한국산학기술학회논문지
Vol. 12, No. 3, pp. 1356-1366, 2011

TPM 명령어 인가 프로토콜에 대한 내부자 공격 취약점 분석 및 대응책

오두환[1], 최두식[1], 김기현[2], 오수현[1], 하재철[1*]
[1]호서대학교 정보보호학과, [2]충북대학교 컴퓨터공학과

Vulnerability Analysis of Insider Attack on TPM Command Authorization Protocol and Its Countermeasure

Doo-Hwan Oh[1], Doo-Sik Choi[1], Ki-Hyun Kim[2], Soo-Hyun Oh[1] and Jae-Cheol Ha[1*]
[1]**Dept. of Information Security, Hoseo University,**
[2]**Dept. of Computer Eng., Chungbuk National University**

요 약 TPM(Trusted Platform Module)은 신뢰된 컴퓨팅 환경을 구성하기 위해 플랫폼 내부에 부착된 하드웨어 칩이다. TPM의 핵심 명령어들 중에서 정당한 사용자만이 TPM을 사용할 수 있도록 명령어에 대한 인가(authorization)가 선행되어야 한다. 즉, 사용자는 TPM 칩에게 명령어 인가를 받기 위해 OIAP(Object-Independent Authorization Protocol)이나 OSAP(Object-Specific Authorization Protocol) 프로토콜을 사용한다. 그러나 최근 Chen과 Ryan은 단일 플랫폼 내의 멀티유저 환경에서 내부 공격자가 TPM으로 위장하는 공격에 취약함을 밝히고 그 대응책으로 SKAP(Session Key Authorization Protocol) 프로토콜을 이론적으로 제안하였다. 본 논문에서는 실제 PC에 TPM 칩을 장착한 상태에서 OSAP에 대한 내부자 공격이 실제로 가능함을 인가 프로토콜 실험을 통해 확인하였다. 또한 이전의 대응 방법인 SKAP에서 명령어 구조 변경 및 대칭 키 암호 연산이 필요했던 점을 개선하여 보다 효과적인 내부자 공격 대응책을 제안하였다. 제안 프로토콜에서는 OSAP 명령어 체계만 간단히 수정하고 사용자 및 TPM 칩에서 각각 RSA 암복호 연산 한번만 추가하면 내부자 공격을 막을 수 있다.

Abstract The TPM(Trusted Platform Module) is a hardware chip to support a trusted computing environment. A rightful user needs a command authorization process in order to use principal TPM commands. To get command authorization from TPM chip, the user should perform the OIAP(Object-Independent Authorization Protocol) or OSAP(Object-Specific Authorization Protocol). Recently, Chen and Ryan alerted the vulnerability of insider attack on TPM command authorization protocol in multi-user environment and presented a countermeasure protocol SKAP(Session Key Authorization Protocol). In this paper, we simulated the possibility of insider attack on OSAP authorization protocol in real PC environment adopted a TPM chip. Furthermore, we proposed a novel countermeasure to defeat this insider attack and improve SKAP's disadvantages such as change of command suructures and need of symmetric key encryption algorithm. Our proposed protocol can prevent from insider attack by modifying of only OSAP command structure and adding of RSA encryption on user and decryption on TPM.

Key Words : TPM, OSAP, SRK secret, Authorization, Insider attack

1. 서론

TPM(Trusted Platform Module) 칩은 안전하고 신뢰할 수 있는 컴퓨팅(TC : Trusted Computing) 환경을 구성하기 위해 컴퓨터의 플랫폼 장착되어 사용되고 있다. 현재, TPM 칩은 안전한 플랫폼을 지원하기 위해 PC 및 노트북

*교신저자 : 하재철(jcha@hoseo.edu)
접수일 2011년 01월 25일 수정일 2011년 02월 10일 게재확정일 2011년 03월 10일

내부에 장착되어 사용되고 있다. 향후 다양한 장와 스마트폰 플랫폼에서 신뢰성 제공을 위해 모바일용 TPM(MTM : Mobile Trusted Platform Module)으로 응용될 수도 있다[1, 2].

신뢰된 컴퓨팅을 위해 조직된 TCG[3]는 TPM 칩에 대한 표준화를 진행하였고, TPM 칩의 사양 및 운영방법은 ISO/IEC 표준으로 제정되어 있다[4-7]. TPM은 기본적으로 신뢰된 컴퓨팅 환경을 구축하기 위해 RTM(Root of Trust for Measurenemt), RTS(Root of Trust for Storage), RTR(Root of Trust for Reporting) 보안 기능의 근간이 된다[4].

TPM 칩을 사용하기 위해서는 표준에서 미리 지정된 120여개의 명령어를 이용한다[7]. 이 명령어들은 인가(authorization) 여부에 따라 나눌 수 있는데, 플랫폼 사용자가 중요한 명령어를 사용할 경우에는 한 번 또는 두 번의 인가가 필요한 명령어들을 사용해야 한다. 명령어 인가를 위해서는 인가 데이터(authorization data)인 authdata라는 것과 명령어 인가 프로토콜(command authorization protocol)을 이용한다.

authdata는 사용자가 자신이 정당한 사용자임을 나타내는 사용자 비밀 정보(usage secret)를 이용하여 만들게 되고, 사용자는 이를 TPM으로 전송하여 자신이 정당한 사용자임을 TPM에게 증명하게 된다. 여기서 사용자 비밀 정보는 보통 사용자가 입력한 패스워드를 SHA-1[8]으로 해쉬한 값을 이용한다. TPM에서는 명령어 인가 프로토콜로 OIAP (Object- Independent Authorization Protocol)이나 OSAP (Object- Specific Authorization Protocol)을 사용한다.

그러나 최근 Chen과 Ryan은 단일 플랫폼에서 여러 명의 사용자가 이용하는 멀티유저 환경에서는 내부 공격자가 TPM으로 위장하여 다른 사람의 공유 비밀정보(Shared secret)을 알아내고 이를 이용하여 정당한 사용자의 새로운 인가 데이터 newAuth를 추출해 낼 수 있음을 이론적으로 밝혀냈다. 공격자가 다른 사람의 newAuth를 알면 저장되어 있는 비밀 정보를 알 수 있게 되므로 매우 위협적 공격이 된다. Chen과 Ryan은 이러한 내부자 공격에 취약함을 밝히고 대응책으로 SKAP(Session Key Authorization Protocol) 프로토콜을 제안하기도 하였다[9].

본 논문에서는 Chen과 Ryan의 내부자 공격 방법이 실제 TPM 칩을 장착한 PC 플랫폼 환경에서 실제로 동작할 수 있는지 그 가능성을 실험하였다. 실험 결과, 공격자는 정당한 사용자와 TPM 칩 간의 전송되는 메시지를 모두 중간에서 가로채기(intercept)를 할 수 있었으며 이 경우 사용자의 newAuth를 복구할 수 있음을 검증하였다. 또한 이전의 대응 방법인 SKAP에 비해 명령어 구성이 보다 효과적으로 내부자 공격에 대응할 수 있는 새로운 방법을 제안하고 비교 분석하였다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로서 TPM 칩의 개요, TPM 명령어 인가 과정, Chen과 Ryan의 내부자 공격과 대응 방법을 설명한다. 3장에서는 제안하는 프로토콜을 설명하고, 4장에서는 실제 컴퓨팅 환경에서 내부자 공격에 대한 실험적 결과를 설명하고 제안하는 프로토콜의 효율성을 비교한다. 마지막으로 5장에서 결론을 맺는다.

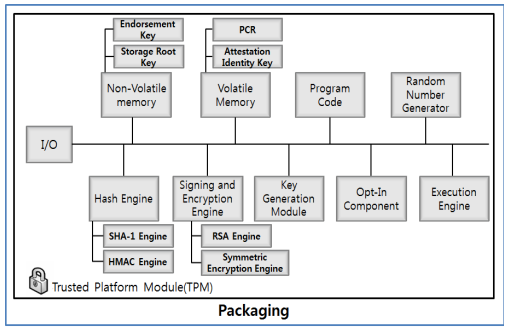
2. 관련 연구

2.1 TPM 칩 개요

2.1.1 TPM 칩 구성요소

TPM 칩은 신뢰 컴퓨팅 환경을 조성하기 위해서 RTM, RTS, RTR과 같은 기능들을 제공하여야 하며 이러한 기능들을 제공하기 위해서는 다양한 암호학적 연산, 키 생성, 키 저장, 외부와의 통신 등의 다양한 기능을 제공하여야 한다[5]. 따라서 TPM은 이러한 기능들을 제공하기 위해 그림 1과 같은 다양한 컴포넌트로 구성되어 있다.

그림 1에 있는 각 컴포넌트들의 주요 기능은 다음과 같다.



[그림 1] TPM의 기본구조

- Signing and Encryption Engine : 다양한 암호학적 연산을 위한 프로세서
- Key Generation : 2048비트 RSA 키 쌍 생성
- HMAC Engine : 인가 값에 대한 검증 및 응답 명령어 구성 시 사용하는 해쉬 MAC 엔진
- SHA-1 Engine : SHA-1 해쉬 연산을 수행
- RNG : nonce값 또는 키 생성을 위한 임의의 숫자 생성기
- Execution Engine : 칩 운영체제와 명령어를 처리하

는 엔진
- Non-Volatile Memory : 비휘발성 메모리로서 SRK와 EK를 저장
- Volatile Memory : 휘발성 메모리로서 PCR 레지스터와 AIK를 저장

2.1.2 TPM 키 구조

TPM의 제조사가 사용자에게 TPM을 전달할 때에게 TPM의 소유권이 없으며 TPM의 비휘발성 메모리에는 TPM의 제조 과정에서 생성되는 EK(Endorsement Key)가 존재하게 된다. EK는 TPM을 식별할 수 있는 TPM만의 고유의 비밀 키로서 2048비트 RSA 키 쌍으로 이루어져 있다. 또한, EK는 절대 TPM 외부로 노출되지 않으며 TPM의 소유권 획득과 RTR에서 사용되는 AIK (Attestation Identity Key)를 생성하는 경우에만 사용된다.

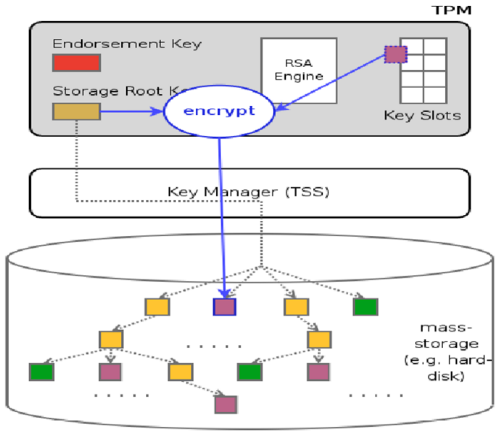
사용자가 TPM을 사용하기 위해서는 TPM의 소유권을 획득하여야 한다. 소유권을 획득하기 위해서는 TPM_ TakeOwnerShip이라는 명령어를 사용하게 된다. 사용자가 이 명령어를 사용하게 되면 TPM은 SRK(Storage Root Key)를 생성하고 이를 비휘발성 메모리에 저장한다. SRK는 2048비트 RSA 키 쌍으로 EK와 마찬가지로 외부로 노출되지는 않는다.

TPM은 데이터 암복호나 서명에 사용되는 키를 모두 내부에 저장할 수 없기 때문에 모든 키는 SRK를 이용하여 계층적으로 관리하게 된다. 즉, TPM은 보호된 영역에 최상위 키인 SRK를 저장하고 새롭게 생성된 키들은 부모 키(예, SRK)로 암호화하여 일반 저장장치에 저장한다. 이후, 암호화 되었던 키를 사용할 때는 그 키의 부모 키로 복호화하여 사용하게 된다. TPM에서는 이러한 키들을 그림 2와 같이 SRK 키를 최상위로 하는 트리형식으로 관리하고 있다.

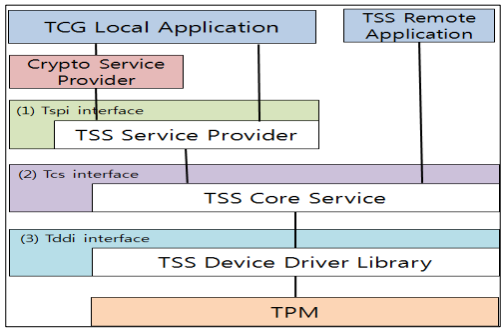
2.1.3 TSS(TCG Software Stack)

TPM은 하드웨어이기 때문에 이에 대한 드라이버와 사용자 인터페이스가 필요하다. TPM 드라이버는 각 제조업체가 제공하고 있으며 사용자 인터페이스는 TSS를 이용함으로써 제공받을 수 있다. TSS란 TPM에 대한 사용자 인터페이스를 제공하고, TSS를 이용하여 응용 프로그램을 개발함으로써 사용자가 TPM의 기능을 사용할 수 있도록 도와준다[10]. 그림 3은 TSS의 기본 구조를 나타내고 있다.

사용자 프로그램과 TPM 사이에는 총 4개의 계층 즉, CSP(Crypto Service Provider), TSP(TSS Service Provider), TCS(TSS Core Service), TDDL(TSS Device Driver Library)이 존재한다. 사용자는 TSS 계층을 이용하여 개발된 응용 프로그램을 사용하여 TPM을 동작시킬 수 있다.



[그림 2] TPM 키 계층 구조



[그림 3] TSS 기본 구조

TPM 칩은 일반적으로 여러 사용자가 하나의 칩을 사용할 수 있는 멀티 유저 환경을 기반으로 설계되었다. TCS는 서버(server)처럼 백그라운드로 운영이 되며 TSP가 여러 명의 클라이언트(client)와 같은 역할을 함으로써 다수의 사용자가 사용할 수 있게 된다. 이러한 멀티 유저 환경에서는 사용자들이 모두SRK를 알고 있음을 전제로 하는데 이러한 설계상의헛점을 이용하여 이후에 설명하는 내부자 공격이 가능하다.

2.1.4 TPM 명령어 사용

TPM 칩의 동작을 위해서는 120여개의 명령어를 요구-응답(Request-Response) 방식으로 사용하게 된다[7]. 이 명령어들은 인가 여부에 따라서 3종류로 나눌 수 있는데, 첫 번째로는 명령어 인가가 필요 없는 명령어, 두 번째는 한 번의 인가가 필요한 명령어, 세 번째는 두 번의 인가가 필요한 명령어로 나눌 수 있다. 사용자가 중요한 명령

어를 사용할 경우에는 한 번 또는 두 번의 인가가 필요한 명령어들을 사용해야 한다.

TPM 명령어 인가를 받기 위해서는 인가 데이터 authdata라는 것과 명령어 인가 프로토콜을 이용한다. 사용자는 사용자 비밀 정보(usage secret)를 이용하여 authdata를 만들게 되고 이를 TPM으로 전송하여 자신이 정당한 사용자임을 증명하게 된다. 여기서 usage secret은 사용자가 입력한 패스워드를 SHA-1으로 해쉬한 160비트 값을 이용한다.

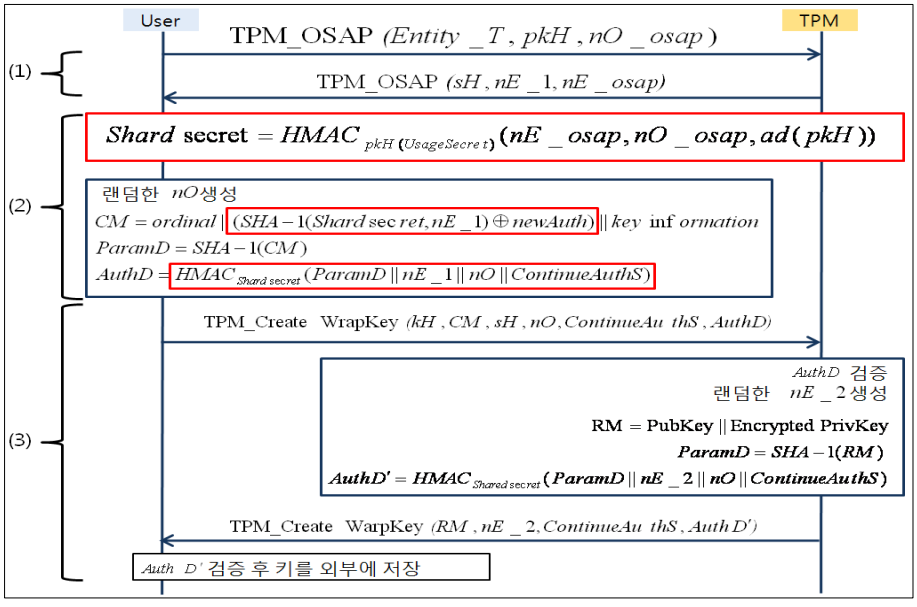
일반적으로 usage secret에는 TPM 소유권 획득시 입력하게 되는 소유자 비밀 정보(Owner secret), SRK(Storage Root Key)를 사용하기 위한 SRK 비밀 정보(SRK secret), 그리고 플랫폼에서 TPM에 의해 관리되고 있는 키들을 사용하기 위한 키 사용 비밀 정보(key usage secret) 등이 있다. 또한, TPM에서는 명령어 인가 프로토콜로 OIAP과 OSAP를 이용한다.

2.2 TPM 명령어 인가 및 내부자 공격

2.2.1 OSAP

TPM의 명령어 인가 프로토콜 중 OSAP은 개체 명시적인 인증 프로토콜로서 새로운 키를 생성하거나 특정한 키를 이용하여 데이터 암호(sealing)를 수행하는 명령어에 대한 인가를 수행한다. 사용자는 명령어 인가를 받기 위해 TPM_OSAP이라는 명령어를 먼저 TPM으로 전송하여 인가 세션을 형성한다. 그림 4는 OSAP을 이용하여 명령어 인가를 수행하는 과정을 예로 나타낸 것이다. 그림 4의 예에서 사용되는 실제 명령어는 TPM_CreateWrapKey라는 명령어로서 새로운 RSA 키 쌍을 TPM 내부에서 생성하는 명령어이다. 사용되는 파라미터는 다음과 같다.

- Entity_T : 개체 타입
- pkH : 생성할 키의 부모 키에 대한 핸들 값
- nO_osap : 사용자가 생성한 랜덤 값
- sH : 현재 인가 세션의 핸들 값
- nE_osap : TPM이 생성한 랜덤 값
- nE_1, nE_2 : TPM이 생성한 랜덤 값
- pkH(UsageSecret) : 생성하고자 하는 키의 부모 키에 대한 사용 비밀 정보
- Shared secret : TPM과 사용자가 공유되는 비밀 정보이고 AuthD와 AuthD'을 생성할 때 HMAC의 비밀키로 사용되며, newAuth를 암호화할 때도 사용
- nO : 사용자가 생성한 랜덤 값
- ordinal : 명령어 구분자
- newAuth : 생성하고자 하는 키를 사용하기 위한 비밀 정보로서 사용자 패스워드를 SHA-1으로 해쉬한 값
- CM : ordinal + 요청 명령어를 구성하는 파라미터들로 newAuth를 암호화한 값과 생성할 키 정보를 포함
- ParamD : SHA-1(CM) 또는 SHA-1(RV)
- AuthD, AuthD' : 인가 데이터



[그림 4] OSAP를 이용한 TPM_CreateWrapKey 명령어 인가

- ContinueAuthS : 인가 세션 플래그
- RM : return code + ordinal + 응답 명령어를 구성하는 결과 값들
- kH : 키의 핸들 값

OSAP를 이용한 TPM_CreateWrapKey명령어 인가 절차는 다음과 같다.

① 사용자가 TPM에게 Entity_T, pkH, nO_osap이 포함된 TPM_OSAP 요청 명령어를 TPM으로 전송한다.

② TPM은 sH, nE_1, nE_osap이 포함된 TPM_OSAP 응답 명령어를 사용자에게 전송한다.

③ 사용자와 TPM은 pkH(UsageSecret)를 사용하여 Shared secret을 생성한다.

④ 사용자는 Shared secret과 nE_1를 SHA-1한 후 newAuth와 XOR하여 newAuth를 암호화한 후 생성할 키 정보(key information)와 함께 요청 명령어에 필요한 파라미터들을 구성한다. 단, 여기서 암호화는 간단한 XOR 연산만으로 처리하고 있다는 점을 주의해야 한다. 사용자는 ordinal과 파라미터들을 SHA-1으로 해쉬하여 ParamD를 생성한다. 사용자는 랜덤값 nO를 생성하고 ParamD, nE_1, nO, continueAuthS의 정보를 Shared secret를 키로 한 HMAC을 계산한다. AuthD를 생성한 후 사용자는 AuthD와 함께 여러 파라미터를TPM_CreateWrapKey 요청 명령어에 포함하여 TPM에게 전송한다.

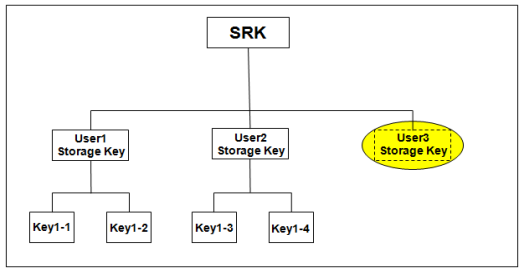
⑤ TPM은 요청 명령어를 수신한 후에 Shared secret를 이용하여 AuthD를 계산하고 사용자가 보낸 AuthD와 같으면 명령어를 처리하고, AuthD가 다르면 명령어를 처리하지 않고 사용자에게 에러 코드를 전송한다. 그 다음 TPM은 랜덤 값 nE_2를 생성한 후에 Shared secret을 이용하여 응답 명령어에 대한 AuthD'를 계산하고 사용자에게 응답 명령어를 보낸다.

⑥ 사용자는 응답 명령어를 TPM으로부터 수신한 후 자신이 알고 있는 Shared secret을 이용하여 AuthD'를 계산한 다음 TPM으로부터 수신한 AuthD'과 비교를 통해 응답 명령어를 검증한다.

2.2.2 OSAP에 대한 내부자 공격

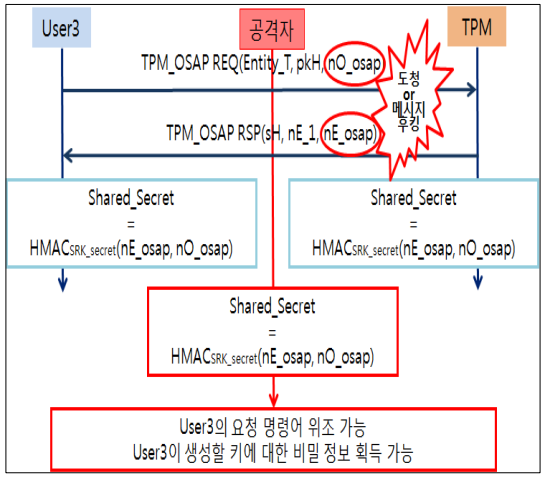
TPM 국제 표준 문서 디자인 규정 파트 14.5, 14.6장에서는 SRK secret이 여러 사용자에게 공유될 수 있는 멀티유저 환경을 가정하였다[5]. Chen과 Ryan은 이러한 약점을 이용하여 OSAP에 대한 내부자 공격을 제시하고 이에 대응할 수 있는 프로토콜인 SKAP을 제안하였다.

하나의 플랫폼을 여러 명이 사용하는 멀티유저 환경에서는 사용자들이 SRK를 사용하기 위한 비밀 정보인 SRK secret 공유하고 있어야 한다. 예를 들면 그림 5와 같이 여러 명의 사용자가 TPM이 장착된 하나의 플랫폼을 사용하고 있을 때 새로운 사용자 User3이 SRK의 자식키인 User3 Storage Key를 생성하기 위해서는 SRK secret을 알고 있어야 한다. 그렇지만 SRK secret은 SRK의 자식키들을 사용하고 있는 다른 사용자인 User1과 User2에게도 알려져 있다.



[그림 5] OSAP에 대한 내부자 공격 환경

그림 6은 내부자 중 User1과 User2 중 한 명이 공격자가 되어 User3과 TPM과의 통신 중에 명령어 가로채기 공격을 시도한다고 가정했을 경우, 공격을 수행하는 과정을 나타낸 것이다.



[그림 6] OSAP에 대한 내부자 공격 과정

그림 6과 같이 공격자가 User3과 TPM 사이에 위치하여 요청 및 응답 명령어를 가로채는 프로그램을 구동하거나 TPM을 가장하는 프로그램을 구동한다고 가정해 보자. 공격자는 이러한 프로그램을 통하여 송수신되는 메시지를 획득할 수 있다. 따라서 공격자는 메시지 가로채기를 통해 nO_osap과 nE_osap을 알아낼 수 있고 이를 이용

하여 Shared secret을 다음과 같이 계산할 수 있다.

$$HMAC_{pkH(UsageSecret)}(nE_{osap}, nO_{osap})$$

여기서 pkH(UsageSecret)는 생성하고자 하는 키의 부모 키에 대한 키 사용 비밀 정보이므로 SRK secret을 키를 의미하는데 이는 공격자인 User1이나 User2도 모두 알고 있기 때문이다. 따라서 공격자는 User3과 TPM이 공유하고 있는 Shared secret을 공격하여 추출할 수 있다.

이 공격이 성공하면 공격자는 가로챈 CM 값도 알 수 있으므로 User3의 newAuth도 알게 된다. 공격자가 newAuth를 알았다는 의미는 다음에 User3이 생성한 키로 데이터를 암호화해도 이 키를 이용하여 User3의 데이터를 복호화할 수 있어 매우 위협적인 공격이 된다.

2.2.3 Chen과 Ryan의 대응 방법

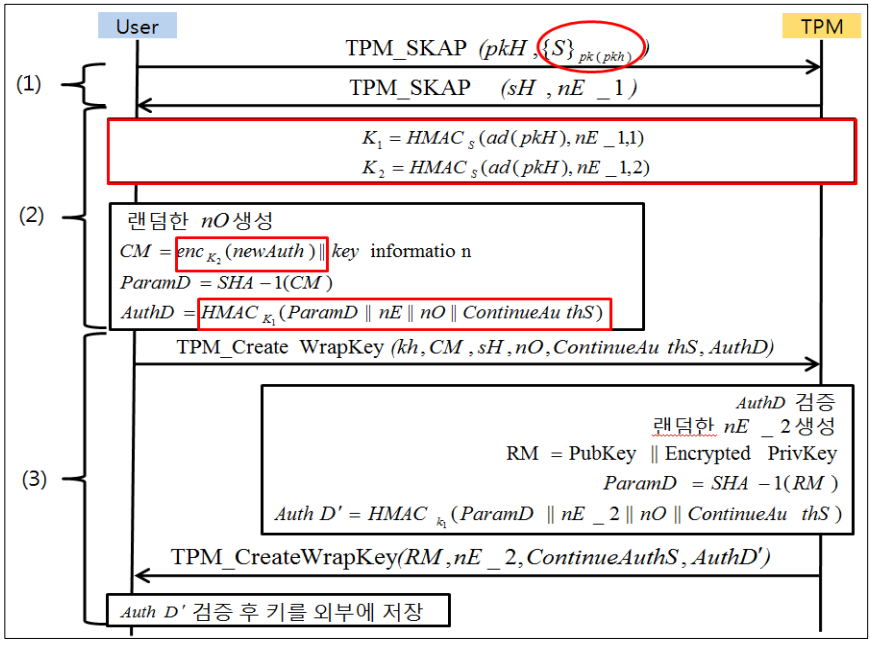
Chen과 Ryan은 TPM 내부자 공격과 함께 이에 대한 대응 방법으로 SKAP를 소개하였다. OSAP을 이용한 명령어 인가의 취약점은 SRK secret을 플랫폼 내의 모든 사용자들이 공유하고 있다는 사실이다.

그림 7은 저자들이 제안한 SKAP 프로토콜을 이용하여 TPM_CreateWrapKey 명령어 인가를 수행하는 과정을 도시한 것이다. 단계 (1)에서 정당한 사용자(User3)는 랜덤값 S를 선택하고 부모키의 공개 키(예: SRK)로 암호화하여 TPM으로 전송한다. TPM에서 관리하는 모든 키들의 개인키는 TPM 내부에서만 사용될 수 있기 때문에 암호화된 S는 TPM만이 복호화할 수 있다. 따라서 S는 사용자(User3)와 TPM만이 공유하는 값이 된다. 따라서 같은 플랫폼 내의 다른 사용자(User1 혹은 User2)들은 S 값을 알 수 없다.

그 후 사용자와 TPM은 단계 (2)와 같이 S를 HMAC 키로 하여 세션 비밀 정보 K_1과 K_2를 생성할 수 있고 K_1은 명령어에 대한 무결성 유지 및 인가 데이터 생성하기 위해 사용되며 K_2는 생성할 키 사용하기 위한 비밀 정보인 newAuth의 기밀성을 유지하기 위해 암호화 키로 사용된다.

이 대응 방법의 핵심은 한명의 사용자는 자신만이 아는 비밀키를 공통의 공개키로 암호화하여 TPM에게 전송함으로써 자신과 이를 복호화할 수 있는 TPM 칩만이 비밀 키 S를 공유하고자 하는 방법이다. 따라서 같은 플랫폼 내의 악의적인 공격자들은 K_1과 K_2를 생성하는데 사용되는 HMAC의 입력들은 메시지 가로채기를 통해서 알 수 있지만 HMAC 키로 사용되는 S를 알 수 없기 때문에 K_1과 K_2를 복구할 수 없게 된다.

그러나 이 대응 방법은 SKAP 다음에 사용하는 명령어마다 서로 다른 CM이나 AuthD를 생성하므로 매우 복잡한 구조를 갖게 된다. 즉, 명령어 인가를 위해 SKAP 프로토콜을 사용하면 TPM_CreateWrapKey 명령어의 요청 및 응답 구조도 바꿔야 하고 TPM_Seal 명령어 구조도



[그림 7] SKAP를 이용한 TPM_CreateWrapKey 명령어 인가

바꿔야 하는 것과 같이 SKAP을 사용하는 모든 명령어는 그 명령어 구조 자체를 변경해야 하는 어려움이 있다.

3. 제안하는 프토로콜

상기한 바와 같이 OSAP 프로토콜을 이용하여 명령어 인가를 수행할 경우 내부자에 의해 다른 사람의 비밀 정보가 노출되는 취약점이 있었다. 이를 방어하는 방법으로 SKAP 프로토콜이 있었지만 이 경우는 모든 명령어 체계 자체를 다시 구성해야 하는 단점이 있었다. 본 논문에서는 이러한 비효율적인 면을 개선한 새로운 내부자 공격 방어용 프로토콜을 제시하고자 한다. 제안하는 제안 프로토콜의 전체 요도는 그림 8에 나타내었다.

이 프로토콜은 사용자와 TPM이 공유하고 있는 Shared secret을 다른 사용자들도 계산해 낼 수 있다는 OSAP의 취약점에 대응하기 위해 사용자가 UK(User Key)를 랜덤하게 생성하여 TPM에게 전송하는 방법을 사용하였다.

기존의 TPM 칩과의 호환성을 최대한 유지하기 위해 그림 8의 단계 (1)과 같이 정당한 사용자(User3)가 UK(UserKey)를 랜덤하게 선택하여 부모키의 공개 키로 암호화한 후 기존 TPM_OSAP 프로토콜에 추가하여 TPM으로 전송하게 된다. 이렇게 함으로써 TPM 칩은

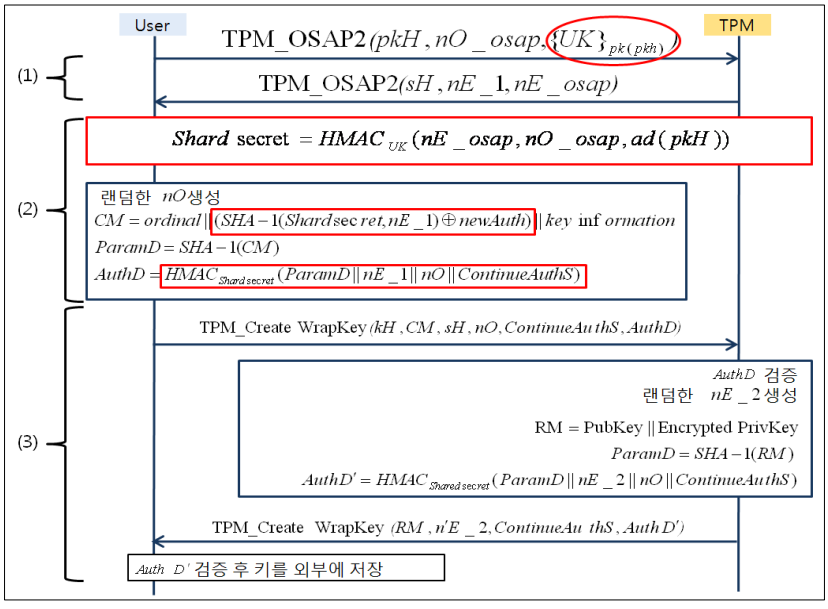
UK 값을 복호화할 수 있고 그림 8 단계 (2)와 같이 UK를 HMAC 키로 하여 세션을 위한 공유 정보 Shared secret을 생성한다. 이후의 명령어들은 기존의 명령어 체계를 그대로 유지하면서 공유 정보 Shared secret을 이용하여 명령어를 구성한다. 따라서 제안하는 프로토콜 역시 OSAP과 동일한 방식으로 newAuth에 대한 기밀성을 유지하게 된다. 그러나 다른 사용자들(User1 혹은 User2)은 UK를 알지 못하기 때문에 Shared secret를 생성할 수 없게 된다.

4. 내부자 공격 실험 및 효율성 분석

4.1 내부자 공격 실험

본 논문에서는 실제로 TPM이 장착된 PC상에서 TPM 명령어 인가 프로토콜에 대한 내부자 공격 실험을 수행하여 그 공격 가능성을 확인하고자 한다. 공격 시뮬레이션은 Infineon사의 TPM 칩인 SLB9535가 장착된 PC[11]에서 수행하였다. Windows7 운영체제를 사용하였으며 Visual Studio 2008을 이용하여 개발하였다.

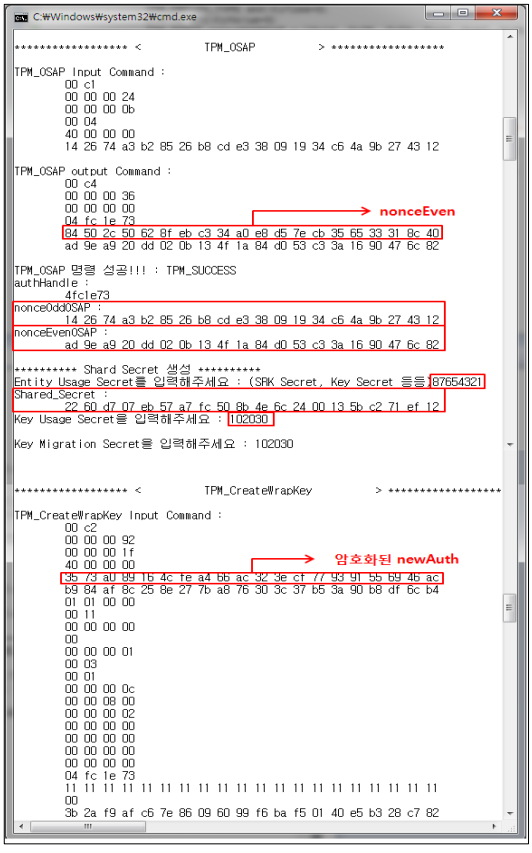
먼저 그림 9는 3.1에서 기술한 일반적인 OSAP프로토콜을 사용하여 새로운 키를 생성하는 TPM_CreateWrapKey 명령어 과정을 나타낸 것이다. TPM_CreateWrapKey 명령어에 대한 인가 과정에서 사용되는 파라미터들을 표 1에서 정리하였다. 표 1에서 정리한 SRK secret, nO_osap, nE_osap을 이용하여 Shared secret을 계산할 수 있고 Shared secret과 nE_1을 이용하여 암호화된 newAuth를 만들어낼 수 있다.



[그림 8] 제안하는 프로토콜을 이용한 TPM_CreateWrapKey 명령어 인가

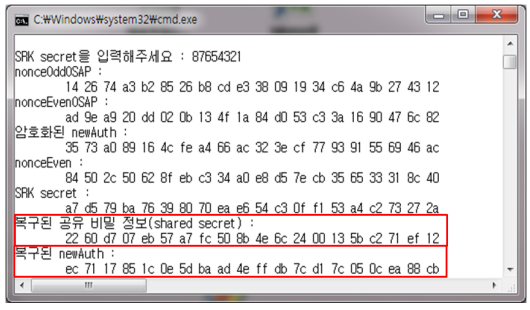
[표 1] TPM_CreateWrapKey 명령어 파라미터

파라미터	값
SRK secret	"87654321"의 SHA1 해쉬 값 : a7 d5 79 ba 76 39 80 70 ea e6 54 c3 0f f1 53 a4 c2 73 27 2a
nO_osap	14 26 74 a3 b2 85 26 b8 cd e3 38 09 19 34 c6 4a 9b 27 43 12
nE_osap	ad 1b 65 1c 99 40 ee c7 5d a9 50 f8 e2 9b d0 e1 b5 8c 5e dd
Shared secret	bc c9 69 bb b6 24 01 01 76 0a d5 d2 b2 16 44 3b 14 8b 14 a3
nE_1	4c dc 20 f4 d2 fe 7b bf c6 e9 4f bb 4b a7 70 9f cf be bc 2e
newAuth	"102030"의 SHA1 해쉬 값 : ec 71 17 85 1c 0e 5d ba ad 4e ff db 7c d1 7c 05 0c ea 88 cb
암호회된 newAuth	52 00 f6 3b 93 a1 ea db 9e c8 61 9b af 1b 4f b9 fe 0d 81 67



[그림 9] 명령어 인가 과정 시뮬레이션

플랫폼 내부의 공격자가 메시지 가로채기를 이용하여 얻을 수 있는 값은 nO_osap, nE_osap, nE_1, 암호화된 newAuth이다. 그리고 SRK secret은 미리 공유되어있기 때문에 공격자가 알고 있다는 것을 가정한다. 가로채기를 통해 얻어낸 4개의 값을 통하여 공격자는 정당한 사용자와 TPM이 공유한 비밀 정보인 Shared secret과 생성할 키를 사용하기 위해 필요한 비밀 정보인 newAuth를 복구할 수 있다. 본 논문에서는 내부 공격자가 비밀 정보인 Shared secret과 newAuth를 복구할 수 있음을 실험을 통해 검증할 수 있었으며 그 결과를 그림 10에 나타내었다.



[그림 10] 내부 공격자의 공격 실험 결과

4.2 효율성 비교

제안하는 프로토콜은 그림 8 단계 (1)과 단계 (2)에서 세션 공유 정보 Shared secret를 생성하는 과정만 기존의 OSAP과 다를 뿐 명령어를 구성하는데 있어서 OSAP과 같은 과정을 수행하기 때문에 기존의 TPM을 구동시키는 어플리케이션에 적용하기 수월하다. 이에 반해 SKAP 프로토콜은 2개의 세션 공유 정보 K_1과 K_2를 사용하기 위하여 그림 7의 단계 (2)와 같이 별도의 명령어 구성이 필요하기 때문에 기존의 TPM을 구동시키는 어플리케이션에 적용하기 어렵다.

SKAP 프로토콜은 newAuth의 기밀성을 유지하기 위해 AES와 같은 대칭키 암호화 알고리즘[12]을 사용한다. 하지만 TPM에서 대칭 키 알고리즘에 대한 지원은 제조사의 옵션 사항이다. 실제로 현재 상용화되어 사용 중인 대부분의 TPM칩에는 대칭 키 암호용 엔진이 장착되어 있지 않다. 따라서 SKAP 프로토콜은 AES와 같은 대칭키 알고리즘을 지원하지 않는 TPM에서는 활용할 수 없다.

하지만 제안한 프로토콜은 OSAP과 같은 방법으로 newAuth에 대한 기밀성을 유지하기 때문에 대칭키 암호화 알고리즘을 지원하지 않는 TPM에서도 적용할 수 있다는 장점을 지니고 있다. 결론적으로 내부자 공격에 대응하는 제안 프로토콜은 OSAP 프로토콜만 최소한으로

변형하여 사용함으로써 확장성과 호환성이 뛰어난 효율적인 프로토콜이다.

명령어 인가 프로토콜을 사용할 경우의 연산량의 증가를 비교한 것이 표 2이다. 표 2의 연산량 비교에서 TMP_OSAP, TPM_CreateWrapKey 명령어의 요청 및 응답 메시지 구성에 관한 부분은 공통적인 내용이므로 비교에서 생략하고 연산량이 달라지는 부분만 요약하였다. OSAP에 비해 SKAP은 연산량면에서도 HMAC 연산과 공개키 암복호 연산 그리고 대칭 키 암복호 연산이 추가적으로 필요하다. 반면, 제안 프로토콜에서는 랜덤 수를 암복호하는 연산만 추가하고도 내부자 공격을 방어할 수 있다. 특히, 정당한 사용자가 선택한 값인 UK는 고정적으로 사용될 수 있기 때문에 UK에 대한 공개키 암호화 연산은 사전에 한 번만 계산되어 저장하고 이후에는 추가적인 계산없이 사용할 수 있다. 즉, UK를 고정적으로 사용하더라도 사용자와 TPM이 만드는 nO_osap과 nE_osap에 의해 Shared secret은 매 세션마다 랜덤하게 생성할 수 있다.

[표 2] 연산량 비교(* : 사전 계산 가능)

구분	OSAP		SKAP		Proposed	
	User	TPM	User	TPM	User	TPM
HMAC	2	2	3	3	2	2
SHA-1	2	1	2	1	2	1
공개키 암호			1*		1*	
공개키 복호				1		1
대칭키 암호			1	1		

논문에서는 표 2에서 사용되는 암호 알고리듬의 연산량을 성능 실험을 통해 비교해 보았다. 이를 정리한 것이 표 3이다. 표에서 연산 시간은 Intel i7 860(2.8GHz) PC에서 100회 실행 후 그 평균 시간을 의미한다. HMAC에서는 SHA-1 함수를 기반하여 수행하며 RSA 암호에서는 표준에서 제시한 2048비트의 키 길이를 사용하였다. 단, TPM의 공개키는 $2^{16}+1$을 사용하도록 하고 있어 복호에 비해 계산시간이 매우 적다. 대칭 키 암호는 AES를 사용하여 실험하였으나 위에서 언급한 바와 같이 현재 TPM 칩에는 AES가 구현되어 있지 않아 칩 내부의 연산 시간은 측정할 수 없었다. 보는 바와 같이 OSAP은 사용자나 TPM 칩 모두 약 60μ s의 시간이 소요되었다. SKAP 대응 방법의 경우는 사용자의 HMAC , RSA 암호화, 대칭 키 암호 연산 시간이 각각 20μ s, 15ms, 2μ s정도 추가되었다. 본 논문에서 제안하는 프로토콜은 대칭 키 암호 방식을 사용하지 않고도 내부자 공격을 방어할 수 있다. 그러나 OSAP에 비해서는 사용자와 TPM 칩에서 공개 키 암·복호 연산 시간이 추가적으로 요구된다.

[표 3] 실험적인 연산시간 비교

구분	OSAP		SKAP		Proposed	
	User	TPM	User	TPM	User	TPM
HMAC	40μ s	40μ s	60μ s	60μ s	40μ s	40μ s
SHA-1	20μ s	10μ s	20μ s	10μ s	20μ s	10μ s
공개키암호 (RSA)			15ms		15ms	
공개키복호 (RSA)				620ms		620ms
대칭키암호 (AES)			2μ s	측정 불가		

5. 결 론

본 논문에서는 멀티유저 환경하에서 사용하는 TPM 칩의 명령어 인가 프로토콜인 OSAP 수행 시 내부자에 의해 다른 사람의 비밀 정보가 노출되는 취약성이 있음을 분석하였다. 이러한 내부자 공격은 TPM 키 계층구조의 최상위 키인 SRK를 공유해야 된다는 취약점에 기초하고 있었다.

따라서 본 논문에서는 내부자 공격 이 실제 TPM칩이 장착된 PC 환경 하에서 가능한지를 시뮬레이션을 통해 검증하였다. 또한, 이러한 내부자 공격을 차단할 수 있는 변형된 OSAP 프로토콜을 제안하였다. 제안 방식은 기존의 대응 방법에 비해 OSAP 명령어 이외에 다른 명령어 체계의 구성이 필요 없다. 또한 대칭키 암호화 알고리즘을 지원하지 않는 TPM에서도 적용 가능하다. 따라서 제안하는 내부자 공격 대응 프로토콜을 이용하면 보다 안전한 신뢰 컴퓨팅 환경 구축에 사용될 수 있을 것이다.

참고문헌

[1] 김영수, 박영수, 박지만, 김무섭, 김영세, 주홍일, 김명은, 김학두, 최수길, 정성익, “신뢰 컴퓨팅과 TCG 동향”, 전자통신동향분석, 제22권, 제1호, pp. 83-96, 2007.

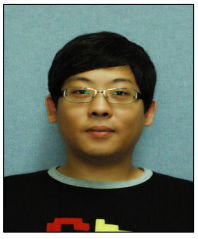
[2] 강동호, 한진희, 이윤경, 조영섭, 한승완, 김정녀, 조현숙, “스마트폰 보안 위협 및 대응 기술”, 전자통신동향분석, 제 25권 3호, 2010.

[3] Trusted Computing Group, "About TCG", Available at http://www.trustedcomputinggroup. org

[4] ISO/IEC 11889-1 : Information technology - Security

techniques - Trusted Platform Module - Part 1: Overview, 2009.
[5] ISO/IEC 11889-2 : Information technology - Security techniques - Trusted Platform Module - Part 2: Design principles, 2009.
[6] ISO/IEC 11889-3 : Information technology - Security techniques - Trusted Platform Module - Part 3: Structures, 2009.
[7] ISO/IEC 11889-4 : Information technology - Security techniques - Trusted Platform Module - Part 4: Command, 2009.
[8] NIST, "Secure Hash Standard", FIPS PUB 180-1, 1994.
[9] L. Chen and M. Ryan, "Attack, solution and verification for shared authorization data in TCG TPM", 6th International Workshop on Formal Aspects in Security and Trust(FAST'09), pp. 201-216, 2009.
[10] Trusted Computing Group, "TCG Software Stack(TSS) Specification Version 1.2 Level 1 Errata A", 2007.
[11] Infineon, "Trusted Platform Module TPM 1.2 SLB 9636 TT 1.2", Available at http://www.infineon.com/tpm
[12] NIST, "Advanced Encryption Standards", FIPS PUB 197, 2001.

오 두 환(Doo-Hwan Oh) **[정회원]**



- 2010년 2월 : 호서대학교 정보보호학과 (공학사)
- 2010년 3월 ~ 현재 : 호서대학교 대학원 정보보호학과(석사과정)

<관심분야>
신뢰 컴퓨팅, 부채널 공격, 네트워크 보안

최 두 식(Doo-Sik Choi) **[정회원]**

- 2010년 2월 : 호서대학교 정보보호학과 (공학사)
- 2010년 3월 ~ 현재 : 호서대학교 대학원 정보보호학과(석사과정)

<관심분야>
신뢰 컴퓨팅, 네트워크 보안, 부채널 공격

김 기 현(Ki-Hyun Kim) **[정회원]**

- 1993년 2월 : 경북대학교 전자공학과(공학사)
- 1995년 2월 : 경북대학교 전자공학과(공학석사)
- 2010년 2월 : 충북대학교 컴퓨터공학과(박사 수료)
- 2009년 7월 ~ 현재 : 에스지에이㈜ R&D부문 총괄사장

<관심분야>
시스템 및 네트워크 보안, 보안관제, 전자문서보안

오 수 현(Soo-Hyun Oh) **[정회원]**

- 1998년 2월 : 성균관대학교 정보공학과 졸업(공학사)
- 2000년 2월 : 성균관대학교 전기전자 및 컴퓨터공학과 석사(공학석사)
- 2003년 8월 : 성균관대학교 전기전자 및 컴퓨터공학과 박사(공학박사)
- 2004년 3월 ~ 현재 : 호서대학교 정보보호학과 교수

<관심분야>
암호학, 네트워크 보안 프로토콜, 정보보호 평가인증

하 재 철(Jae-Cheol Ha) **[종신회원]**

- 1989년 2월 : 경북대학교 전자공학과 (공학사)
- 1993년 8월 : 경북대학교 전자공학과 (공학석사)
- 1998년 2월 : 경북대학교 전자공학과 (공학박사)
- 1998년 3월 ~ 2007년 2월 : 나사렛대학교 정보통신학과 부교수
- 2007년 3월 ~ 현재 : 호서대학교 정보보호학과 부교수

<관심분야>
정보보호, 네트워크 보안, 부채널 공격