AES-NI를 이용한 TPM 기반 암호화 성능 향상 기법

구윤기, 신재복, 박우람, 박찬익
포항공과대학교 컴퓨터공학과
{ggungnae, zstormx, wizrampa, cipark}@postech.ac.kr

Improving Performance of TPM-based Encryption by AES-NI

YounKi Ku, Jaebok Shin, Wooram Park, Chanik Park
Department of Computer Science and Engineering, POSTECH

요 약

TPM (Trusted Platform Module)은 하드웨어 칩셋으로 암호화 키를 하드웨어 기반으로 안전하게 보호하거나, 사용자 환경의 무결성 (integrity)을 검증 (attestation)하는 기능을 제공한다. 그리고 TPM 자체적으로 RSA (Riverst, Shamir, Adleman) 암호화 방법을 제공하나, 낮은 성능으로 인하여 TPM은 암호화 키를 보호하는데 사용하고 기존 AES (Advanced Encryption Standard) 암호화 알고리즘을 이용하여 데이터를 암호화 한다. 하지만 이와 같은 방법은 TPM에서 발생하는 성능 오버헤드에 의하여 암호화 성능을 저하시킨다. 따라서 본 논문에서는 하드웨어 기반 암호화 가속 기법인 AES-NI (Advanced Encryption Standard-New Instructions)를 이용한 TPM 기반 AES 암호화 성능을 분석하여 데이터를 보다 안전하게 보호함과 동시에 AES-NI를 이용함으로써 큰 파일의 경우에는 소프트웨어 기반 AES 암호화 방식보다 최대 35배 빠른 성능으로 TPM에 의하여 발생하는 성능 저하를 보완할 수 있음을 보였다.

1. 서 론

최근 들어 악성코드 감염 및 악의적 목적을 가진 외부 공격으로부터 보안이 중요한 사용자 데이터의 유출이 빈번하게 발생하고 있다. 일반적으로 중요한 데이터들은 대부분 자신의 로컬 디스크에 플레인 텍스트로 저장되는데, PC가 악성 코드에 감염되거나 외부 공격을 받을 경우 데이터가 비교적 쉽게 유출될 수 있다. [1]

이러한 문제를 기존의 데이터 암호화 방법인 소프트웨어 기반 AES 암호화 알고리즘으로 해결하고자 한다면, 소프트웨어 기반 암호화 된 데이터는 사용자의 로컬 저장소나 클라우드 스토리지와 같은 원격 저장소에 저장된다. 하지만 이와 같은 방법은 소프트웨어 암호화 키가 로컬 저장소에 보관되고, 사용자 암호를 기반으로 하여 생성되기 때문에 기기에서 동작 중인 악성 코드에 의하여 외부로 유출될 가능성이 있다.

이러한 문제점을 해결하기 위해, 하드웨어적으로 플랫폼이 안전한지 검증 가능한 신뢰할 수 있는 컴퓨팅 환경(Trusted Computing)이 중요한 보안 이슈로 떠오르고 있다. 신뢰할 수 있는 컴퓨팅 환경을 통해 사용자는 보안 정책이 정확히 적용 되었는지 판단 할 수 있다.

이와 같은 환경을 구성하기 위하여 하드웨어 보안 칩셋인 TPM(Trusted Platform Module)이 TCG에 의해 정의되었다. TPM은 암호화 키 보호, PCR(Platform Configuration Register)에 저장된 무결성 정보를 이용한 원격 검증 및 안전한 플랫폼 구성, 데이터 바인딩 (binding) 등의 기능을 제공한다. 무결성이 검증된 환경에서만 바인딩 된 데이터가 언바인드 되어 사용되도록 구성함으로써, 악성 코드가 실행되는 경우나, 시스템 코드가 변조된 상태에서는 데이터를 사용할 수 없다[2]. 하지만 TPM만으로 데이터를 암호화 하기엔 성능이 매우 떨어지기 때문에[3], 일반적으로 TPM을 통해 AES(Advanced Encryption Standard)와 같은 암호화 알고리즘에서 사용할 암호화 키를 바인드 하여 관리하고, 그 암호화 키를 쓰기 위해서는 언바인드 과정이 필요하다.[4]

위에서 사용하는 AES 암호화 알고리즘의 성능을 개선하기 위해 인텔에서 제안한 하드웨어 기반 AES-NI (Advanced Encryption Standard-New Instructions) 암호화 기법이 있다[5]. AES-NI는 기존 AES 암호화 알고리즘을 하드웨어 기반으로 암호화를 가속화하는 새로운 암호화 명령어 집합이다.

본 논문에서는 AES-NI를 이용해 TPM기반 암호화 성능 향상을 위해 몇 가지 성능 평가를 하였다. TPM을 통한 데이터 암호화 시 TPM이 지원하는 키 관련 함수의 성능을 분석함으로써 TPM이 가지는 오버헤드를 측정하고, 소프트웨어 기반 AES 알고리즘과 하드웨어 기반 AES-NI의 성능 분석 비교를 통해서 AES-NI가 소프트웨어 기반 AES 알고리즘에 비교해 얼마나 성능상 이점을 가지는지 분석한다. 최종적으로 TPM으로 AES 암호화 키를 관리하였을 때, 소프트웨어 기반 AES 암/복호화 성능과 하드웨어 기반 AES-NI의 성능 분석을 통해, AES-NI 암/복호화를 수행함으로써 암호화

750

성능 향상을 통해 TPM에서 발생하는 성능 저하를 보완하면서 큰 파일의 경우에는 소프트웨어 기반 AES 암호화보다 빠름을 보이고자 한다.

2. 암호화 성능 분석

TPM과 AES, AES-NI를 위해 사용한 테스트 환경은 다음과 같다. 인텔 i7-2600 프로세서, 램 8GB, 리눅스 커널 버전 3.2.23에 AES-NI 모듈을 추가하였고, AES-NI 성능 벤치마크 툴[6]을 사용하였다.

AES 알고리즘은 AES-256-CBC를 사용하였다.

2.1. TPM 암호화 성능

일반적으로 TPM을 이용하여 AES와 같은 암호화 알고리즘에서 사용할 암호화 키를 바인드 하여 관리하고, 실제 데이터 암호화는 소프트웨어 암호화 알고리즘을 사용한다. 암호화 된 데이터를 사용하기 위해서는 TPM을 이용하여 바인드 된 암호화 키를 언바인드 한 후, 그 키를 통해 사용자의 데이터를 전체/부분적으로 접근 할 수 있다. 그러나 TPM을 이용해 키를 바인드/언바인드 하는 과정 역시 성능 오버헤드를 가지고 있기 때문에 전체 데이터 암호화 성능의 저하가 발생한다.

먼저, TPM으로 AES 암호화 키를 관리 시, TPM의 키 관리 오버헤드를 측정하기 위하여, TPM이 제공하는 키 생성 함수(Tspi_Key_Create)와 생성된 키를 언바인드 하는 함수(Tspi_Data_Unbind)의 성능 평가를 하였다. 성능 평가 시 trousers 0.3.7, tpm-tools 1.3.7을 사용하였다.[7]

표 1. TPM의 키 생성과 언바인드

TPM 함수	시간(초)
바인딩 키 생성(Tspi_Key_Create)	3.75
언바인드(Tspi_Data_Unbind)	1.29

표 1에서 TPM이 제공하는 키 생성 함수는 3.7초 정도의 성능 오버헤드를 가진다. 하지만 암호화 키를 생성하는 과정은 한번만 발생하기 때문에 생성 이후 데이터 암호화의 성능에는 영향을 끼치지 않는다. 또한, 바인드 된 키를 언바인드 할 시, 1.3초 정도의 성능 오버헤드를 가진다. 키는 데이터를 복호화 할 때마다 언바인드 해야 하므로 TPM 기반의 데이터 암호화 구현 시 주요한 성능 저하 요인이다.

2.2. AES / AES-NI 암호화 성능

TPM을 제외한 AES와 AES-NI의 성능 분석을 위해 인텔에서 제공하는 AES-NI 라이브러리 벤치마크 툴을 사용하였고, C 라이브러리에 기반한 AES 암호화 성능 테스트와 AES-NI에 기반한 AES 암호화 성능 테스트를 하였다. 일반적으로 보안이 필요한 데이터는 대부분 문서 및 사진과 같은 작은 용량의 파일이기 때문에 본 실험에서는 이와 같은 상황을 가정하고 16KB - 16MB 사이의 데이터에 대한 암호화 시간을 각 10번씩 측정하여 평균값을 계산하였다.

표 2. 4KB-4MB에서의 AES/AES-NI 암호화 시간

파일 크기	암호화 방법		성능 향상(배)
	AES (초)	AES-NI (초)	
16 KB	0.0052	0.0001	35.41
32 KB	0.0045	0.0001	28.89
64 KB	0.0089	0.0002	36.47
128 KB	0.0176	0.0005	36.16
256 KB	0.0352	0.0010	35.33
512 KB	0.0705	0.0019	35.80
1 MB	0.1447	0.0040	36.07
2 MB	0.2876	0.0080	35.76
4 MB	0.5769	0.0161	35.90
8 MB	1.1495	0.0321	35.81
16 MB	2.3295	0.0645	35.12

표 2에서 AES-NI 방식은 소프트웨어 기반 AES 암호화보다 데이터 크기에 상관없이 약 35배 빠르다.

2.3. AES-NI를 TPM 환경에 적용

표 1에서 TPM 함수 중 키를 언바인드 하는데 약 1.3초의 수행 시간을 가진다는 것과, 표 2에서는 하드웨어 기반 AES-NI가 소프트웨어 기반 AES 알고리즘보다 약 35배 성능 향상이 있음을 보였다.

AES-NI를 사용해 TPM기반 암호화 성능을 개선시키기 위해서, 본 논문에서 제안하는 것은 TPM에서 키를 관리할 때 동일한 AES 키를 관리하되 데이터 암/복호화 할 때는 AES-NI를 쓰는 것이다. 이렇게 함으로써 AES-NI의 성능상 이점을 고스란히 이용할 수 있다.

이 경우 데이터 암/복호화 시 TPM에서 AES 암호화 키의 언바인드 성능 오버헤드인 약 1.3초정도가 추가적으로 발생한다.

751

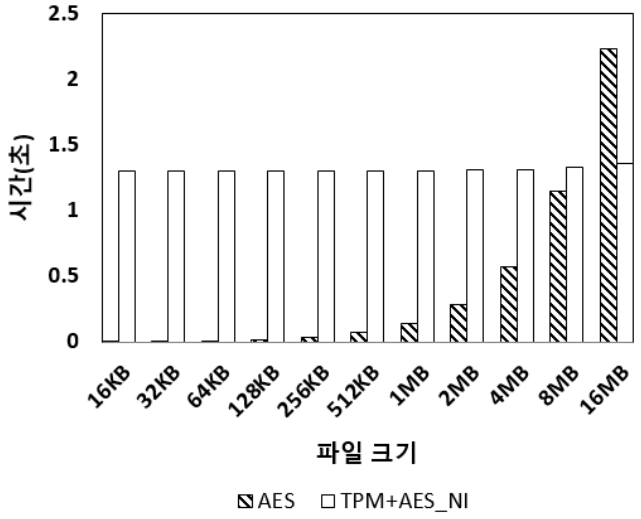

그림 1 AES/AES-NI를 TPM기반 암호화에 적용 시 성능 측정

그림 1은 AES-NI와 소프트웨어 기반 AES 암호화를 TPM기반 암호화에 적용한 결과이다. 각 막대 그래프에는 TPM 키 언바인드 시간인 약 1.3초가 포함되어 있다.

작은 문서나 그림 파일의 경우에는 암호화 시간이 짧기 때문에 상대적으로 TPM 오버헤드가 크다. 그림 1과 표 2에서 16KB에서 8MB의 작은 파일의 경우에는 암호화 시간은 1.1495초보다 작기 때문에 TPM의 오버헤드(1.3초)가 더 크다. 그렇기 때문에 AES-NI를 써도 TPM 오버헤드가 더 크다. 하지만 음성파일 및 PPT같은 8MB에서 16MB 사이의 파일일 경우 AES 암호화 시간이 1.1495초에서 2.3295초 이므로 TPM의 오버헤드인 1.3초와 비슷하거나 크다. 결국 파일 크기가 8MB와 16MB 사이일 경우에는 AES-NI를 적용하게 되면 소프트웨어 기반 AES와 거의 성능차이가 없다. 파일 크기가 16MB를 넘어가는 시점에서는 암호화 시간이 2.2395초보다 커지므로 TPM 언바인드 오버헤드가 전체 성능에서 차지하는 비중이 적기 때문에 소프트웨어 기반 AES 암호화보다 성능이 뛰어나다.

그러므로 AES-NI를 TPM환경에서 파일 암호화에 적용할 때, 작은 파일의 경우에는 성능 개선의 효과가 미미하다고 볼 수 있으며 파일 크기가 커지면 커질수록 상대적으로 TPM 오버헤드가 작기 때문에 하드웨어 기반 AES-NI를 사용함으로써 암호화 성능 최적화가 가능하다.

3. 결론 및 향후 연구

본 논문에서는 소프트웨어 기반 AES 암호화 방식에 비해 인텔에서 제공하는 프로세서 명령어 집합인 AES-NI가 가지는 장점을 분석하고, TPM의 암호화 성능상 약점을 분석하였다. 기존 소프트웨어 기반 AES 암호화 방식 대신 암호화 방법으로 TPM 기반 AES-NI를 사용함으로 16MB 이상인 파일의 경우에는 AES-NI의 빠른 암호화를 통해 암호화 성능을 크게 개선 할 수 있다는 것을 보였다. 최근 고화질 사진, 대용량 미디어 파일 등, 큰 용량의 파일이 증가하고 있기 때문에, TPM 기반 AES-NI 암호화 방식으로 이러한 고용량 파일의 암호화를 빠르게 할 수 있다.

파일 단위 암호화의 단점은 암/복호화 할 때마다 TPM에서 해당 키를 언바인드 해야 된다는 것이다. 언바인드 할 때마다 약 1.3초의 오버헤드가 발생하므로 한 파일을 읽거나 쓰는데 최소한 1.3초가 걸리게 된다. 그래서 파일 단위로 암호화를 하지 않고 블록 단위로 암호화를 하거나 스토리지 단위로 암호화를 하는 방식이 향후 연구 방향이다.

Acknowledgement

이 논문은 2011년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구(No. 2011-0016972) 및 미래창조과학부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음 (NIPA-2013-H0301-13-3002)

참 고 문 헌

[1] McAfee Labs, McAfee Threats Report: Second Quarter, 2012
[2] Trusted Computing Group Website, TPM Main Specification, http://www.trustedcomputinggroup.org
[3] Ramakrishna Kotla and Tom Rodeheffer, Pasture: Secure Offline Data Access Using Commodity Trusted Hardware, OSDI 2012
[4] W. Park, C. Park, Data Firewall: A TPM-based Security Framework for Protecting Data in Thick Client Mobile Environment, JCSE 2011
[5] Intel, Securing the Enterprise with Intel AES-NI, 2010
[6] Intel Website, Intel AESNI Sample Library, http://software.intel.com/en-us/articles/download-the-intel-aesni-sample-library
[7] TrouSerS Website, http://trousers.sourceforge.net

752