최소 전제조건 개념을 이용한 성질 기반 데이터 실링☆

A Property-Based Data Sealing using the Weakest Precondition Concept

박 태 진* 박 준 철**
Tae-Jin Park Jun-Cheol Park

요 약

신뢰 컴퓨팅(Trusted Computing)은 하드웨어를 통해 컴퓨터 사용자가 제어할 수 있는 범위 이상에서 보안성을 제공하고, 컴퓨팅 장치에 대한 신뢰를 보장하는 기술을 말한다. 신뢰 컴퓨팅에서 규정한 TPM(Trusted Platform Module)이라는 전용 보안 칩은 신뢰성 있는 데이터 저장 및 신뢰성 있는 플랫폼 상태 보고를 가능하게 하는 핵심 장치이다.

데이터 실링은 데이터를 암호화할 때 특정한 플랫폼의 상태와 연관시켜 기존의 암호화를 통한 기밀성 보호를 한층 더 강화한 강력한 보안 기법이다. 데이터 실링에는 이진 해쉬 값 기반의 플랫폼 구성 상태를 직접 활용하는 방식과 플랫폼의 제공 성질(property)을 실링에 활용하는 방식이 있다. 본 논문에서는 Dijkstra가 제안한 최소 전제조건 개념을 이용한 새로운 성질 기반의 데이터 실링 및 해제 프로토콜을 제안한다. 제안하는 프로토콜은 실링 할 때와 동일한 성질에서 실링 해제를 가능하게 하여 정상적 시스템 업데이트 시 실링 해제가 가능하도록 한다. 그러면서 기존 성질 기반 실링 방식의 문제점인 제3자 신뢰 기관의 비현실성 및 플랫폼의 프라이버시 노출 문제를 해결하였다. 본 논문은 제안 프로토콜의 보안성을 분석적으로 제시하였으며, Strasser가 개발한 소프트웨어 TPM 에뮬레이터 환경에서 프로토콜을 구현하여 그 동작성을 실험적으로 입증하였다. 제안 기법은 데스크탑 PC는 물론 무선 이동 네트워크 상의 PDA나 스마트 폰 플랫폼에도 적용 가능하다.

Abstract

Trusted Computing is a hardware-based technology that aims to guarantee security for machines beyond their users' control by providing security on computing hardware and software. TPM(Trusted Platform Module), the trusted platform specified by the Trusted Computing Group, acts as the roots for the trusted data storage and the trusted reporting of platform configuration.

Data sealing encrypts secret data with a key and the platform's configuration at the time of encryption. In contrast to the traditional data sealing based on binary hash values of the platform configuration, a new approach called property-based data sealing was recently suggested. In this paper, we propose and analyze a new property-based data sealing protocol using the weakest precondition concept by Dijkstra. The proposed protocol resolves the problem of system updates by allowing sealed data to be unsealed at any configuration providing the required property. It assumes practically implementable trusted third parties only and protects platform's privacy when communicating. We demonstrate the proposed protocol's operability with any TPM chip by implementing and running the protocol on a software TPM emulator by Strasser. The proposed scheme can be deployed in PDAs and smart phones over wireless mobile networks as well as desktop PCs.

☞ Keyword: 보안, TPM(Trusted Platform Module), 데이터 실링, 무선이동 네트워크, 신뢰 컴퓨팅(Trusted Computing)

1. 서 론

최근, 기밀 데이터의 노출이나 바이러스, 웜과 같은 소프트웨어 공격에 의한 시스템 손상과 외부로부터의 위협이 날로 증가되고 있다. 공격자들은 예전에 네트워크나 서버를 주요 공격 대상으로 삼았지만, 여러 보안 기법들이 많이 발달되고 널리 쓰여 짐에 따라 최근에는 공격 대상을 상대적으로 보안이 취약한 클라이언트 측으로 돌리게 되었다. 그 결과 클라이언트 정보의 보안 문제는 갈수록 심각해지고 있다.

* 정 회 원 : 홍익대학교 컴퓨터공학과 석사 졸업
tjpark@mail.hongik.ac.kr
** 정 회 원 : 홍익대학교 컴퓨터공학과 교수
jcpark@hongik.ac.kr(교신저자)
[2008/02/14 투고 - 2008/02/21 심사 - 2008/07/03 심사완료]
☆ 본 논문은 ETRI 정보통신연구개발사업의 위탁연구과제 (7010-2007-011)로 수행한 연구결과임

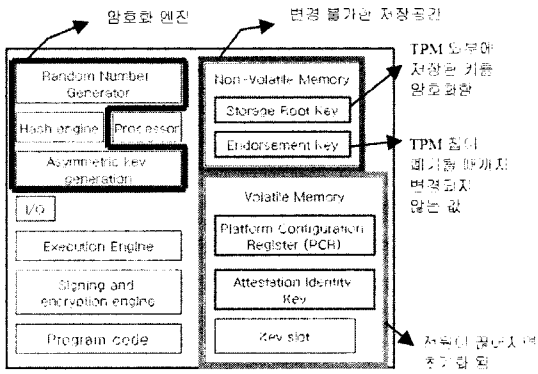
지난 몇 년 동안 소프트웨어를 주로 사용하여 클라이언트에 대한 정보보안을 제공하려고 노력 하였지만 실효를 거두지 못하고 있다. 이와 같이 소프트웨어에 의존하던 보안 문제를 하드웨어를 이용해 해결해 보고자 하는 시도에서 신뢰 컴퓨팅(Trusted Computing)이라는 개념이 시작되었다. 신뢰 컴퓨팅[1,2,3,4,11]이란 하드웨어를 이용하여 컴퓨터 사용자가 제어할 수 있는 범위 이상에서 보안성을 제공하고, 하드웨어와 소프트웨어로 구성되어 있는 컴퓨팅 장치에 대한 신뢰를 보장하는 것을 말한다. 예를 들어, DVD에 저장되어 있는 콘텐츠를 보호하기 위해 DVD의 불법적인 복제를 막고 싶다고 하자. 현재의 PC에서는 DVD의 데이터라고 할 수 있는 영상과 음성은 PC를 구성하고 있는 하드웨어와 소프트웨어를 거쳐 사용자에게 전달이 된다. 그러므로 DVD 데이터에 접근하는 모든 하드웨어와 소프트웨어에 대한 보안을 강화하는 것만이 복제를 막는 길이 된다. 여기에는 DVD에 직접 접근할 수 있는 DVD ROM과 운영체제, DVD에 담겨진 데이터를 재생하기 위한 소프트웨어, 그래픽 카드, 심지어 모니터까지도 포함될 것이다. 이러한 하드웨어와 소프트웨어의 구성 요소 중 하나라도 불법 복제 방지를 고려하지 않았다면 그 DVD에 담긴 데이터는 불법 복제의 대상이 될 수 있다. 이러한 문제점을 해결하기 위한 방법 중 하나는 정해진 것 이외의 어떠한 프로그램이나 장치도 사용될 수 없는 닫힌(closed) 플랫폼을 강제하는 것이다. 닫힌 플랫폼을 따르는 시스템에서는 플랫폼을 구성하는 하드웨어와 소프트웨어를 엄격하게 제어할 수 있다. 운영체제를 변경하거나, 허가되지 않은 응용 프로그램이 실행되는 것이 불가능에 가깝다면, 데이터와 작업 처리에 대한 무결성을 보장하기가 훨씬 쉽다. 하지만 닫힌 플랫폼은 PC와 같은 열린(open) 플랫폼에 비해 유연성이 많이 떨어지고 그 특성상 PC 시장을 완전히 대체할 수 없을 것으로 판단된다. 신뢰 컴퓨팅에서는 열린 플랫폼에서도 컴퓨팅 장치의 신뢰성을 보장할 수 있는 방법을 제공하려고 하며, 이를 위해서 TPM(Trusted Platform Module)[1]이라고 하는 특수 목적의 하드웨어 칩(chip)을 이용한다. TPM은 다양한 형태의 플랫폼에 탑재되어 강력한 사용자 인증과 하드웨어 검증(attestation) 방법을 제공해 준다.

TPM이 제공하는 중요한 기능 중 하나가 데이터 실링(sealing) 기능이다. 실링은 특정 플랫폼에서의 데이터 기밀성을 암호화를 이용해 보호하는 것이다. 플랫폼의 데이터 기밀성을 제공해 주는 데이터 실링은 멀티미디어 콘텐츠 보호 등 그 활용도가 넓어서 필요성이 더욱 커지고 있다. 데이터 실링에는 이진 해쉬 값을 이용 또는 플랫폼의 제공 성질(property)을 이용하는 두 가지 방식이 적용될 수 있다. 본 논문에서는 제공 성질을 이용한 새로운 실링 프로토콜을 제안하고, 이를 TPM 에뮬레이터[10] 환경에서 구현하여 그 동작성을 입증한다. 제안하는 프로토콜은 실링 할 때와 동일한 성질에서 실링 해제가 가능 하기 때문에 빈번히 발생하는 소프트웨어 패치 등의 시스템 업데이트 상황에서 실링된 데이터를 활용할 수 있게 한다. 제안 프로토콜은 기존의 성질 기반 방식과 달리 인증 과정에 참여하는 TTP(Trusted Third Party)의 부담이 최소화 되었다. 또한 인증 과정에서 플랫폼의 구성 상태 전체를 제조업자에게 노출 시키지 않아 사용자의 프라이버시를 보장하였다.

본 논문의 구성은 다음과 같다. 제 2장에서는 본 논문에서 제안하는 데이터 실링 기법과 관련 된 배경 지식과 관련 연구에 대해 서술한다. 제 3장에서는 최소 전제조건 개념[9]을 이용한 새로운 성질 기반의 데이터 실링을 제안하고, 그 특징을 설명한다. 제 4장에서는 제안한 방법을 에뮬레이터 환경에서 구현한 결과와 그 동작성을 평가한다. 제 5장에서는 결론을 맺고 향후 연구 방향을 서술한다.

2. 배경 및 관련 연구

매우 강력한 암호화 기법을 사용한다 할지라도 암호화에 사용된 주요 정보가 메인 메모리나 하드디스크에 저장되어 있다면 공격에 취약하게 된다.



(그림 1) TPM 구성 요소[1]

그러나 모든 연산과 암호 관련 정보가 침입자의 접근이 불가능한 칩에 저장되어 있다면 안전하게 보호할 수 있다. 이러한 기능을 제공하는 것이 TPM 칩이며 신뢰 컴퓨팅 기술 개발 및 정의를 위한 비영리단체 TCG(Trusted Computing Group)에서 규정한 명세의 핵심 부분이다. TPM은 (그림 1)과 같이 랜덤 넘버 생성기, 비휘발성 메모리, 키 생성 알고리즘, RSA와 같은 암호화 기능, 해쉬(SHA-1) 기능 등을 제공한다.

TPM 칩은 이러한 기능들을 이용하여 신뢰할 수 있는 데이터 저장 및 신뢰할 수 있는 플랫폼 상태 보고(reporting)를 가능하게 한다.

플랫폼 상태를 표현하는데 사용되는 값은 PCR(Platform Configuration Register)이라 부르는 레지스터에 저장된 값이다. 이 PCR 값의 갱신은 PCR 확장(extend) 연산에 의해 이루어진다. PCR 확장은 PCR 값과 플랫폼의 특정 소프트웨어나 하드웨어 모듈을 측정한 SHA-1 해쉬 값을 연결(concatenation, '||'로 표현함)한 후 다시 SHA-1 해쉬하는 과정으로 이루어진다. 이 때 얻어진 결과가 다음과 같이 새 PCR 값으로 저장된다. $PCR_i := SHA\text{-}1(PCR_i \| 측정값)$. TPM에서 측정하는 대상 entity는 실행 파일, 설정 파일, 데이터 파일 등이 될 수 있다. 측정은 A entity에서 B entity를 측정한다고 할 때, A가 B의 측정 결과인 "fingerprint"를 계산한 후 이 값을 3장에서 설명할 Stored Measurement Log(SML)라는 자료구조에 저장하고, 다시 이 fingerprint를 PCR에 확장 연산을 통해 저장한다. 그 후에 A는 B로 제어권을 넘긴다. 이런 과정이 신뢰할 수 있는 BIOS의 부트(boot) 블록 코드 부분에서 시작되어, BIOS, 부트로더 (bootloader), OS를 거쳐 응용 프로그램들까지 확장됨으로써, 신뢰성 있는 상태 측정의 체인(chain)이 만들어진다.

TCG 명세에서 정의한 실링 기법에 따르면, 특정 플랫폼에서 실링 된 데이터를 사용하기 위해서는 현재 이 데이터를 쓰려고 하는 플랫폼이 데이터를 실링 했을 때의 플랫폼과 동일해야 한다. 만약 동일하다는 것이 입증되지 않으면 실링 된 데이터를 사용하지 못하기 때문에 허락 받지 못한 플랫폼에서 데이터가 이용되는 것을 방지할 수 있다. 입증 방법은 데이터를 실링 할 때 사용되었던 PCR 값과 현재 플랫폼의 PCR 값이 동일함을 보이는 것이다.

TCG 명세에 따르면 보안 칩 TPM 의 PCR 레지스터들이 플랫폼의 상태를 안전하게 저장할 수 있음을 보장한다. 이와 같이 PCR 값에 의존하는 실링 방식은 신뢰할 수 없는 소프트웨어나 시스템에 대한 공격으로 시스템 상태가 변경되었을 경우에는 적절하게 대처할 수 있다. 하지만 주기적으로 발생하는 소프트웨어 업데이트나 보안 정책에 따른 소프트웨어 패치를 적용시키는 데에는 큰 장애요인이 된다. 소프트웨어의 업데이트와 보안 패치는 플랫폼의 보안 수준을 이전 상태와 동일하게 유지하거나 오히려 더욱 강화하게 된다. 하지만 현재 TCG 명세에 정의되어 있는 방식은 어떠한 이유에서든 PCR 값이 변경되었다면 기존의 실링 된 데이터에 대한 접근을 불가능하게 만든다[5].

이러한 시스템 업데이트 문제를 해결하기 위해 성질(property) 기반의 접근 방식이 제안되었다 [5,6,7,8]. 플랫폼의 성질은 보안과 관련된 요구사항과 같이 플랫폼에서 그 만족여부가 판단 가능한 특성이나 기능, 조건 등을 의미한다. 성질 기반의 검증은 특정 성질을 이용해서 데이터를 실링 하는 것이다. 성질 기반 방식에서는 데이터를 실링할 때 플랫폼의 PCR 값 대신에 특정 성질을 지정하기만

하면 구체적인 사용자 플랫폼의 구성 상태 변화에 관계없이 요구되는 성질의 만족 여부에 따라 실링된 데이터의 이용이 가능할 수 있다. 기존의 성질 기반 데이터 실링 및 원격 인증 기법들은 어떤 플랫폼의 현재 상태가 원하는 성질을 만족함을 신뢰성 있는 제3자 기관에 의뢰하여, 이 기관에서 플랫폼의 이진 상태와 성질을 연결시키도록 하고 있다. 즉 제3자 기관이 모든 플랫폼의 모든 이진 상태에 대해 주어진 성질이 만족되는지 판단할 것을 요구한다. 따라서 플랫폼에 설치될 수 있는 다양한 제조사의 수많은 제품들이 해당 성질에 어떤 영향을 미치는지 모두 파악해야 하기 때문에 현실성이 적으며, 실현 가능하다 하더라도 고려할 제품의 수가 늘어나면서 확장성의 문제를 가진다. 또한 판단 과정에서 플랫폼의 프라이버시가 검증 기관에 노출된다는 단점을 가진다.

3. 최소 전제조건 개념을 이용한 성질 기반의 데이터 실링

본 논문에서는 이러한 기존의 성질 기반 데이터 실링 및 원격 검증 방식의 한계를 극복한 새로운 방식을 최소 전제조건(weakest precondition)의 개념[9]을 활용하여 제안한다. 제안하는 성질 기반 데이터 실링 및 해제 기법의 특징은 다음과 같다. 첫째 특징으로 플랫폼의 프라이버시 보장을 들 수 있고, 나머지 세 특징은 제안 프로토콜의 실행을 위한 오버헤드가 크지 않으며, TCG에서 제정한 규격을 준수하는 어떤 TPM 칩을 통해서도 안전하게 제안 프로토콜을 구현할 수 있음을 의미한다. 이들 특징은 모두 기존의 성질 기반 플랫폼 검증 및 실링 방식의 공통적 한계를 극복하려는 시도에서 도출되었으며, 제안하는 방식의 다양한 플랫폼에서의 구현 가능성을 높인다는 측면에서 그 의미가 있다.

- 실링 해제 과정 동안에 플랫폼의 프라이버시를 최대한 보장
- TTP의 부담을 최소화하여 실현 가능성을 최대로 높임
- 실링 해제 과정에서 플랫폼의 현재 상태 정보의 무결성 보장
- 실링 및 실링 해제를 위한 TPM 명령어의 구현이 현재 생산되는 TPM 칩에서 가능

제안하는 프로토콜은 플랫폼의 구성 상태와 관련된 Stored Measurement Log(SML)를 이용하여 각 제조사에게 반복적으로 최소 전제조건에 해당하는 성질을 질의하는 과정, 부분적 플랫폼 구성 상태와 중간 단계의 전제조건 성질을 연관시키는 과정, 인증서를 통해 실링 된 데이터를 TPM에서 해제하는 과정으로 구성된다.

3.1 SML (Stored Measurement Log)

SML[4]은 시스템 갱신과 관련된 정보를 저장하고 있는 테이블이다. 사용자 시스템의 구성 상태에 변화가 발생하면 해당 업데이트에 관련된 정보가 현재 SML의 탑(top)에 해당하는 인덱스의 엔트리에 저장된다. SML은 PCR 레지스터 값들과 달리 TPM 외부의 적절한 위치(예: 하드 디스크)에 저장되어 쉽게 접근이 가능하다. SML은 시스템 업데이트에 사용된 모든 제품들에 대해 그 업데이트 순서에 따라 스택의 형태로 제품 내역, 제조사 등의 정보를 저장한다. 따라서 주어진 SML 테이블을 이용하여 이 내용이 PCR 값들로 표현되는 특정 구성 상태를 만드는데 사용된 것인지를 판별할 수 있다. SML의 구성 내용은 <표 1>과 같다.

SML 각 엔트리의 구성 요소는 다음과 같다.

- Entity: 시스템의 구성 상태에 변화를 주는 제품(소프트웨어, 하드웨어)의 상세 내용. 제품의 이름과 버전, 기타 이 제품을 유일하게 지칭하는데 필요한 사항들 포함.
- Vendor: 제품을 생산한 제조사에 대한 상세 내용. 제조사 이름, 접속 주소, 암호화 및 서명확인을 위한 공개 키 값들 포함.

- Fingerprint: 시스템에 적용된 해당 제품의 해쉬 값.
- PCR index: 해당 제품의 fingerprint 값이 확장(extend) 된 PCR 레지스터의 번호.
- Pointer to properties: 해당 행까지 설치된 플랫폼 구성 상태에서 만족된다고 신뢰 기관에 의해 입증된 성질들에 대한 포인터.

테이블은 스택의 형태로 확장되며, 아래로부터 0, 1, ... 의 순서로 인덱스가 매겨진다. 인덱스 k는 체크포인트(checkpoint)로 유저가 시스템을 구입할 당시의 구성 상태(예를 들어 OS가설치 된 상태) 정보이다. 구입 이후 설치된 어플리케이션 소프트웨어에 해당하는 정보는 k+1부터 채워진다고 가정한다. m은 현재 SML의 탑에 해당하는, 마지막으로 업데이트 된 제품의 SML 인덱스를 의미한다. SML 인덱스 k+1에 해당하는 제품의 PCR index(레지스터 번호)는 체크포인트에 해당하는 SML 인덱스 k 제품의 PCR index와 달라서 새로운 PCR 레지스터에 그 fingerprint 값이 확장된다고 가정한다.

3.2 데이터 실링/해제(Unsealing) 프로토콜

실링 된 데이터를 가진 플랫폼에서 이 데이터를 실링 해제하기 위한 프로토콜을 제안한다. 플랫폼이 가지고 있는 데이터는 해당 플랫폼의 TPM을 통해 특정 목적 성질을 만족하는 플랫폼의 구성 상태에서 해제될 수 있도록 실링 되었다고 가정한다. 이러한 실링은 기존의 TPM_Seal[1]명령어에서 사용하는 특정 플랫폼 구성 상태(PCR 값들로 표현) 대신에 특정 성질을 제공하는 임의의 플랫폼을 사용하도록 수정하면 달성할 수 있다.

Dijkstra가 제안한 최소 전제조건[9]은 문장 S와 조건 Q가 있을 때, S를 실행시켜 종료된 후 Q가 만족되려면 어떤 초기 상태에서 실행을 시작해야 하는지를 규정하는 개념으로 wp(S,Q)로 표기한다. 이 때의 초기 시작 상태는 최소의 요구 사항을 포함하기 때문에 "최소"라고 불린다. 따라서 이 개념을 플랫폼 구성요소를 설치하는 것에 적용하면, 최종적으로 요구되는 성질을 만족시키기 위해서 설치 이전에 어떤 조건이 만족되어야 하는지를 표현할 수 있다. 제안 프로토콜의 핵심 아이디어는 인증하고자 하는 성질의 범위를 확대하는 과정을 반복하여, 그 과정의 어떤 플랫폼 구성 상태에서 확대된 성질이 만족됨을 보이는 것이다.

(표 1) SML(Stored Measurement Log)

m	Entity	Vendor	Fingerprint	PCR index	Pointer to properties
	....	....	....	....	.....
k	Entity	Vendor	Fingerprint	PCR index	Pointer to properties
	....	....	....	....	.....
1	Entity	Vendor	Fingerprint	PCR index	Pointer to properties
0	Entity	Vendor	Fingerprint	PCR index	Pointer to properties

플랫폼 구성 상태 S_m이 P_m 성질을 제공하는가?

$wp(U_m, P_m) = P_{m-1}$

$wp(U_{m-1}, P_{m-1}) = P_{m-2}$

...

$wp(U_{k+1}, P_{k+1}) = P_k$

플랫폼 구성 상태 S_k가 성질 P_k를 제공

⇒ 플랫폼 상태 S_m이 성질 P_m을 제공함.

S_m: 현재 플랫폼 구성 상태(PCR 값들 표현).

U_i: 플랫폼에 설치되어 구성 상태를 갱신하는 제품. U_i 까지가 적용된 후의 상태가 S_i임.

P_i: 성질.

$wp(U_i, P_i)$: U_i를 적용한 후 P_i가 되기 위한 최소(가장 넓은)의 전제조건 성질.

(그림 2) 최소 전제조건을 활용한 성질 입증 절차

어떤 플랫폼이 특정 성질에 연관되어 실링 된 데이터를 실링 해제 하는 프로토콜은 다음과 같다. 프로토콜의 참여자들은 실링 된 데이터를 해제하고자 하는 플랫폼 및 여기에 포함된 TPM, 플랫폼에 장착된 제품의 제조사들, 플랫폼의 상태가 목적 성질을 만족함을 보증하는 verifier이다. 플랫폼은 제

조사들을 통해 최소 전제조건을 확대해 나가고, 최종적으로 verifier에게 자신이 수집한 정보를 보내면서 검증을 요구한다. 이에 verifier는 수집 정보를 확인한 후 목적 성질이 특정 플랫폼에서 만족됨을 확인한다. 이 제안 프로토콜을 플랫폼에서의 실행 과정을 중심으로 설명한다.

(1) $P \leftarrow P_{target}$ // P_{target} : 입증하려는 성질
(2) for i=m **down to** k+1 **do**
{
(3) if (SML[i].properties includes P) **then**
{ A←(SML[i].properties, P);
goto **done** }
(4) send E_{PU_i}(P, SML[i].entity, KEY_i) **to** SML[i].vendor;
//PU_i is the public key(encryption) of
// SML[i].vendor
// KEY_i is a symmetric key for
// encrypting its response
(5) receive E_{KEY_i}(signed(P, SML[i].entity, digest(SML[i].entity),wp(SML[i].entity, P))) **from** SML[i].vendor;
(6) decrypt with KEY_i and verify the answer using the vendor's public key(verification);
(7) $R_i \leftarrow$signed(P, SML[i].entity, digest (SML[i].entity), wp(SML[i].entity, P));
(8) P←wp(SML[i].entity, P);
}
(9) if (SML[k].properties includes P) **then**
B←(SML[k].properties, P);
else {
(10) retrieve the masked set PCR_{LH}={PCR_0, ····, $PCR_{SML[k].PCRindex}$, Ø, ····, Ø} from the TPM;
(11) send E_{PU_k} (P, PCR_{LH}, SML[0, ····, k], KEY_k) **to** SML[k].vendor;
(12) receive E_{KEY_k}(signed(P, $PCR_{summary(LH)}$, Yes/No, hash of query)) **from** SML[k].vender;
(13) decrypt with KEY_k and verify the answer using the vender's public key(verification);
(14) if(Yes) **then** B←signed(P, $PCR_{summary(LH)}$, Yes, hash of query);
}
(15) done :
send E_{PU_v} ($R_{i(j=m, \cdots, i+1)}$, A or B, SML[k+1, ····, m], KEY_v) **to** verifier;

(16) receive E_{KEY_v}(signed($PCR_{summary(FULL)}$, P_{target}, Yes/No, hash of query)) **from** the verifier;
(17) decrypt with KEY_v and verify the answer using the verifier's public key(verification);
(18) retrieve {PCR_0, ····, PCR_n} from the TPM and check if the configuration matches to the $PCR_{summary(FULL)}$
(19) unseal the sealed data within the TPM using the certified fact that {PCR_0, ····, PCR_n} provides P_{target}

(그림 3) 데이터 실링 해제 프로토콜

- 프로토콜 명세의 정당성 설명

(2) ~ (8)까지는 SML에 입증하려는 성질이 이미 증명되었는지 확인하고, 아니라면 제조사들에게 최소 전제조건을 묻는 과정이다. 이 과정은 인덱스 m부터 시작하여 아래로 최대 k+1까지 진행된다.

(4) 최소 전제조건을 알아내기 위해 입증하려는 성질 값과 SML[i]의 entity 값을 해당 제조사에게 보낼 때 응답 메시지를 암호화할 때 사용하도록 랜덤하게 생성한 비밀키 값을 포함한다. 또한 메시지의 기밀성을 보장하기 위해 해당 제조사의 암호화 공개키로 메시지를 암호화하여 보낸다.

(5) (4)에서 플랫폼이 보낸 질의를 받은 제조사는 응답 최소 전제조건을 생성하고 제조사의 개인키로 서명한 후, 질의에 포함된 비밀키로 다시 암호화된 내용을 플랫폼에 보낸다. 서명 시에 제조사가 최소 전제조건을 계산할 때 사용한 entity에 해쉬함수를 적용한 결과 digest 값을 포함시킨다. 이렇게 함으로써 추후 verifier가 이 digest 값을 SML의 해당 엔트리에 포함된 fingerprint 값과 비교하여 다른 제품을 질의에 쓴 경우 그 사실을 감지할 수 있다.

(9) ~ (14)는 SML[k+1]에 이르기까지 (2)~(8)과정을 거친 후에도 입증하려는 성질을 찾지 못하면 실행된다.

(10),(11) SML[k] entity의 제조사에 질의할 때 SML의 인덱스 k 이후에 설치 된 제품 정보를 숨기기 위해 마스킹된 구성 상태 PCR_{LH}={PCR_0, ····, $PCR_{SML[k].PCRindex}$, Ø, ····, Ø}를 사용한다. 이 질의는 수신자인 제조사의 암호화 공개키로 암호화한

후 전송한다. 이 제조사는 우선 SML[0,···,k](인덱스 0부터 k에 해당하는 SML 값들)의 내용을 가지고 PCR_{LH} 를 재생할 수 있는지 판단한다. 이 과정을 통과하면, 제조사는 수신 SML 내용을 이용하여 플랫폼의 구성 상태 PCR_{LH} 가 성질 P를 만족하는지의 여부를 자체적으로 판단, 결정한다.

(12) 플랫폼은 SML[k] entity의 제조사가 플랫폼의 구성 상태 PCR_{LH} 가 성질 P 를 만족하는지의 여부를 제조사의 개인키로 서명하고, 이후 질의에 포함된 비밀키로 암호화한 내용을 수신한다. 응답 메시지에는 PCR_{LH} 대신에 $PCR_{summary(LH)} = h(PCR_{SML[k].PCRindex} \| h(PCR_{SML[k].PCRindex-1} \| \cdots h(PCR_1 \| h(PCR_0)) \cdots))$이 포함된다. 또한 이 메시지에는 플랫폼의 질의 내용의 해쉬 값이 포함되어 있어 타 플랫폼의 상태 또는 타 성질에 대해 확인을 받는 것을 방지한다.

(15) 플랫폼은 지금까지의 과정에서 얻은 값들을 전송하여 verifier에게 검증을 요구한다. 응답 메시지의 암호화를 위한 비밀키가 내용에 포함되며, 내용 전체는 verifier의 암호화 공개키로 암호화되어 기밀성을 보장한다.

(16) 수신한 메시지의 서명을 verifier가 확인한다. 이후 R_j 값들의 서명을 확인한 후, SML[k+1, ····, m]를 이용하여 최소 전제 조건이 SML 엔트리의 entity 및 그 fingerprint와 일치하면서 m 부터 k+1의 순서대로 계산되었는지 검사한다. 그리고 R_j 에 포함된 최소 전제 조건이 $PCR_{summary(LH)}$ 에 의해 제공됨을 제조사가 입증했음을 확인한다. 이들 과정이 모두 확인되면 verifier는 $PCR_{summary(LH)}$ 과 SML[k+1, ····, m]을 이용하여 $PCR_{summary(FULL)} = h(PCR_n \| \cdots \| h(PCR_{SML[k+1].PCRindex} \| PCR_{summary(LH)}) \cdots)$ 을 생성한 후, 해당 성질이 $PCR_{summary(FULL)}$ 로 요약되는 플랫폼 PCR 값들에서 제공됨을 인증하는 서명 메시지를 제작한다. 서명된 메시지는 질의에 포함된 비밀키에 의해 암호화된 후 전송된다.

기존의 TPM_Unseal[1] 명령어에서는 플랫폼의 지정된 특정 PCR 상태를 입증해야 했지만, 여기서는 특정 성질을 만족하는 임의의 PCR 상태를 지정해서 실링되었기 때문에 어떤 플랫폼 상태에서도 해당 성질이 만족되기만 하면 실링의 해제가 가능하다. 이 프로토콜에서 플랫폼의 검증 요구에 대해 verifier는 수집된 정보를 확인한 후 목적 정보가 특정한 PCR 요약 값을 가지는 플랫폼에서 만족됨을 인증하고 이를 서명한다. 따라서 제안 프로토콜은 데이터 실링 및 해제뿐 아니라 성질 기반의 원격 검증(attestation)을 위한 용도로도 활용될 수 있다.

3.3 보안성 분석

이 절에서는 제안 프로토콜이 인터넷 환경에서 실행될 때 가해질 수 있는 다양한 공격에 대한 적절한 방어 또는 감지 기법을 제공함을 보인다. 플랫폼을 제외한 모든 참여자의 공개키는 CA 등의 신뢰성 있는 기관을 통해 다른 참여자들이 이미 정확하게 알고 있다고 가정한다.

1. 도청을 통한 메시지 내용 획득 시도 및 이를 통한 플랫폼의 프라이버시 노출 시도

플랫폼이 제조사에게 보내는 질의는 수신자의 공개키로 암호화시켜서 보내며, 제조사의 응답은 질의에 포함된 일회용 랜덤 비밀키로 암호화시켜서 보냄으로써 메시지의 기밀성을 보장한다. 이후 플랫폼과 체크포인트에 해당하는 제조사와의 메시지 교환 및 플랫폼과 verifier 사이의 메시지 교환도 같은 방법으로 메시지의 기밀성을 제공한다. 따라서 메시지를 도청한 공격자는 제조사의 복호화 개인키 및 암호화된 내용에 포함된 비밀키를 알지 못하기 때문에 메시지의 내용을 해독할 수 없다.

2. 메시지 변경 및 훼손 공격

제조사나 verifier가 보내는 응답 메시지에 질의 내용이 그대로, 또는 해쉬 된 채 포함되어 있기 때문에 질의 내용의 변경 시 해쉬의 일방향성에 의해 그 변경 사실을 질의자가 감지할 수 있다. 또한 응답 메시지의 변경은 응답 내용에 대해 제조사나

verifier가 자신의 개인키로 서명한 부분이 함께 전달되기 때문에 역시 질의자의 검증 과정에서 발견이 된다. 따라서 모든 프로토콜 참여자가 공인 기관의 인증서를 통해 통신 상대방의 공개키를 가지고 있는 경우 어떠한 변경이나 훼손도 감지된다.

3. 프로토콜 참여자에 의한 플랫폼의 개인 정보 노출

체크포인트 이전의 최소 전제조건을 구하는 과정에서 플랫폼은 제조사들에게 질의를 통하여 원하는 성질 및 제품명을 제공한다. 이를 통해 각 제조사는 질의한 플랫폼에 설치된 제품 하나와 이 제품이 설치된 이후의 최소 요구 성질 정보를 알 수 있다. 제품 정보는 목적 성질을 제공하는지 여부를 판단하는 과정에서 관련 제조사에게 제공해야 하는 최소의 정보라 볼 수 있다. 각 제조사에게는 질의를 통해 서로 다른 한 제품에 대한 정보만 제공되기 때문에 플랫폼의 전체 구성 상태 노출을 피할 수 있다. 체크포인트에 해당하는 제조사에게는 중간 단계 최소 전제조건을 질의하는 과정에서 플랫폼의 부팅 이후 체크포인트에 해당하는 제품 설치까지의 정보가 SML을 통해 전달된다. 하지만 이 역시 체크포인트 이후에 설치된 제품들의 정보가 빠져 있기 때문에 플랫폼의 전체 구성은 노출되지 않는다.

질의 과정에 포함된 목적 성질로부터 플랫폼에 설치된 제품 정보를 유추하는 것도 가능하다. 목적 성질이 특정 제품에서만 제공되는 특별한 것인 경우 특히 그러하다. 하지만, 목적 성질이 매우 구체적이며 특정 제품을 가정하고 있는 경우는 어떤 방법을 쓰더라도 플랫폼의 제품 설치 상황에 대해 완전한 기밀성을 보장하는 것이 불가능하다. 또한 연속된 질의 과정에서 각 제조사는 최소 전제조건에 해당하는 성질을 응답으로 제공하기 때문에, 이 응답의 특성으로 인해 이후의 질의에서는 모호함이 계속해서 추가된 성질들을 사용하게 된다. 따라서 질의응답에 참여하는 제조사들은 점점 더 초기의 목적 성질을 추출해내기 어려워진다. 그러므로 질의에 포함된 성질로부터 플랫폼의 완전한 구성 상태를 유추하는 것은 매우 어렵다고 볼 수 있다.

최종 단계로 플랫폼이 verifier에게 자신이 수집한 내용들을 보낼 때 체크포인트까지의 PCR 값들의 요약인 $PCR_{summary(LH)}$ 및 SML[k+1, …, m]이 포함된다. Verifier는 SML 정보를 통해서 체크포인트 이후에 설치된 제품들은 알 수 있지만, 체크포인트까지는 PCR 값들의 요약 형태밖에 모르기 때문에 해쉬의 일방향성에 의해 일부 PCR 레지스터들의 내용 파악은 불가능하다. 따라서 verifier 역시 플랫폼의 일부 구성 상태밖에 알 수 없다.

4. 플랫폼이 거짓 정보를 이용하여 인증 시도

TCG 명세에 따르면, 플랫폼이 자신의 TPM을 통해 거짓된 PCR 레지스터 값을 현재 구성 상태인 것처럼 외부에 속이는 것은 불가능하다.

플랫폼이 체크포인트 이전에 각 제조사에 보내는 정보는 SML으로부터 얻은 제품 정보 및 인증하고자 하는 성질이기 때문에, 플랫폼이 원하는 경우 이를 속이는 것이 가능하다. 즉, 현재 설치되어 있지 않은 제품을 포함시키거나, 설치되어 있는 제품을 변경 또는 고의로 누락시키는 것이 가능하다. 각 제조사는 최소 전제조건에 해당하는 성질을 응답하면서 사용한 제품의 digest 값을 포함시켜 서명한 메시지를 반환하게 된다. 예를 들어, 플랫폼이 SML에서 어떤 entity의 fingerprint 값은 그대로 둔 채 그 entity를 다른 entity인 것처럼 위장해서 제조사에게 플랫폼에 없는 entity에 대해 질의를 한다고 하자. 이 경우 질의의 응답에 포함된 digest와 SML 해당 제품의 fingerprint 사이의 불일치가 발생하기 때문에 쉽게 발견이 된다. 만약 기존 엔트리의 바꿔 치기가 아니라 새로운 가짜 엔트리를 SML에 포함시켜 해당 제조사에 질의를 한다면 최종적으로 verifier에서 SML로부터 PCR 값들을 재구성할 때 불일치를 만들기 때문에 역시 발견된다. 설치된 제품의 엔트리를 SML에서 고의로 누락시키는 공격에

대해서도 마찬가지로 SML과 PCR 값들의 불일치를 초래하게 된다. 그리고 플랫폼이 체크포인트에 해당하는 제조사에 중간 단계의 성질 만족 여부를 질의할 때는 TPM을 통해서 레지스터들 일부의 값들이 그대로 포함되기 때문에 PCR 레지스터 값들의 위조는 불가능하다.

플랫폼은 verifier에게 자신이 수집한 내용을 통해 최종적으로 플랫폼의 현재 상태가 목적 성질을 만족함을 인증 받는다. 이 때 SML[k+1, ..., m]의 값을 전달하는데 이 내용이 위조될 수 있다. 이 경우 위에서 설명한 대로 제조사의 서명된 응답, SML 내용, TPM의 PCR 레지스터 값들 사이의 불일치가 발생해 발각된다. 제조사가 서명한 메시지를 변경하는 경우는 해당 서명을 검증할 때 그 변조 사실이 드러나게 된다. 플랫폼이 verifier에게 받은 서명된 메시지를 통해 실링 된 데이터를 해제할 때, TPM의 현재 PCR 값들을 요약한 결과가 verifier의 서명 내용에 포함된 $PCR_{summary(FULL)}$ 과 일치하는지 확인한다. $PCR_{summary(FULL)}$ 의 계산과정에서 전체 PCR 레지스터 값이 모두 사용되고, 또 그 적용 순서가 최종 결과에 영향을 주기 때문에 실제의 값들과 한 레지스터라도 차이가 있다면 결과 값이 달라지기 때문에 발각이 된다. 따라서 플랫폼이 처음부터 내용이 바뀌거나, 빠져 있거나, 추가된 가짜 SML을 가지고 프로토콜을 실행했다면, 최종 단계에서 verifier가 서명한 내용에 포함되는 PCR 요약은 실제의 PCR 값들로부터 만든 것과 달라질 것이다. 그 결과 실링 된 데이터의 해제는 불가능하다.

4. 실험

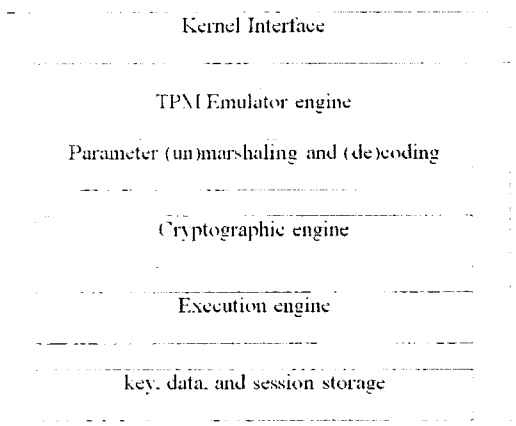
제안 프로토콜을 구현하였으며, 제안 프로토콜이 TCG 명세를 따르고 있는 TPM 에뮬레이터 환경에서 원활히 작동함을 확인함으로써 현재 양산되고 있는 TPM 칩에 적용 가능함을 보였다.

4.1 TPM 에뮬레이터

TPM 에뮬레이터[10]는 2004년에 결성된 TPM 에뮬레이터 프로젝트라는 오픈 소스 프로젝트의 결과물이다. 이 프로젝트의 목적은 소프트웨어 기반의 TPM 에뮬레이터와 TCG 디바이스 드라이버 라이브러리(TCG Device Driver Library: TDDL)의 제작이다. TPM 에뮬레이터는 TPM의 동작을 흉내 내는 소프트웨어이며, TPM을 활용한 프로그램의 작성과 실험을 도와주어 관련 연구를 활성화시키고자 제작되었다. TPM 에뮬레이터는 커널(kernel)모드에서 동작하며 리눅스 커널 2.6.x 이상의 버전에서 작동한다.

TPM 에뮬레이터는 (그림 4)와 같이 커널 인터페이스와 에뮬레이터 엔진 두 부분으로 구성된다. TPM 에뮬레이터엔진은 실제 TPM 칩의 동작을 흉내 내는 부분이며, 커널 인터페이스는 에뮬레이터 모듈을 구동 및 조작하는데 필요한 명령어들의 인터페이스를 제공한다. TPM 에뮬레이터 엔진의 각 부분은 다른 부분들과 서로 밀접하게 연동되어 있다.

TPM 에뮬레이터는 TCG 명세에서 정의 되어 있는 라이브러리(TDDL)와 TPM 명령어들을 제공한다. TPM 제조업자는 칩 제조 시 TDDL과 TPM 장치



(그림 4) TPM 에뮬레이터의 구조[10]

사이의 연결을 정의해야 한다. 대부분의 플랫폼에서 OS가 구동 될 때 TPM 어플리케이션과 TDDL이 실행된다.

TPM 에뮬레이터에서 TPM 명령어와 응답은 (그림 5)와 같이 바이트 문자열로 표현된다.

tag	size	ordinal	req. params.	authorization trailer

| ← 요구명령 헤더 → |

tag	size	result	resp. param.	authorization trailer

| ← 응 답 헤 더 → |

tag: 명령 또는 응답의 종류
size: 전체 명령의 크기(헤더와 트레일러 포함)
ordinal: TCG 명세에 따른 명령 번호
result: 결과 코드 (0은 성공을 뜻하며 다른 숫자는 에러를 뜻함)

(그림 5) TPM 명령 구조[10]

TCG 명세에 정의되어 있는 TDDL 인터페이스 함수는 Tddli_Open( ), Tddli_Close(), Tddli_Cancel( ), Tddli_GetCapability( ), Tddli_SetCapability( ), Tddli_GetStatus( ), Tddli_TransmitData( ) 인데, TPM 에뮬레이터가 제공하는 TDDL에 이 함수들이 모두 구현되어 있다.

4.2 구현 및 가상 실험

본 구현에서는 위 함수 중 Tddli_Open( )과 Tddli_Close( ), Tddli_TransmitData( )를 사용하여 플랫폼과 TPM 사이의 통신을 실현하였다. Tddli_Open( )은 TPM 디바이스 드라이버와의 연결을 설정한다. 그러므로 이 함수는 다른 함수들을 이용해 TPM과 통신하기 이전에 반드시 실행되어야 한다. 이 함수는 TPM 디바이스와의 배타적인 연결을 보장해 줌으로써 통신의 무결성을 제공해 준다. Tddli_Close( )는 설정 된 연결을 종료하는 함수이다. 연결이 종료 된 후엔 TPM 디바이스 드라이버와의 연결에 사용된 모든 자원이 반환된다. Tddli_TransmitData( )는 플랫폼이 TPM에게 보내려고 하는 TPM 명령어들과 그에 대한 응답을 전달해 주는 역할을 한다. 이 함수는 TPM 명령어를 TPM 디바이스 드라이버에 직접적으로 전달해 준다.

본 구현에서 프로토콜 통신에 참여하는 각 노드는 TPM 에뮬레이터[10] 0.5버전이 설치되어 있는 리눅스(Slackware 12.0) 환경에서 C언어로 작성하였다. 성질 값은 임의의 16 바이트 문자열이라 가정하였다.

본 구현에서는 PCR 값들을 TPM으로부터 얻기 위해 TPM_PCRRead 명령어를 사용하여 TPM 에뮬레이터로부터 PCR 값들을 얻어낸다. TPM_PCRRead 명령어는 특정 숫자를 지정하여 원하는 PCR 인덱스의 번호를 명시할 수 있는데, 실험에서는 TPM 명령어 문자열을 Tddli_TransmitData() 함수를 통해 에뮬레이터에 직접 전달하여 PCR 값들을 얻었다.

구현한 프로토콜의 동작성을 보이기 위해 가상의 플랫폼을 가정하였다. 각 제조사 및 verifier 는 소켓 프로그래밍을 이용한 TCP segment를 통해 플랫폼과 서로 메시지를 교환하도록 했다. 이 가상의 플랫폼은 SML 테이블 및 TPM 에뮬레이터를 통해 그대로 흉내내는 TPM 칩을 탑재하고 있다고 가정했다. 프로토콜에서 사용되는 암호화 부분은 널리 사용되는 OpenSSL[12] 라이브러리에 구현된 알고리즘들을 그대로 사용하였다. 이와 같은 실험 환경은 실제로 제안 프로토콜이 구현 및 동작되는 상황과 메시지의 형식 및 각 프로토콜 참여자의 암호화 과정, 수신 메시지의 처리, 송신 메시지의 처리 측면에서 동일하다. 따라서 본 실험을 통해 제안 프로토콜이 실링된 데이터를 해제하며, 그 과정에서 플랫폼의 프라이버시가 보장되고, 제조사들의 운용

부담이 다른 성질 기반 방식들에 비해 현저히 적어진다는 점을 현실적으로 확인할 수 있다. 또한, 본 실험에서 사용된 플랫폼의 구성 상태는 OS 및 다수의 응용 소프트웨어를 탑재한 경우를 상정하고 있어 실제의 PC 등, 가장 일반적인 플랫폼의 상황을 그대로 반영하고 있다.

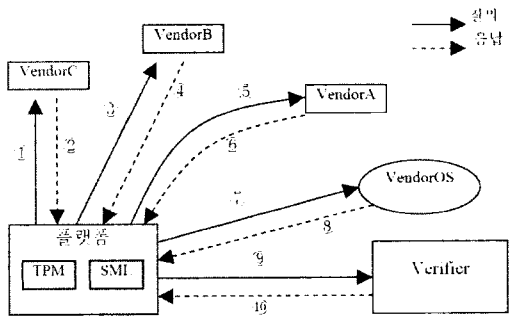
실험에 사용된 플랫폼은 <표 2>와 같은 SML 테이블이 만들어진 상태라고 가정한다. 이 SML은 OS가 설치 된 상태의 플랫폼을 구입한 후 제조사 A, B, C가 제공하는 소프트웨어를 차례대로 설치했음을 보여준다. 체크 포인트 k 값은 0이고 마지막으로 업데이트 된 테이블의 위치인 m 값은 3이다.

(표 2) 가상 플랫폼의 SML

3	EntityC	VendorC	fingerprintC	11	null
2	EntityB	VendorB	fingerprintB	10	null
1	EntityA	VendorA	fingerprintA	9	null
0	EntityOS	VendorOS	fingerprintOS	8	null

아래 (그림 6)은 플랫폼이 위와 같은 SML 및 TMP 칩을 이용하여 실링된 데이터의 해제를 실행하기 위한 과정 및 참여자들을 나타낸다.

중간 단계의 최소 전제조건 성질의 만족 여부를 구분하여 실제로 운용 시 발생할 수 있는 모든 가능한 경우에 대해, 플랫폼은 제조사들, verifier와의 통신 과정을 거쳐서 현재 플랫폼의 상태가 목적 성질을 만족함을 확인할 수 있었다.



(그림 6) 가상 플랫폼의 실링 데이터 해제를 위한 성질 인증 과정

4.3 평가

기존의 성질 기반 데이터 실링 및 원격 검증 방식들은 최소한 하나의 제3자 기관에 플랫폼의 전체 구성 정보를 노출해야 하는 한계를 가진다. 또한 기존 방식들은 플랫폼의 모든 가능한 상태에 대해 어떤 성질이 만족되는지 판단해야 하는 실현 불가능한 요소를 포함하고 있다.

이에 비해 제안 프로토콜은 각 제조사나 verifier의 어떤 기관에도 완전한 플랫폼의 구성 정보는 노출하지 않기 때문에 프라이버시를 최대한 보장한다. 그리고 이 과정에서 각 제조사는 자신이 제작한 제품에 대한 질의만을 처리하기 때문에 전문성과 더불어 실현 가능성을 높였다. 또한 기존 방식들과는 달리 실링 해제 과정에서 플랫폼의 무결성을 보장하는 장치를 갖추어 플랫폼 자체에 의한 공격에 대비한다. 실링 및 해제 과정에서도 모든 연산이 현재 TCG의 규격을 따르는 TPM 칩에서 구현 가능하도록 사용하는 연산의 종류 및 범위를 세밀히 조정하였다.

제안 프로토콜은 실험을 통해 전형적인 플랫폼의 구성 상태에서 장착된 TPM 칩을 통해 목적하는 데이터 실링 및 해제 기능을 보안 특성을 유지하면서 달성할 수 있음을 보였다. 이를 통해, 제안된 프로토콜이 실제로 구현되어 인터넷 환경에서 사용되는 경우에도 그 구체적 적용 환경에 무관하게 해당 기능을 제공함을 신뢰할 수 있다.

5. 결 론

본 논문에서는 최소 전제조건의 개념을 활용하여 성질 기반의 새로운 데이터 실링 및 해제 프로토콜을 제안하였다. 제안 방법을 통해 일상적으로 일어나는 시스템 업데이트 상황에서도 실링 된 데이터를 안전하게 해제할 수 있음을 보였으며, 이 과정에서 기존 성질 기반의 데이터 실링 및 원격 검증 방법들이 공통적으로 가지고 있었던 플랫폼의 전체 구성 상태 노출 및 신뢰 기관의 비현실성 문

제를 해결하였다. 제안 프로토콜은 기존 TCG 명세에 따른 TPM 칩에서 전 과정이 구현 가능하며, 원격 검증 등 타 TPM 제공 기능에도 활용될 수 있다는 장점을 가진다.

실제의 TPM 칩을 흉내 낸 소프트웨어 TPM 에뮬레이터[10]를 사용하여 모의실험을 진행한 결과, 제안 프로토콜이 실링 된 데이터를 해제하는 과정이 원활이 수행됨을 보임으로써 실제의 TPM 칩을 채용한 플랫폼에서도 이 기법이 적용될 수 있음을 확인하였다.

TCG 에서 추진하고 있는 TPM 기반 보안 기술은 서버나 데스크탑 PC 뿐만 아니라 노트북 컴퓨터, PDA 및 휴대폰 등과 같은 무선 이동 네트워크 환경의 플랫폼까지 대상으로 하고 있다. 본 논문의 결과를 무선 이동 네트워크를 고려한 휴대용 단말기에 적용할 수 있도록, 제안 프로토콜 수행 과정에서 이동 플랫폼의 직접 참여 부분을 경량화 및 타 신뢰 기관에 위임하는 방법의 연구가 향후 필요하리라 판단한다. 또한 제조사에서 최소 전제조건 성질을 구한 내용을 미리 안전한 디렉터리 등에 공지하여 제조사에 직접 질의하는 부분의 부담을 경감하는 방안도 추가의 연구가 필요하다고 판단한다.

참 고 문 헌

[1] Trusted Computing Group. "TPM Main Specification," *Version 1.2*, http://www.trustedcomputinggroup.org, November 2003.

[2] P. England, B. Lampson, J. Manferdelli, M.Peinado, B. Willman, "A Trusted Open Platform," *IEEE Computer*, July 2003.

[3] R. L. Kay, "Trusted Computing is Real and it's Here," *Endpoint Technologies Associates* (http://www.ndpta.com) Whitepaper, 2007.

[4] Trusted Computing Group. "TCG Specification Architecture Overview," Revision 1.3, http://www.trustedcomputinggroup.org, March 2007.

[5] U. Kuhn, K. Kursawe, S. Licks, A.-R. Sadeghi, C. Stuble, "Secure Data Management in Trusted Computing," *In Proc. of Cryptographic Hardware and Embedded Systems* (CHES 2005), 2005.

[6] A.-R. Sadeghi, C. Stuble, "Property-based Attestation for Computing Platforms: Caring about properties, not mechanisms," *In the 2004 New Security Paradigms Workshop, VA, USA,* Sept. 2004. ACM SIGSAC, ACM Press.

[7] J. Poritz, M. Schunter, E. Van Herreweghen, M. Waidner, "Property Attestation - Scalable and Privacy-friendly Security Assessment of Peer Computers," *Technical Report RZ 3548, IBM Research,* May 2004.

[8] L. Chen, R. Landfermann, H. Lohr, M. Rohe, A.-R. Sadeghi, C. Stuble, "A Protocol for Property-based Attestation," *In Proc. of ACM Scalable Trusted Computing,* 2006.

[9] E.W. Dijkstra, "Guarded Commands, Nondeterminacy and Formal Derivation of Programs," *Communications of the ACM, Vol. 18, No. 8,*1975, pp. 453-458.

[10] M. Strasser, "Software-based TPM Emulator for Linux," *Department of Computer Science,"Swiss Federal Institute of Technology Zurich,* 2004.

[11] Trusted Computing Group. "TCG Software Stack Specification," *Version 1.1*, http://www.trustedcomputinggroup.org, August 2003.

[12] OpenSSL: The Open Source toolkit for SSL/TLS, http://www.openssl.org

◐ 저 자 소 개 ◑

박 태 진(Tae-Jin Park)
2005년 홍익대학교 컴퓨터공학과 졸업(학사)
2008년 홍익대학교 대학원 컴퓨터공학과 졸업(석사)
관심분야 : 컴퓨터 및 네트워크 보안
E-mail: tjpark@mail.hongik.ac.kr

박 준 철(Jun-Cheol Park)
1986년 서울대학교 계산통계학과(학사)
1988년 KAIST 전산학과(석사)
1998년 University of Maryland, College Park, 전산학과 (박사)
2001년 - 현재: 홍익대학교 컴퓨터공학과 부교수
관심분야 : 클라이언트 보안, P2P 네트워크, 무선 이동 네트워크 플랫폼
E-mail: jcpark@hongik.ac.kr