 41 00 82 29 50 35 8f 16
76 8b ae ca 8a 08 23 8c b1 69 a6 f2 e7 16 54 33
b0 aa bf d3 9e 54 82 6e 97 b3 87 b5 60 64 bf 5c
01 af 81 0e 82 cb 11 fc 81 f4 ab 3c 3b fd 50 ac
fa 42 2b cc 10 f2 91 2a 27 9f a0 27 30 25 06 0a
2a 83 1a 8c 9a 44 0a 01 01 03 31 17 03 15 00 81
c0 2d 52 22 cf 71 e8 d3 a3 f5 58 b2 01 41 6d fd
31 79 18
```

〔그림 5〕 패딩 처리가 되지 않은 복호화 메시지 예

```
02 dd d2 53 01 1f d5 22 98 20 68 99 9a 55 54 81
26 d2 3e aa 7f d4 57 38 2b b6 7a e0 ab 20 bc 6a
9f 5a a8 b7 5f 45 5f 02 41 00 82 29 50 35 8f 16
76 8b ae ca 8a 08 23 8c b1 69 a6 f2 e7 16 54 33
b0 aa bf d3 9e 54 82 6e 97 b3 87 b5 60 64 bf 5c
01 af 81 0e 82 cb 11 fc 81 f4 ab 3c 3b fd 50 ac
fa 42 2b cc 10 f2 91 2a 27 9f a0 27 30 25 06 0a
2a 83 1a 8c 9a 44 0a 01 01 03 31 17 03 15 00 81
c0 2d 52 22 cf 71 e8 d3 a3 f5 58 b2 01 41 6d fd
31 79 18 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d
```

〔그림 6〕 패딩이 포함된 복호화 메시지 예



〔그림 7〕 개인키 패스워드 정당성 판별

붙이게 된다[2].

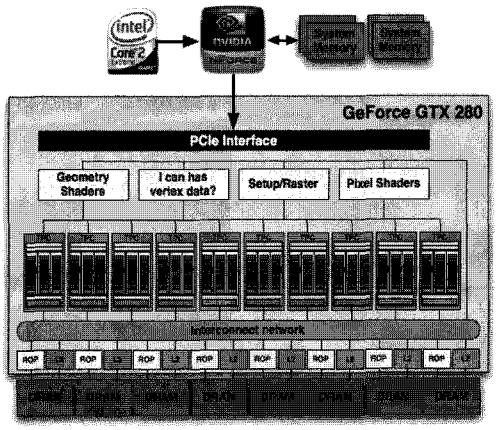
이렇게 암호화된 개인키 파일은 공인인증서 사용 시 사용자가 입력한 패스워드를 이용하여 복호화 된다. 이 때 패스워드의 일치여부는 패스워드를 사용하여 복호화 한 메시지에 PEBS에서 정의한 올바른 패딩이 존재하는지 여부를 통해 확인할 수 있게 된다[16]. [그림 5]와 [그림 6]은 각각 패딩처리가 되지 않은 복호화 메시지와 패딩처리가 된 복호화 메시지를 나타낸다. [그림 6]의 가장 마지막 부분에 0d 0d 0d 와 같은 패딩 비트가 추가 되어 있음을 알 수 있다.

[그림 7]은 위에서 설명된 SEED 블록 암호화 방법과 패딩 처리 방식에 의해 사용자가 입력한 패스워드를 이용하여 암호화된 개인키를 복호화 하는 과정을 전반적으로 나타내었다. 처음 사용자의 패스워드가 입력되면 PBKDF1함수에 패스워드와 8바이트의 솔트 값, 그리고 반복 횟수를 파라미터로 하여 출력된 DK의 처음 16바이트를 비밀 키로 사용하고, 하위 4바이트를 SHA1 해쉬를 이용하여 IV로 사용한다. 이렇게 유도된 비밀 키와 초기화 벡터를 SEED 블록 암호화 알고리즘을 이용하여 메시지를 복호화 한다. 복호화 된 메시지의 마지막 블록에 존재하는 패딩을 정당성을 조사함으로써 입력된 패스워드의 진위여부를 판별 할 수 있게 된다.

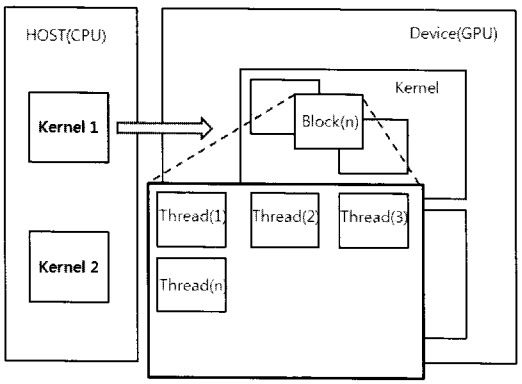
Ⅲ. GPU를 이용한 개인키 복호화 수행

2장에서 설명한 방법을 통하여 Intel Core2Duo 2.1GHz의 CPU와 4GB의 RAM 의 PC사양에서 10개의 병렬 쓰레드로 금융권 공인인증서 개인키 샘플에 대해 10,000번의 복호화를 수행한 결과 회당 평균 3.1020 ms이 소요되는 결과를 얻을 수 있었다. 이와 같은 결과는 현재 가장 많이 사용되고 있는 영소문자와 숫자가 혼합된 8자리 패스워드를 공격하기 위해 약 1088일 소요됨을 의미한다. 따라서 복호화 과정의 수행속도 개선을 위해 앞서 소개한 NVIDIA社의 CUDA 라이브러리를 이용하여 패스워드 생성 알고리즘과 SEED블록 암호 처리과정을 멀티 코어에서 병렬처리 될 수 있도록 구현 과정을 설명한다.

NVIDIA社의 GPU는 G80계열의 GPU부터 CUDA를 지원하게 되어 있으며, 그 구조는 [그림 8]과 같다. 현재의 NVIDIA GPU는 240개의 코어를 가지고 있으며, 8개의 코어가 묶여 30개의 멀티프로세서를 가진다. 특히 NVIDA의 Tesla시스템은 그래픽 출력이 없는 연산전용 하드웨어로서 4GB의 메모리를 장착하고 있다[13]. GPU에서 수행되는 프로그램을 작성하기 위해서는 NVIDIA에서 제공하는 CUDA라이브러리를 이용해야 한다. CUDA를 이용하여 프로그램을 작성하게 되면 최초 CPU에서 수행되는 호스트 프로그램이 시스템에 존재하는 GPU칩셋을 찾아 GPU 서브 프로그램을 로드하게 된다. 이때 이 GPU 서브 프로그램을 그리드라 부른다[13]. 이 그리드 블록은 여러 개의 쓰레드 블록들로 구성되어 있으며 각각의 쓰레드 블록은 GPU내에 존재하는 수 십개의 멀티프로세서에 할당되어 독립적으로 수행되게 된다[8].



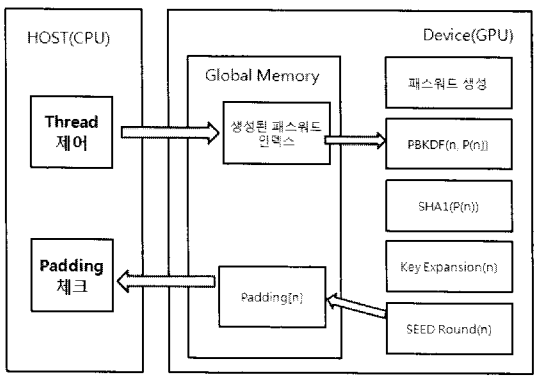
〔그림 8〕 NVIDIA社의 Geforce GTX 280 GPU의 구조[8]



〔그림 9〕 GPU에서의 쓰레드 실행

[그림 9]는 그 수행 구조를 나타낸다. GPU내에는 각 쓰레드들의 수행결과를 공유 할 수 있는 Global 메모리가 존재하며 CPU에 존재하는 호스트 프로그램은 이 Global Memory에 직접 접근하여 GPU내 각 쓰레드의 수행을 제어할 수 있게 된다. CUDA를 이용하여 GPU내에 존재하는 각각의 코어에서 쓰레드를 병렬처리 하기 위해서는 메모리의 계층구조를 통하여 각 쓰레드의 수행결과를 공유함으로써 쓰레드의 수행내용을 수시로 동기화해야 한다[13].

CUDA라이브러리를 이용하여 GPU내에서 패스워드 생성 및 SEED 복호화 함수를 수행하기 위해서는 GPU의 멀티 코어 쓰레딩 환경을 고려한 알고리즘이 적용되어야 한다[13]. 따라서 [표 2]와 같이 불필요한 I/O작업을 최소화하고 SEED블록에 대한 복호화 작업을 분리함으로써 각 쓰레드 작업의 효율을 최대화 할 수 있다. 아래에서 CPU와 GPU내에서 각각 수행되어야할 쓰레드의 역할을 분리하였다.



〔그림 10〕 쓰레드 단위의 수행 관계

〔표 2〕 쓰레드 수행모드 분리

CPU	GPU
· 커널 할당 · 쓰레드 제어 · 패딩 체크	· 패스워드 생성 및 Indexing · 비밀키 유도 · IV 유도 · KEY Expansion · SEED 라운드 수행

이와 같이 분리된 각 작업의 수행은 100개 단위의 쓰레드로 동시에 수행 될 수 있도록 쓰레드 인덱스 번호를 파라미터로 이용하여 각 쓰레드의 절차적 작업 순서를 따른다. [그림 10]은 각 쓰레드의 수행 절차에 관해 나타내었다.

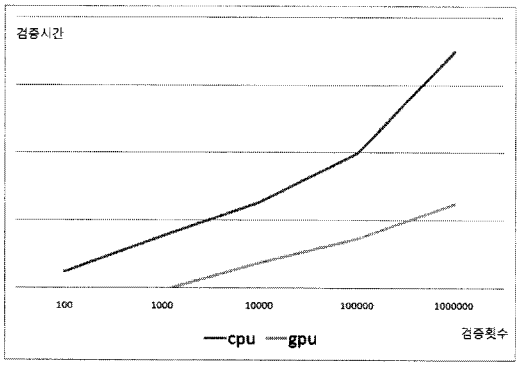
각각의 쓰레드는 그리드내의 블록좌표와 쓰레드 좌표블록을 통해 고유 번호로써 사용[13]될 수 있으며 이 고유 번호에 따라 각 쓰레드에 서로 다른 작업을 할당할 수 있다. 위의 구현 과정에서 각각의 그리드 블록은 일련의 작업을 수행하기 위한 기능 단위로 할당되었으며 CPU의 호스트 프로그램은 각 쓰레드의 일련번호를 통해 GPU내의 쓰레드를 동기화 한다. 복호화 작업의 결과인 마지막 메시지 블록의 패딩은 Global Memory에 저장되어 CPU의 호스트 프로그램에 의해 검증된다.

Ⅳ. 패스워드 무차별 대입공격 결과 분석

GPU를 이용한 인증서 패스워드 무차별 대입공격을 위해 속도 측정 환경으로 Core2Duo 2.1GHz의 CPU와 4GB RAM을 가지는 PC에 NVIDIA Geforce GTX 280와 Geforce 8800GTX의 2개의 그래픽 카드를 사용하였다. 수행속도 체크에서 오는 오버헤드를 피하기 위하여 각 검증 실험에서 100번 공격을 시도한 시간으로 평균 속도를 유추하였다. 속도 측정 결과는 [표 3]와 같다. 프로그램 수행결과 100개의 패스워드에 대한 검증

〔표 3〕 패스워드 공격 시도 횟수에 따른 평균 소요 시간

검증 횟수	검증 소요 시간(Sec)
100	0.0912
1000	0.8540
10000	8.5614
100000	88.2240
1000000	912.6233



〔그림 11〕 검증 소요시간 비교

속도가 0.0982 Sec로 측정 되었으며 이는 회당 평균 0.9ms의 검증속도를 가진다.

이는 2장에서 수행되었던 CPU단위에서 수행되었던 패스워드 검증속도의 약 30배에 달하는 속도 개선이 있음을 보여주고 있다. 또한 패스워드의 검증과정에서 100번째와 1000번째 수행 평균 속도가 급격히 감소하는 결과를 나타났는데 이는 초기 프로그램 수행 시 CPU프로그램과 Global Memory의 I/O 과정에서 발생하는 오버헤드가 있었음을 의미하고 있다.

[그림 11]은 CPU와 GPU의 패스워드 검증에 소요된 시간의 차이를 비교하고 있다. CPU를 통해 패스워드 검증에 소요된 시간은 GPU의 각 코어에서 수행된 쓰레드의 개수에 비례한다는 것을 확인 할 수 있었다. 이 실험은 CUDA를 지원하는 NVIDIA의 일반 그래픽 카드 사양을 통해 이루어졌지만 연산 전용 그래픽카드인 Tesla의 경우 현재 448개의 멀티코어를 지원하고 있으며 6GB의 전용 메모리를 가지고 있어 이를 이용할 경우 패스워드 검출 속도는 이에 몇 수십 배 증가할 것으로 예상된다.

현재 공인인증서 시스템에서 요구하는 패스워드 기준은 대/소문자를 구분하는 영문자 52가지와 숫자 10가지, 그리고 빈칸과 특수문자 32가지를 포함한 총 94개이다. 그러나 현실적으로 상당수의 사용자들이 사용하고 있는 패스워드 조합을 살펴보면 영소문자와 숫자만을 포함하고 있는 36가지 패스워드 조합이 가장 많이 쓰인다는 점을 감안할 때 패스워드 검증을 위한 경우의 수는 그다지 많지 않다는 것을 알 수 있다. 또한 일반 사용자들이 흔히 쓰는 패스워드 유형에 따라 생성된 패스워드 조합으로 공격을 시도할 경우 [표 4]와 같은 결과를 산출 할 수 있다.

〔표 4〕 패스워드 조합에 따른 공격 소요 시간

패스워드 조합	경우의 수	공격 소요시간
영소문자3개+ 숫자5개	$26^3 * 10^5$	약 20일
영소문자4개+ 숫자4개	$26^4 * 10^4$	약 52일
영소문자6개+ 숫자2개	$26^6 * 10^2$	약 354일

이와 같은 패스워드 검출 속도는 기존 워크스테이션급 PC보다 적게는 약 4배에서 많게는 20배 정도 향상된 결과다.

V. 개인키 암호화 패스워드 공격 대응방안

현재의 인증서 개인키 암호화 패스워드 정책은 영소대문자와 숫자, 그리고 특수문자를 포함한 8자리이상의 문자열로서 94가지의 문자 조합이다. 그러나 현실적으로 대다수의 사용자들이 영소문자와 숫자만을 포함한 8자리 문자열을 주로 사용하고 있다. 이는 36가지의 문자 조합만이 주로 사용 된다는 것을 의미한다. 또한 현재 일반사용자들이 사용하고 있는 공인 인증서 파일은 시스템내의 고정적인 폴더경로를 가지고 있다. 만일 공격자가 악성코드 등을 이용하여 대량의 인증서 파일을 획득 하였을 경우 GPU환경을 통하여 인증서 패스워드를 찾아내는 것은 어렵지 않은 일이 될 것이다. [표 5]은 일반 포털사이트와 공인 인증서간 패스워드 인증 정책 비교를 통해 공인 인증서의 개인키 암호화 패스워드가 일반 포털 사이트보다 공격에 더욱 취약한 환경임을 보여주고 있다.

이와 같은 공인 인증서의 패스워드 인증 방식은 전자서명인증체계에서 불가피한 선택 일 수 있지만 사용자의 패스워드 복잡도 증가를 유도하는 방법으로 어느 정도 극복 할 수 있다. 현재 민간 공인 인증서 소프트웨어를 통한 인증서 발급 과정에서 패스워드를 설정할 때 영어소문자와 숫자만을 포함한 문자열 조합으로 입력을 제한하고 있다. 이 경우 패스워드 조합에 따라 공격 시도 가능 횟수는 36^8 에 지나지 않는다. 그러나 패스워드 입력 시 영어소문자와 대문자 그리고 숫자가 조합된 문자열로 입력을 제한 할 경우엔 62^8, 영어 소문자와 대문자, 그리고 숫자와 특수문자가 조합된 문자열로 입력을 제한할 경우엔 94^8의 경우의 수를 갖는다. 현재의 공인

〔표 5〕 포털 사이트와 공인인증서 패스워드 정책 비교

정책	A 포털	공인 인증서
패스워드 검증 방식	서버 인증	블록 패딩 검증
패스워드 유효기간	6개월	1년
입력 오류 제한	3회	무제한 허용 가능

인증서 유효기간이 발급 후 1년의 시간을 갖는다고 할 때 이와 같은 경우의 수를 공격하기는 현재의 GPU 환경에서도 어려운 일이 될 것이다.

하지만 하드웨어의 성능은 지속적으로 발달 하고 있고 GPU의 경우 매년 2배 이상의 연산속도가 증가 하고 있어 단순히 패스워드 조합을 늘리는 것은 궁극적인 해결책이 되지 못하므로 보다 강화된 개인키 보관 정책이 강구되어야 할 것이다.

Ⅵ. 결론

본 논문에서는 현재 공인인증서 시스템에서 사용자의 패스워드를 이용한 개인키 암호화 과정에 대해 살펴보고 개인키 파일만을 이용하여 패스워드 무차별 대입 공격을 통해 패스워드를 획득하는 절차에 대해 설명하였다. 또한 기존 CPU의 연산 처리능력을 능가하는 GPU를 이용하여 개인키 암호화에 사용되는 SEED블록 암호 알고리즘을 수행함으로써 보다 빠른 패스워드 무차별 대입 공격이 시도될 수 있음을 설명하였다. 현재의 공인인증서 개인키 암호화 패스워드는 암호화된 개인키 파일만을 이용하여 패스워드 진위 여부를 확인 할 수 있기 때문에 공격자가 사용하는 시스템 성능에 따라 공격에 대한 성공 가능성이 달라 질 수 있다. 또한 현재 공인인증서의 패스워드 정책은 현재의 하드웨어 발달 수준이 고려되지 않은 최소한의 문자 조합으로 제한하고 있다. 따라서 공인인증서 소프트웨어 수준에서의 단순 패스워드 조합에 대한 제한과 함께 보다 강력한 인증서 키 보관 정책이 검토되어야 할 것이다.

참고문헌

[1] Andrew Zonenberg, *"Distributed Hash Cracker: A Cross-Platform GPU-Accelerated Password Recovery System"*, 2009

[2] 임선간, 강성우, *"블록암호알고리즘 SEED의운영모드표준 (TTAS.KO-12.0025)"*, 2004

[3] Internet X.509 Public Key Infrastructure Certificate Management Protocols, RFC, 2004

[4] Arnold K. L. Yau?, Kenneth G. Paterson and Chris J. Mitchell, *"Padding Oracle Attacks on CBC-mode Encryption with Secret and Random IVs"*, 2006

[5] The GPGPU Resources and Forums, http://www.gpgpu.org/.

[6] The first GPGPU workshop, Proceedings of the first Workshop on General Purpose Processing on Graphics Processing Units, http://www.ece.neu.edu/GPGPU, 2007.

[7] D. L. Cook, J. Ioannidis, A. D. Keromytis, J.Luck *"CryptoGraphics : Secret Key CryptographyUsing Graphics Cards"* CT-RSA, SpringerLNCS 3376, 2005.

[8] NVIDIA CUDA Forum, http://cuda.nvidia.com

[9] Michael Kipper, Joshua Slavkin, Dmitry Denisenko

[10] University of Toronto, *"Implementing AES on GPU Final Report"*, 2009

[11] Owen Harrison and John Waldron, *"AES Encryption Implementation and Analysis on Commodity Graphics Processing Units"*, 2009

[12] 정환석, 이호정, 구본욱, 송정환, *"64비트 마이크로프로세서에 적합한 블록암호 ARIA 구현방안"*, 정보보호학회논문지 제16권 3호, 2006

[13] ETRI, 한양대학교, 염용진, 조용국, *"GPU용 연산 라이브러리 CUDA를 이용한 블록 암호 고속 구현"*, 정보보호학회, 2008

[14] IETF, Security Area, *"X.509(pkix)Document"*, http://www.ietf.org, 1999

[15] 최윤성, 이영교, 이윤호, 박상준, 양형규, 김승주, 원동호, *"삭제된 공인인증서의 복구 및 개인키 암호화 패스워드 검출"*, 정보보호학회논문지, 2007

[16] 강신범, 정현철, *"인터넷 뱅킹 해킹 유형과 대응기술"*, 정보보호학회지, 제15권 제 4호, p.29-38,2005

[17] 정보보호 진흥원 암호인증기술팀, 공인인증서표시를위한기술규격[V1.00], 정보보호진흥원, 2002

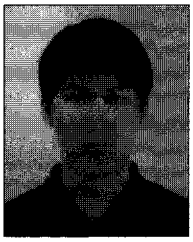
[18] 정보보호 진흥원 암호인증기술팀, SEED 알고리즘을 이용한 개인키 암호화 기술 규격 〔v1.00], 정보보호진흥원, 2004.

[19] 최희봉, 오수현, 홍순좌, 원동호, "*PKI 연동키복구 암호 시스템 설계에 관한 연구*", 정보보호학회논문지12(1), pp. 11-19, 2002.

[20] 이태건, 김종성, 이창훈, 성재철, 이상진, "*다중 운영 모드에 대한 패딩 오라클 공격*", 정보보호학회지, 2006

[21] 김선종, 권정옥, "*금융 보안 서버의 개인키 유출 사고에 안전한 키 교환 프로토콜*", 정보호호학회논문지, 2009

〈著者紹介〉



김 종 희 (Kim Jong Hoi)

학생회원

2007년 3월~현재 : 단국대학교 정보컴퓨터과학부

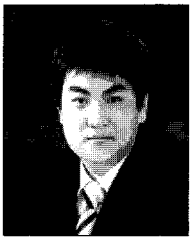
관심분야 : 정보보호

안 지 민 (Ahn Ji Min)

2002년 3월~2007년 2월 : 제주대학교 사범대학 컴퓨터교육과

2008년 3월~현재 : 제주대학교 대학원 컴퓨터교육 수료

관심분야 : 정보보호

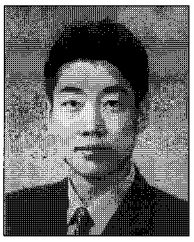


김 민 재 (Kim Min Jae)

2004년 3월~2008년 2월 : 단국대학교 정보컴퓨터과학부

2008년 3월~2010년 2월 : 단국대학교 대학원 컴퓨터과학 석사

관심분야 : 정보보호, 임베디드



주 용 식 (Joo Yong Sik)

1995년 3월~2002년 2월 : 건양대학교 수학과

2009년 3월~현재 : 성균관대학교 정보통신대학원 정보보호학과

관심분야 : 정보보호, 암호이론