*Journal of The Korea Institute of Information Security & Cryptology* ISSN 1598-3986(Print)
VOL.29, NO.5, Oct. 2019 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2019.29.5.949

# TPM을 활용한 임베디드 시스템 환경의 보안 부팅 구현*

김 진 우,† 이 상 길, 고 재 용, 이 철 훈‡
충남대학교 컴퓨터공학과

# An Implementation of Secure boot Using TPM in Embedded System*

Jin-Woo Kim,† Sang-Gil Lee, Jae-Yong Ko, Cheol-Hoon Lee‡
Chungnam National University

요 약

최근 임베디드 시스템은 전자기기의 소형화와 IoT(Internet of Things, 사물인터넷)의 발전과 함께 일상 서비스, 의료, 군사, 자율주행 자동차 등 다양한 분야에서 사용되고 있다. 하지만 임베디드 시스템을 위한 기초적인 보안이 미비하여 보안 사고에 대한 잠재적인 위협이 되고 있다. 이에 본 논문에서는 임베디드 시스템 환경의 무결성을 보호하기 위해 TPM(Trusted Platform Module)을 활용한 보안 부팅을 구현하였다. 제안된 설계 방식은 임베디드 시스템에서 요구되는 가용성을 고려하였으며, Boot 과정에서 TPM을 통해 시스템의 변조를 감지한다. 또한 보안 부팅 간 커널의 AES 암호화를 통해 커널에 대한 기밀성을 강화했다.

ABSTRACT

Due to miniaturization of electronic devices and development of IoT(Internet of Things), embedded system have been used in various field. Meanwhile, there is a potential vulnerability by the insufficient of system's security. In this paper, we implement secure boot using TPM to protect the integrity of embedded system environment. The Suggestion considers the required availability in the embedded system and detects the system's tampering at secure boot process via TPM. In addition, we have reinforced the confidentiality through AES encryption of the kernel at secure boot.

**Keywords:** Secure boot, Embedded system, TPM, Integrity

## I. 서 론

오늘날 휴대용 기기의 보급이 증가하고 IoT 기기 또한 널리 이용됨에 따라 이를 제어하는 임베디드 기기 시장 또한 빠른 추세로 발전하고 있다. 2018년 시장 정보 회사 IC-Insights의 자료 조사에 따르면 Fig. 1.과 같이 전 세계적으로 임베디드 시스템에 사용되는 MCU(Micro-Controller Unit)는 2018년 가을을 기준으로 기준 누적 300억 개 이상이 출하되었고, 향후 수년간 평균 30억 개 이상의 MCU가 출하될 것이라 전망하였다[1].

이처럼 임베디드 시장은 꾸준히 발전하고 있음에도 불구하고 임베디드 환경을 위한 보안 장치 연구는 여전히 부족한 상황이다. 그 원인으로는 임베디드 기기 생산에 요구되는 경제적 효율성이 있으며, 그 특성상 보안 성능은 우선되어 고려되지 않기 때문이다. 결국 별도의 보안장치가 마련되어 있지 않은 경우가 많은 임베디드 환경은 다양한 경로를 통해 시스템을 위협

Received(05. 22. 2019), Modified(08. 12. 2019), Accepted(09. 25. 2019)
* 본 논문은 2018년도 정부(교육부)의 재원으로 한국기본연구재단의 지원을 받아 수행된 기초연구사업임 (No. NRF-2017R1D1B03034775).
† 주저자, jinu@cnu.ac.kr
‡ 교신저자, clee@cnu.ac.kr(Corresponding author)

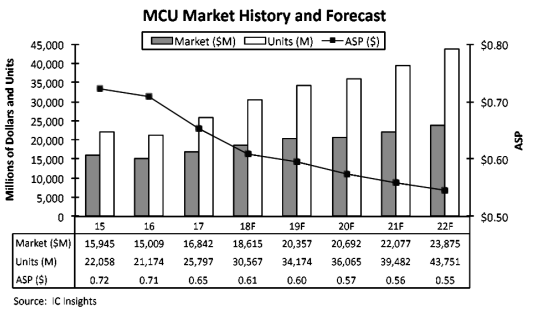


Fig. 1. Yearly world MCU market size

받는다. 임베디드 시스템의 보안과 취약점을 분석한 관련 연구에 따르면, 3,826건의 CVE(Common Vulnerabilities and Exposures) 분석 결과 대다수의 공격이 인터넷을 통한 접근과 로컬, 원격 환경에서의 접근으로 이루어졌으며 OS와 펌웨어가 그 공격의 주된 공격 대상이었다[2]. OS와 펌웨어에 대한 공격은 민감한 정보에 대한 훼손 또는 새로운 OS와 펌웨어의 설치와 같이 시스템의 무결성을 해치는 형태로 이루어졌으며, 공격이 성공한 이후에는 시스템의 관리자 권한을 악용하여 민감한 정보에 대한 접근, 악성 소프트웨어의 설치와 실행, DoS(Denial of Service)와 같은 동작이 있었다.

실례로 2018년 1월 경, 한 사용자의 해킹으로 인해 사용자가 소유한 통신회사의 무선 공유기의 커널이 변조되어 기존의 커널이 아닌 커스텀 펌웨어(사용자의 임의로 조작된 펌웨어 환경)가 탑재된 경우가 있었다[3]. 해킹된 공유기의 환경은 부트로더로 널리 사용되는 U-BOOT를 통해 지정된 커널을 로드하는 형태였으며, 해킹 과정에서 공유기 내부에는 시스템의 기밀성을 위한 어떠한 보안도 존재하지 않았다. 더욱이, 시스템에서 사용되는 암호는 이미 취약점이 많이 노출되어 더 이상 보안목적으로 사용하지 않는 해시 기술인 MD5(Message Digest 5)[4]를 사용하는 등, 보안에 대한 위협이 다수 존재했었다.

이와 같은 보안해이는 비단 위와 같은 통신사 공유기의 사례만이 아닌, 대다수의 임베디드 시스템 환경에서 발견되고 있다. 사용자가 임베디드 기기를 사용함에 있어 불편함을 느끼지 않도록 임베디드 시스템 설계 시 각종 보안 절차를 건너뛰거나 포함시키지 않기 때문이다. 하지만 편의성을 고려한 설계는 결국 잠재적인 위협이 되고 있다. 최근에는 이러한 보안 해이를 방지하기 위해 다양한 방면으로 보안 대책과 기준이 연구되고 있으며, 그 기준 중에는 보안 부팅과 같은 펌웨어 단계에서의 시스템 무결성 검증 기술이 임베디드  시스템의 기저에 존재해야 한다고 제시되고 있다[5].

이에 본 논문에서는 보안 부팅의 개념을 설명하고, 임베디드 환경에서의 TPM을 통한 시스템 무결성 보호와 이를 바탕으로 한 기초적인 보안 부팅 구조를 제안 및 구현한다. 구현할 보안 부팅은 TPM이 탑재된 임베디드 리눅스 환경을 기준으로 구현 과정과 방법에 대해 서술한다. 서론에 이어 본 논문의 2장에서는 보안 부팅의 원리와 관련 연구로 PC환경의 UEFI 보안 부팅, 모바일 환경의 Samsung Knox, FPGA 환경에서의 보안 부팅과 같이 다양한 플랫폼에 적합하게 설계된 보안 부팅의 설명과 장점, 그리고 그 한계점을 소개한다. 다음으로 3절에서는 기존의 Non-Secure 부트로더와 TPM의 기능 설명에 이어 TPM을 활용한 보안 부팅 설계 모델에 대해 기술한다. 4절에서는 3절의 설계를 기반으로 한 실제 구현과 성능에 대한 분석, 기존의 연구와 비교하여 기술한다. 마지막으로 5절에서는 결론과 향후 연구 방향을 제시한다.

## II. 관련연구

보안 부팅은 신뢰 체인(CoT, Chain of Trust)으로 일컬어지는 무결성 확장 방식을 사용한다. 신뢰 체인은 Fig. 2.와 같이 검증된 RoT(Root of Trust)로부터 연쇄적으로 연결된 시스템 계층의 무결성이 훼손되지 않았음을 보장하는 방법이다.

시스템에 대한 무결성의 증명은 사용자 자신의 프로그램이 제조사로부터 인증된 안전한 환경에서 작동한다는 신뢰성을 제공한다[6]. 보안 부팅 구현을 위해 제조사는 일반적으로 신뢰성이 검증된 최하위 단계의 BootROM 코드를 비롯하여 SHA, RSA 인증서와 같은 보안 알고리즘과 시스템에 탑재한 별도의 보안 회로와 같은 보안 설계를 통해 RoT를 제공한다. 보안 부팅은 이렇게 설계된 RoT를 기반으로 부팅 과정에서 상위 단계로 전환이 발생하는 경우 상위 단계의 무결성을 하위 단계에서 검증할 수 있도록 한다[7]. 만일 부팅 단계에서 무결성이 훼손된 계층이 존재하는 경우엔 설계에 따라 단순히 부팅을 멈추거나, 시스템의 복구를 수행하거나, 단순 경고 메시지를 출력하고 부팅을 진행하거나, 악의적인 공격의 위협으로부터 시스템의 정보를 보호할 수 있도록 시스템의 보안기능에 대한 잠금(Lock)을 수행한다[8].

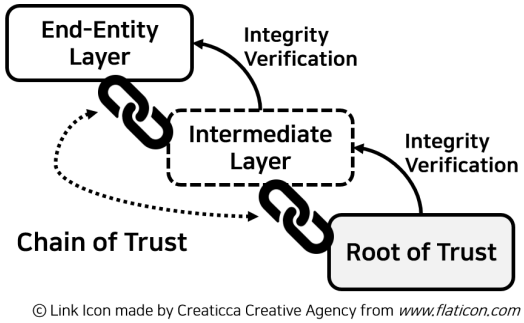


Fig. 2. RoT based trust chain architecture

이처럼 시스템의 무결성 정보는 RoT를 위한 시스템 보안의 가장 중요한 요소로써, 무결성 정보에 대한 보안성은 곧 보안 부팅의 보안성이라고 볼 수 있다. 현재 시장에 상용화된 보안 부팅은 시스템의 H/W적 사양을 비롯한 사용 환경과 목적에 따라 각기 다른 형태로 무결성 정보를 보호하고 있으며, 이를 통해 다양한 보안 기능을 제공하고 있다.

## 2.1 UEFI 보안 부팅

일반 PC 사용자를 위한 보안 부팅 환경으로는 PC의 메인보드에서 제공하는 UEFI(Unified Extensible Firmware Interface, 통일 확장 펌웨어 인터페이스)의 보안 부팅이 있다. Windows와 Linux와 같이 검증된 OS와 드라이버만을 실행시키기 위한 구조의 UEFI 보안 부팅은 무결성 검증 방식으로 RSA 디지털 서명을 사용한다[9].

Fig. 3.에 묘사된 바와 같이 보안 부팅 간 사용할 제조사의 RSA 공개키를 미리 지정해 두기 위해 UEFI와 인증 협약을 맺어야 하며, 이후 메인보드 제조사에서는 메인보드의 ROM에 인증된 제조사의 RSA 공개키를 저장하여 제공한다. 사용자가 보안 부팅을 사용하도록 설정하면, 이후 보안 부팅 루틴에서는 로드할 OS 부트로더에대한 제조사 RSA 인증서를 검증하게 된다. 인증 대상이 되는 OS 부트로더에는 Microsoft社, Redhat社, Apple社와 같은 인증된 기업의 개인키로 디지털 서명된 RSA 인증서가 포함되어있어야 하며, RSA-2048과 SHA-256 이상의 알고리즘으로 생성된 인증서만을 사용해야 한다. 부트로더에 인증서가 포함되어 있지 않거나, 디지털 서명 검증이 실패하는 경우 시스템이 중단된다.

위 과정을 통해 UEFI 보안 부팅은 RSA 인증서 수준의 보안을 보장할 수 있고, OS에 대한 인증에서

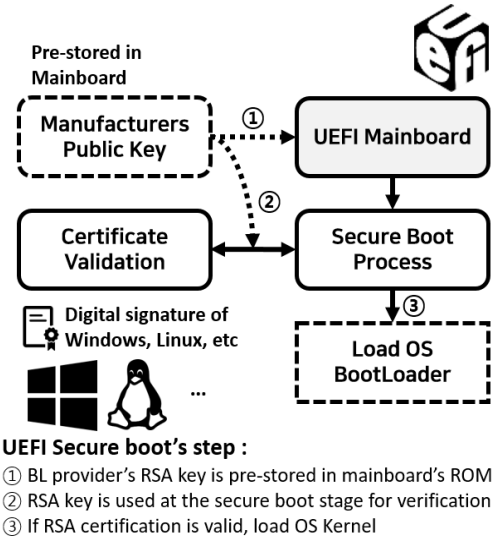


Fig. 3. UEFI Secure boot process

나아가 시스템에 연결된 디바이스 드라이버에 대한 검증 또한 가능하다. 하지만, 메인보드 제조사와 제휴된 OS 제조사가 아닌 경우엔 보안 부팅 지원이 난해하며, 더욱이 만일 제조사의 RSA 개인키가 유출 또는 도난당하는 경우와 공격자의 물리적 탈취로 인한 메인보드 ROM의 변조행위가 발생하는 경우와 같이 보안 부팅 외적인 요소가 부트로더 무결성 보호에 위협이 될 가능성이 있다. 이런 문제를 해결하기 위해 UEFI에서는 TPM 지원을 통한 시스템에 대한 네트워크 무결성 인증과 Windows의 BitLocker와 같은 디스크 암호화 기능과 같이 추가적인 시스템 무결성 보안을 제공한다.

## 2.2 Samsung Knox

현대에 이르러 사용자가 가장 흔하게 접하는 컴퓨터 환경이 데스크톱 PC에서 스마트폰과 같은 휴대용 단말기로 변화함에 따라 모바일 기기에 대한 강력한 보안이 요구되게 되었다. 이에 안드로이드 OS를 사용하는 스마트폰 제조사인 삼성에서는 안드로이드 OS를 사용하는 스마트폰 환경을 위한 보안 솔루션인 Knox를 개발하여 시장에 출시하였다[10].

Knox는 여러 보안 기능을 융합하여 시스템 무결성을 유지하는데, 중요한 요소로 ARM社가 설계한 프로세서 아키텍처인 Cortex의 TrustZone 기능이 있다. TrustZone은 Cortex 프로세서 내부에 일반

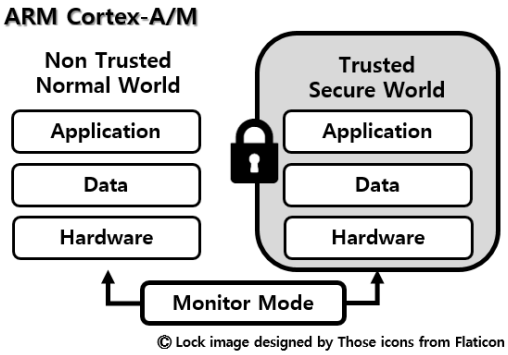


Fig. 4. ARM TrustZone's Secure world[11]

어플리케이션 영역인 Normal World와 보안을 위한 Secure World 영역을 구분하여 제공한다. Fig. 4.와 같이 Secure World는 자신의 영역 외부로 자신의 정보를 노출시키지 않으며, Secure World에서 Normal World로의 접근 및 제어는 가능하지만, 반대로 Normal World에서 Secure World로의 접근은 불가능하고 오직 Secure World 관리자 또는 인증된 어플리케이션만이 접근 가능하다[11]. TrustZone의 이 특성을 활용하여 Knox에서는 Secure World 내부에 존재하는 보호된 메모리 영역에 자신의 무결성 정보를 저장하고, 제조 단계에서 TrustZone에 인증된 자사의 Knox 보안 부팅만이 이 무결성 정보에 접근이 가능토록 설계되었다.

비단 프로세서의 기능을 활용한 보안 기능 외에도 Knox에서는 보안 침해 사실을 알리는 Knox 보증 퓨즈를 비롯하여 상대적으로 보안이 취약한 낮은 펌웨어 레벨로의 업데이트를 방지하는 롤백 방지 퓨즈와 같이 다양한 보안 장치를 통해 시스템의 무결성을 보호한다.

이처럼 Knox는 보안 부팅 과정에서 프로세서의 보안 기능을 적극 활용하여 스마트폰 환경에서의 시스템 무결성을 보호하고, 나아가 보안 부팅으로부터 이어진 CoT를 통해 사용자 환경에서의 TEE(Trust Execution Environment, 신뢰 실행 환경)기능을 제공한다. 비록 현재는 iOS와 구현 안드로이드OS 등 일부 모바일 환경에선 Knox 솔루션을 제공받을 수 없지만, 그 보안 우수성을 인정받아 2014년 구글 I/O에서 안드로이드 환경에 기본적으로 Knox가 탑재될 것이라 발표되었다[12].

## 2.3 FPGA 환경에서의 보안 부팅

PC, 스마트폰 외 FPGA(Field Programmable Gate Array)와 같은 환경에서는 보안 부팅을 구현하기 위해 블록 램(Block RAM)과 같은 기능을 통해 무결성을 보호할 수 있다[13].

블록 램은 설정에 따라 RAM 또는 ROM으로 사용이 가능한데, Fig. 5.와 같이 ROM으로 설정된 블록 램에 시스템의 무결성 정보를 저장함으로서 보안 부팅에서 요구되는 RoT를 수정과 훼손으로부터 보호할 수 있다. 블록 램에 저장되어 보호된 RoT를 부트 과정에서 측정된 커널의 해시값과 비교하여 현재 로드할 커널이 변조, 훼손되지 않았음을 알 수 있다. 여기서 시스템이 업데이트 될 때마다 블록 램에 저장된 해시값을 갱신해야하는 번거로움을 해소하기 위해 해시값이 아닌 RSA 공개키를 저장해 RSA 인증서를 활용한 디지털 서명 방식 또한 가능하다.

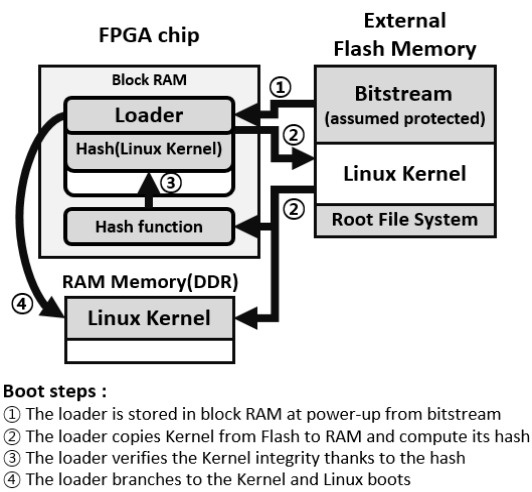


Fig. 5. Secure boot on FPGA environment[13]

## 2.4 기존 보안 부팅의 한계

상용화된 대부분의 보안 부팅은 자사 제품군에 대한 최적화된 보안을 위해 개발되었다. Table 1.에 정리된 바와 같이 UEFI 보안 부팅의 경우 MS사의 인증을 통한 UEFI와의 기술 협약 없이는 UEFI 환경에서의 Secure Boot를 개발할 수 없는 한계가 있으며, 임베디드 환경에서의 보안 부팅 개발은 제조사별로 다른 H/W 환경으로 인해 서로 다른 보안 부팅 모델과 규격, 제조사에 대한 보안 의존성이 생기게

Table 1. Summary of present secure boot

| Classified List | Root of Trust | OS Support | Extended features | Advantage / Disadvantage |
|---|---|---|---|---|
| UEFI 보안 부팅 | Flash ROM in mainboard | Windows, Linux, iOS. | Device Driver Access Control | PC optimized / Need Technical Agrmt. |
| Samsung Knox | Processor, ROM, Various Fuses | Android | Knox Service | Finished security / Mfg. dependency |
| FPGA 보안 부팅 | Block RAM | Embedded OS | N/A | Flexible development / Unstandardized |

되었다. 임베디드 환경에서의 규격화되지 않은 보안 부팅방식은 소규모 임베디드 환경에서 개발이 중단, 지연되는 원인이 될 수 있으며, 보안업체와의 기술 협약의 경우 기술에 대한 인세가 부담이 될 수 있다.

## III. 보안 부팅 모델 설계

이에 보안 부팅을 탑재한 임베디드 시스템을 개발하고자 하는 개발자를 위해 기초적인 설계와 구현 방식을 구체적으로 제안하고자 한다. 인가되지 않은 사용자, 악의적인 공격자로 인한 무결성을 훼손을 방어하기 위한 보안 부팅을 구현하기 위해서는 기존의 부트 과정에 대한 이해와 공격자로부터 노출된 보안 취약점, 제조사의 개발 환경을 고려하여 설계해야한다. 우선 기존 임베디드 시스템의 부팅 과정을 알아본 다음, 임베디드 환경에 보안 부팅을 위해 탑재할 TPM의 보안 기능 설명과 이어 이를 활용한 기초적인 보안 부팅 모델을 제안한다.

### 3.1 기존 임베디드 시스템의 부팅 과정

기존의 대다수 임베디드 시스템 환경에서 사용하는 부트 과정은 Fig. 6.과 같이 4단계로 구성된다. 우선적으로 제조사로부터 제공되는 BootROM 영역의 코드를 통해 시스템이 수행된다. 이어 하드웨어의 레지스터, 메모리의 초기화 등을 수행하는 FSBL(1st Stage Boot Loader)과 현재 시스템의 하드웨어에 대한 최적화 수행하는 SSBL(2nd Stage Boot Loader)이 연속적으로 수행되고 최종적으로 커널의 이미지가 호출되는 형태이다. SSBL은 환경에 따라서 별도로 호출되지 않고 바로 커널로 건너뛰는 경우도 존재한다.

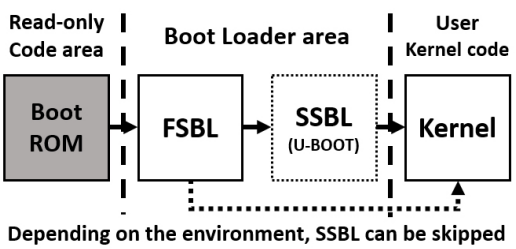


Fig. 6. Basic boot process on embedded system[14]

본 부트 과정은 임베디드 시스템에서 요구되는 간편하고 빠른 부팅을 위해 하드웨어와 시스템의 최적화, 커널 로드만이 이루어지기 때문에 시스템이 로드할 커널에 대한 검증과 부트로더 스스로에 대한 검증 또한 이루어지지 않는다는 취약점이 존재한다.

### 3.2 TPM 1.2의 기능

TPM은 TCG(Trusted Computing Group)에서 지정한 PUF(Physical Unclonable Function, 물리적 복제가 불가능한 기능) 구조의 하드웨어 보안 모듈이다[15]. TPM 내부에는 Fig. 7.과 같이 보안에 필요한 여러 기능이 탑재되어 있으며, 이는 보호된 통신 인터페이스를 통해 프로세서와 통신한다.

본 논문에서는 보안 부팅을 위해 TPM의 주요 기능 일부를 사용한다. 무결성 측정을 위해 TPM의 보안 레지스터인 PCR(Platform Configuration Register)을 사용하며, 측정된 무결성 정보를 암호화하기 위해서 Seal Data(정보 봉인) 기능을 수행한다. 암호화된 무결성 정보는 TPM 내부에 설계된 비휘발성 메모리 영역인 NVRAM(Non-Volatile RAM)에 저장하여 이를 보안 부팅에 활용할 수 있도록 했다.

보안 부팅 간 사용되는 TPM 기능들의 보안성은 보안 해시와 접근제어, 비대칭 암호화를 배경으로 제공된다. 보안 해시의 경우 PCR에 값 입력 시(PCR

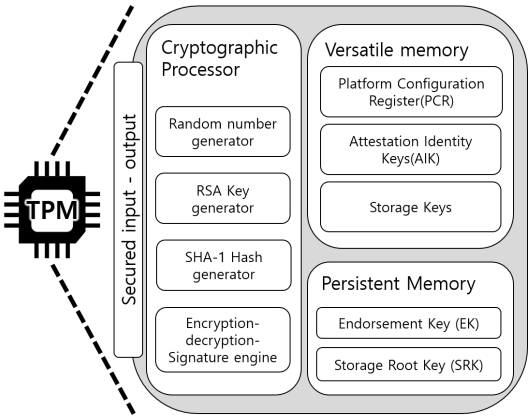


Fig. 7. Security features provided by TPM 1.2

Extend) 부트로더 코드에 대한 SHA-1를 수행하여 나온 해시값(160 bytes)을 저장하며, 데이터를 봉인하는 Seal Data는 TPM 고유의 데이터와 PCR에 저장되어 있는 값, 사용자 패스워드인 SRK(Storage Root Key) 패스워드를 혼합하여 데이터를 암호화한다. 암호화된 데이터는 봉인된 데이터(Sealed Data)로 불리며, 봉인된 데이터는 Seal Data를 수행한 TPM에서만 해제가 가능하다.

NVRAM의 경우 내부 데이터 영역 할당 시 읽기, 쓰기 권한에 대한 설정이 가능하고, 접근제어를 위해 할당된 NVRAM 영역에 대한 패스워드를 설정하여 할당된 영역에 쓰인 정보에 대한 악의적인 수정을 방지할 수 있다. NVRAM은 PCR과 다르게 다양한 보안 정보를 저장할 수 있으며, 접근제어를 기반으로 한 RoT의 기반으로 보안 부팅을 설계하여 저장되는 정보에 따라 다양한 인증 방식을 채택할 수 있다.

### 3.3 보안 부팅 설계 모델

본 논문에서 구현한 보안 부팅은 Fig. 8.과 같이 SSBL 단계 직후에 이루어지며, TPM에 저장된 부트로더의 무결성 측정값을 RoT로 사용한다. 시스템에 대한 무결성 측정은 FSBL 단계에서 이루어지며, 무결성 측정 방식은 BootROM의 CRC(Cyclic Redundancy Check)값과 부트로더 영역의 코드 데이터를 해싱하여 얻는 방식으로 측정한다. BootROM과 FSBL, SSBL에 대한 해싱으로 획득한 3개의 측정값은 PCR 영역에 저장되며 이후 시큐어 부팅 단계에서 무결성 비교에 사용된다. 위와 같은 무결성 측정은 부팅 시 매번 이루어지며, 해당 무

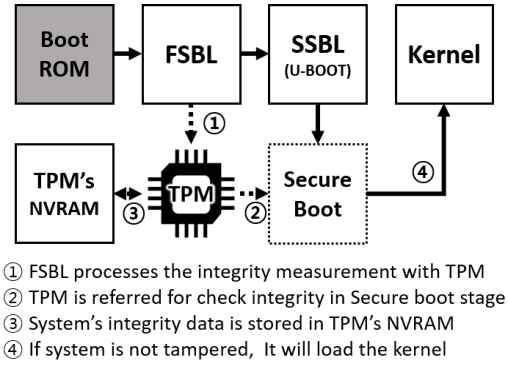


Fig. 8. The Secure boot architecture to implements

결성 값을 참조하여 부트 과정을 수행한다.

#### 3.3.1 최초 실행시의 보안 부팅

보안 부팅의 최초 실행은 제조사 또는 신뢰된 사용자에 의해 수행된다. 본 과정에서는 RoT의 확보와 커널의 암호화 과정, TPM의 초기화가 포함된다.

우선, 임베디드 환경이 구성된 직후에는 TPM과 보안 부팅에 대한 설정이 이루어지지 않은 상태이므로, 보안 부팅은 Fig. 9.와 같이 아직 암호화되지 않은 커널을 로드하는 초기 실행단계로 분기한다. 커널 환경에서 TPM의 사용자 등록과 NVRAM의 접근제한 설정을 마친 후, 커널 이미지에 대한 SHA-1 해싱과 커널 이미지에 대한 AES(Advanced Encryption Standard)-128 암호화를 수행한다. 커널에 대한 무결성 정보(SHA-1 해시값)와 암호화에 사용된 AES-128키는 Seal Data를 통해 암호화를 수행한다. 이 과정에서는 부트로더의 무결성 측정값이 저장된 PCR을 참조하여 수행되어야 하며, TPM 설정에 따라 Seal Data 시 SRK 패스워드를 요구하도록 설정할 수도 있다. 본 논문의 구현에서는 사용자의 편의성을 위해 SRK 패스워드를 별도로 입력하지 않도록 Well-known SRK를 설정하였다. 무결성 측정값이 저장된 PCR을 통한 Seal Data암호화가 완료된 다음에는 암호화된 데이터를 접근제한이 설정된 NVRAM에 저장한다. 해당 NVRAM의 접근제한 설정은 무결성 정보에 대한 훼손을 방지할 수 있도록 인가된 사용자만 쓸 수 있도록 하였으며, 모든 사용자가 읽을 수 있게 설정하였다.

커널에 대한 암호화를 비롯하여 최초 실행시의 보안 부팅 과정이 완료되었다면, 시스템은 다음 부팅부터 무결성 검증 과정을 통한 보안 부팅을 수행하게

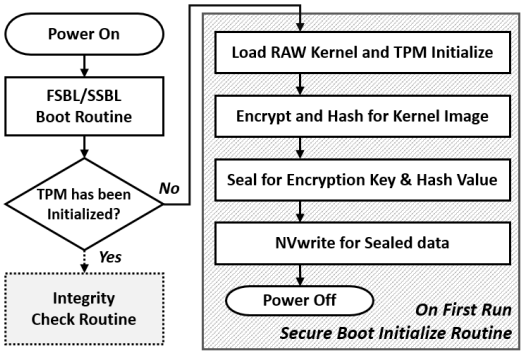


Fig. 9. Secure boot initialize routine at first run

된다. 본 과정은 무결성 저장을 위한 과정으로 NVRAM에 저장되는 정보와 커널에 대한 관리 방식에 따라 다른 방법으로 대체가 가능하다.

### 3.3.2 무결성 검증과정을 통한 보안 부팅

보안 부팅의 최초 실행 과정을 마친 차기 실행부터는 TPM이 초기화되어 있고, NVRAM에 Seal Data를 통해 암호화된 봉인 데이터가 존재하므로 Fig. 10.의 무결성 검증 루틴으로 분기하게 된다.

우선 NVRAM에 저장되어있는 봉인된 데이터에 대한 Unseal data 복호화를 수행한다. 만일, 현재 부트로더에 대한 변조, 훼손이 일어난 경우 Seal Data를 수행한 시점의 PCR과 현재의 PCR 값이 다르므로 복호화가 수행이 되지 않는다. 이 경우 부팅이 중지된다. 하지만 변조, 훼손이 일어나지 않은 경우엔 올바르게 복호화가 이루어지고, 저장되어있던

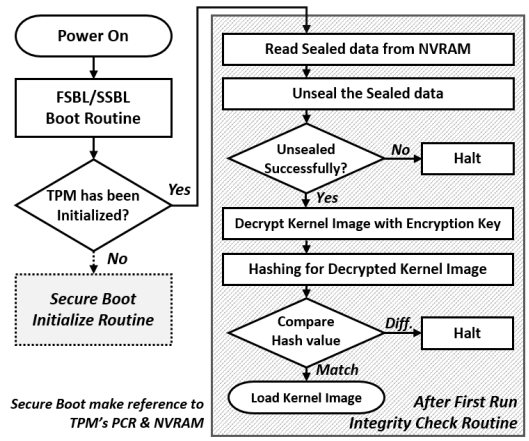


Fig. 10. Integrity verification in Secure boot process

AES-128키와 커널에 대한 해시값을 구할 수 있다. 보안 부팅은 확보한 키를 통해 커널에 대한 복호화를 수행하고, 복호화 된 커널에 대한 SHA-1 해시를 수행한다. 본 과정은 HMAC(Hash-based Message Authentication Code, 해시 기반 메시지 인증 코드)처럼 기존에 봉인되어 있던 해시값과 현재 커널에 대한 해시값을 비교함으로써 커널에 대한 무결성에 대한 검증이 가능하다. 각 해시값이 동일한 경우, 보안 부팅은 복호화 된 커널을 메모리에 적재하고 코드를 실행하여 부팅을 완료한다. 하지만 커널에 대한 해시값이 변화되어 공격이 의심되는 경우에는 올바른 커널 이미지가 아닌 것으로 판별하고 시스템을 종료한다. 본 설계의 전체적인 부팅 흐름은 Fig. 11.에 묘사된 바와 같다.

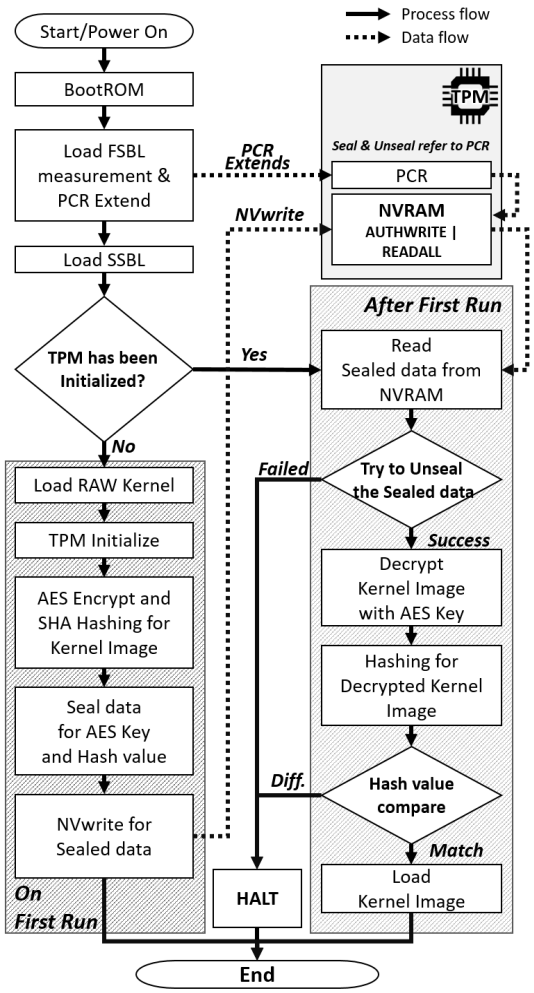


Fig. 11. The overall routine of Secure boot

#### 3.3.3 설계 모델 분석

본 논문에서 사용한 보안 부팅 모델은 임베디드 시스템 환경에서 TPM을 활용하여 무결성을 보호하고, 부팅 중 부트로더와 커널 이미지에 대한 무결성을 기반으로 시스템을 검증한다.

만일 공격자로 인한 부트로더 또는 커널 이미지가 위·변조된 경우, 커널 덮어쓰기가 발생한 경우에는 TPM을 활용하여 기존에 저장된 암호화키를 통해 올바르게 복호화가 이루어졌는지, 또는 추가적으로 저장한 무결성 정보의 비교를 통해 부트 과정의 훼손 여부 판별이 가능하다, 또한 공격자는 AES로 암호화된 커널에 대해서도 해시 충돌 공격의 과정에 AES 암호화 과정을 추가적으로 수행해야 하므로 공격의 난해함을 강화할 수 있다. 비록 커널에 대한 AES 암호화 과정은 무결성 검증 과정에서는 크게 요구되는 사항이 아니지만, 단순한 커널 이미지 덮어씌우기 공격을 간단하게 방어할 수 있으며 부팅 연산의 부하에 있어서도 AES의 연산 속도는 128bit 기준 SHA-1과 10% 내외의 차이를 보이는 매우 빠른 알고리즘[16]으로 TPM과의 통신에 비교하면 실제 부팅 속도에는 매우 미미한 영향을 줄 것으로 고려하였다.

마지막으로, 본 설계의 무결성 보안 강화를 위해서는 공격자의 PCR 해시 충돌을 고려하여 본 설계에서 사용된 TPM 1.2의 SHA-1이 아닌 보안 우수성이 증명된 TPM 2.0의 SHA-2, 또는 그 이상의 해시 알고리즘을 통해 해시 충돌의 가능성을 최소화할 수 있다.

## IV. 보안 부팅의 구현

### 4.1 구현 환경

보안 부팅의 구현에 있어 우선 TPM의 동작 환경과 탑재할 시스템에 대해 이해하는 것이 중요하다. 프로세서의 동작 환경을 비롯하여 TPM과의 통신 방식과 GPIO 핀의 연결 등, 사소한 H/W 설정과 같은 호환성을 고려하여 제품을 선택해야 한다. 본 논문에서는 보안 부팅 구현을 위해 Table 2.에 작성한 바와 같이 FSBL과 같은 부트 시 매우 낮은 레벨에서의 코딩이 가능한 칩으로 Xilinx社의 Zynq-7000[17] 프로세서가 탑재된 임베디드 보드인 Zedboard를 사용하였다. TPM은 프로세서와의 통신에 SPI 방식을 사용하는 Infenion社의 TPM1.2 칩인 SLB9670[18]을, 탑재되는 OS로는 Xilinx社에서 제공하는 임베디드 리눅스[19]를 사용하였다. 부트로더 단계에서는 Xilinx社에서 제공되는 FSBL코드와 오픈소스로 활용되는 SSBL인 U-BOOT를 기반으로 설계하였다.

Table 2. Specification of target embedded system

| Embedded Board Environment | |
|---|---|
| Board Name | Digilent Zedboard |
| Processor | Xilinx Zynq(TM)-7000 |
| RAM | DDR3 512MB |
| TPM | Infineon IRIDIUM SLB 9670 TPM1.2 LINUX |
| OS | Xilinx Embedded Linux |
| **Development Environment** | |
| OS | Windows 10 / 64-bit & Ubuntu 16.04 |
| IDE | Vivado Design Suite 2017.4 & GCC compiler |

본 설계구현에서는 U-BOOT의 커널 탑재 시점에서 TPM과의 통신과 부트로더와 커널의 무결성 검증 단계를 추가하였으며, 이를 위해 SHA-1 알고리즘과 TPM 1.2의 SPI지원을 위한 코드를 추가적으로 삽입하였다.

### 4.2 보안 부팅 개발의 고려사항

TPM을 실제 임베디드 장치에 탑재하여 보안 부팅 과정을 개발하는 경우, 임베디드 환경에 따라 고려해야 할 사항이 존재한다.

우선, 일반적인 FSBL과 같은 부트 초기 단계에서는 TPM을 위한 인터페이스 또는 모듈이 존재하지 않으므로 TPM을 위한 트랜잭션을 직접 제어해 주어야 한다. 트랜잭션에는 데이터 프레임 기반의 TPM 통신과 응답대기시간에 대한 제어가 포함되어야 한다. 이어 SSBL 단계로 선정한 U-BOOT 단계에서는 일부 TPM 모델의 인터페이스를 제공하지만 탑재되지 않은 TPM 모델의 경우, 마찬가지로 직접 TPM과 통신을 하는 소스코드를 탑재하여야 한다. 낮은 단계에서의 부트로더 변경에는 TPM과의 통신 이외에도 SHA 알고리즘과 AES 알고리즘 등을 삽입해야 하므로 부트로더의 용량과 연산 속도를 고려할 필요가 있다.

이후 커널 단계에서는 FSBL과 SSBL과 다른 방

식의 접근이 요구된다. 낮은 부트 단계에서는 TPM의 회로상 주소의 직접 접근이 허용되었지만, 리눅스와 같은 OS의 경우 장치의 보호를 위해 TPM의 접근 주소가 가상화되어 맵핑되므로 직접 접근이 아닌 커널에서 제공하는 접근 방식을 사용하여야한다. 리눅스의 경우 외부 장치와의 통신에 ioctl과 같은 I/O 제어 API를 제공하므로 이를 적극적으로 사용할 수 있다.

TPM과의 통신은 매번 신중하게 이루어져야 하며, 브루트-포스 공격으로부터 TPM을 보호하기 위해 통신 과정에는 일정한 Time-out 딜레이가 존재하므로 이로 인한 통신 오류에 대한 대처가 필요하다. 적절한 대처를 통해 보안 부팅 간 TPM과의 통신 오류로 인한 부트 과정의 지연을 방지할 수 있도록 한다.

### 4.3 보안 부팅 구현 결과

TPM을 임베디드 보드에 탑재한 이후 보안 부팅 실행 디버그 화면은 Fig. 12.와 같다. 커널을 로드하는 단계에서 지금까지 수집한 부트로더의 무결성 값을 통해 NVRAM에 저장되어 있던 Seal Data에 대한 해제를 수행하고, 이를 통해 내부에 저장되어 있던 암호화키를 확보할 수 있었다. 실제 제품에 탑재되는 경우에는 통신 결과에 대한 출력은 삭제해야 하며, 키가 저장된 메모리 또한 복호화에 사용한 이후에는 최대한 빨리 초기화를 수행해야 한다.

```
COM4 - PuTTY
reading uImage
3989504 bytes read in 229 ms (16.6 MiB/s)
reading devicetree.dtb
10123 bytes read in 14 ms (706.1 KiB/s)
reading uramdisk.image.gz
5310018 bytes read in 305 ms (16.6 MiB/s)
do_bootm
*0x2080000 genimag_get_kernel...
*  kernel: cmdline image address = 0x02080000
***Secure Routine

 > Read sealed Data from NVRAM
         > First 32bytes of loaded data
0x01 0x01 0x00 0x00 0x00 0x00 0x00 0x2c 0x00 0x
0x42 0x03 0xad 0xc9 0xfe 0xab 0x5d 0x21 0x90 0x
0x01 0x01 0x00 0x00 0x00 0x00 0x00 0x2c 0x00 0x
         > Sealed data Length : 312
 > Unseal Loaded Data : 32
0x38 0x38 0x38 0x38 0x38 0x38 0x38 0x38 0x38 0x
Unsealed Key :
0x38 0x38 0x38 0x38 0x38 0x38 0x38 0x38 0x38 0x
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x
```

Fig. 12. Secure boot Initialize routine at first run

### 4.4 보안 부팅 구현 결과 분석

부트로더에 대한 용량과 처리시간은 시스템의 보안과 직접적인 연관은 없지만, 상대적으로 낮은 사양에서 간편한 실행과 빠른 응답성으로 사용자에게 편의를 제공하는 임베디드 환경에서는 필수적인 고려 요소이다.

본 구현에서는 사용자의 편의를 고려하여 기존에는 입력이 필요한 SRK 패스워드를 Well-Known Password로 설정해 사용자가 부팅 간 별도의 비밀번호를 입력할 필요가 없도록 구현하였다.

구현 후 보안 부팅으로 인한 오버헤드 측정 결과는 Table 3.에 정리되어 있으며, 부트로더의 용량 측면에서 기존의 FSBL 코드에 SHA 연산과 TPM의 접근, 측정값에 대한 PCR Extends 과정이 추가됨과 함께 U-BOOT에서의 NVRAM에 대한 읽기, 쓰기 및 봉인된 데이터에 대한 Unseal 과정과 무결성 비교를 수행토록 하는 코드, AES 암호화를 비롯한 각종 알고리즘 코드가 추가됨으로서 전체적인 용량이 약 10%가량 늘어난 모습이다. 부트로더의 수행시간 측면에서는 증가한 용량에 비해 상당히 많은 시간이 소요되었는데(수행시간 약 27% 증가), 시스템의 무결성 검증 간 다수의 SHA 연산과 커널의 복호화에는 비교적 많은 시간이 소요되지 않았지만, TPM과의 통신에서 상당한 시간이 소요되었다. 특히 봉인된 데이터를 해제하는 TPM의 Unseal data 함수에서 가장 많은 시간이 소모되었는데, 이는 데이터 봉인을 해제하는 과정에서 사용자에 대한 인증, 무작위 난수 생성 등 프로세서와 TPM간의 잦은 통신과 통신 간 요구되는 응답대기시간으로 인해 전체적인 실행 시간이 크게 증가한 것이다. 본 구현에서 TPM과의 통신에 사용된 SPI 프로토콜은 통신 속도가 연결된 프로세서의 성능에 비례하기 때문에 고성능의 프로세서를 사용한다면 SPI를 사용해도 무관하지만,

Table 3. Overhead measurement of Secure boot

| | File Size | | Execution Time |
|---|---|---|---|
| | U-BOOT | BOOT.bin | |
| Basic Boot | 3,393 KB | 4,216 KB | 4,959 ms |
| Secure boot | 3,486 KB | 4,634 KB | 6,329 ms |
| Diff. | + 73 KB | + 418 KB | 1,370 ms (Unseal data 589ms) |

만일 저성능, 저전력 환경과 같이 제한적인 환경에서 수행시간의 개선이 필요하게 된다면, 프로세서 성능과 무관하게 일정한 통신 속도를 제공하는 I2C 프로토콜의 채택을 고려할 수도 있다.

## V. 결 론

최근 다양한 해킹 툴이 개발되고, 임베디드 기기의 보안 취약성이 드러남에 따라 임베디드 시스템에 대한 공격방식 또한 발전하고 있다. 그 공격 방식으로는 단순 시스템의 훼손과 OS 재설치 등의 공격뿐만이 아니라 부트로더에 대한 랜섬웨어 공격과 같이 사용자에게 더욱 직접적인 피해를 입히는 형태로 발전하고 있으며, 이를 방어하기 위한 대책 마련이 시급한 상황이다. 하지만 임베디드 스타트업 기업에서는 보안 부팅 관련 정보 부족과 개발의 난해함으로 인해 개발 방식에 대한 접근이 어려운 현실이다.

이에 본 논문에서는 사용자가 신뢰 가능한 임베디드 시스템 환경의 구축을 위해 검증된 보안 모듈인 TPM을 활용하여 임베디드 환경의 기초적인 보안 부팅을 구현하였다. 본 구현에서는 부트로더의 무결성 측정값을 기반으로 RoT를 구축하였으며, TPM의 내부저장소와 TPM의 독자적인 데이터 봉인 기능, 커널에 대한 암호화 등을 통해 시스템의 무결성을 보호할 수 있도록 설계하였다. 본 구현을 통해 임베디드 시스템 개발자 및 제조사는 공격자로 인한 임의의 커널 조작이나 커널 덮어쓰기 등의 공격을 효과적으로 방어하기 위한 시스템을 구축할 수 있으며, 사용자는 이를 바탕으로 임베디드 기기에 대한 신뢰성을 보장받을 수 있을 것이다.

향후 본 논문의 구현을 확장하여 구현에 사용된 TPM의 버전을 1.2에서 2.0으로 변경하거나 ARM Trustzone의 보안 격리 메모리영역과의 융합을 통해 더욱 다양한 보안 알고리즘을 활용하는 형태로 보안의 안정성을 높일 수 있으며, 이 외에도 단순 부트로더의 무결성을 통해 시스템의 무결성을 검증하는 것뿐만 아닌, 나아가 사용자의 어플리케이션과 주변 기기에 대한 무결성의 검증으로 까지 더욱 다양한 방면에서 활용 될 수 있을 것이다.

## References


〔1〕 IC Insights, http://www.icinsights.com/data/articles/documents/1101.pdf, 17. May 2019.

〔2〕 D. Papp, Z. Ma, L. buttyan, “Embedded systems security: Threats vulnerabilities and attack taxonomy”, Privacy Security and Trust (PST) 2015 13th Annual Conference on. IEEE, pp. 145-152, 2015.

〔3〕 manatails’ blog, https://manatails.net/blog/2018/01/KT-%EB%AC%B4%EC%84%A0-%EA%B3%B5%EC%9C%A0%EA%B8%B0-%EC%BB%A4%EC%8A%A4%ED%85%80-%ED%8E%8C%EC%9B%A8%EC%96%B4-%EA%B0%9C%EB%B0%9C%EA%B8%B0/, 17. Jan. 2019.

〔4〕 X. Wang and H. Yu, “How to break MD5 and other hash functions,” Advances in Cryptology-EUROCRYPT 2005, vol. 3494 of LNCS, pp. 19-35, Springer, 2005.

〔5〕 Jae-yong Ko, “Technologies Analysis based on IoT Security Requirements and Secure Operating System”, The Journal of the Korea Contents Association(C), 18(4), pp. 164-177, 2018.

〔6〕 Wilkins, R., Nixon, T., The Chain of Trust, https://uefi.org/sites/default/files/resources/UEFI%20Forum%20White%20Paper%20-%20Chain%20of%20Trust%20Introduction_Final.pdf, 16. Jan. 2019.

〔7〕 Junkai Gu, Weiyong Ji, “A Secure bootstrap Based on Trusted Computing,” 2009 International Conference on New Trends in Information and Service Science, IEEE, Beijing, China, Sep. 2009.

〔8〕 William A. Arbaugh, David J. Farber, and Jonathan M. Smith. “A reliable bootstrap architecture,” Proceedings. 1997 IEEE Symposium on Security and Privacy, IEEE, Oakland, CA, May 1997.

〔9〕 UEFI Forum, Unified Extensible

Firmware Interface Specification Version 2.2 D, pp. 1369-1402, Nov. 2010.
〔10〕 Samsung Electronics, Knox Platform for Enterprise White Paper, Samsung, Feb. 2019.
〔11〕 T. Alves, D. Felton, TrustZone: Integrated Hardware and Software Security. ARM White Paper, ARM, 2004.
〔12〕 ZDNet, https://www.zdnet.com/article/google-samsung-integrate-knox-into-android-eye-enterprise-gains/, 1. May. 2019.
〔13〕 F. Devic, L. Torres, "Securing Boot of an Embedded Linux on FPGA," 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum, pp. 189-195, May 2011.
〔14〕 Lester Sanders, Measured Boot of Zynq-7000 All Programmable SoCs, Xilinx, Inc., Mar. 2017.
〔15〕 TCG Published, TPM Main Specification Level 2 Version 1.2, Trusted Computing Group, Inc., Mar. 2011.
〔1〕 Crypto++, https://www.cryptopp.com/benchmarks.html, 8. June 2019.
〔16〕 Xilinx, Inc., Zynq-7000 SoC Technical Reference Manual, July 2018.
〔17〕 verical, https://www.verical.com/datasheet/infi neon-technologies-ag-secure-microcontrolles-and-tpm-slb9670vq12fw640xuma1-4684070.pdf, 3. Mar. 2019.
〔18〕 Github, https://github.com/Xilinx/linux-xlnx , 20. Jan. 2019.

## 〈저자소개〉

김 진 우 (Jin-Woo Kim) 학생회원
2017년 8월: 충남대학교 컴퓨터공학과 졸업
2017년 9월~현재: 충남대학교 컴퓨터공학과 석사과정
〈관심분야〉 임베디드 시스템, 임베디드/IoT 보안, 실시간 시스템

이 상 길 (Sang-Gil Lee) 학생회원
2014년 2월: 충남대학교 컴퓨터공학과 졸업
2016년 2월: 충남대학교 컴퓨터공학과 석사
2016년 3월~현재: 충남대학교 컴퓨터공학과 박사과정
〈관심분야〉 소프트웨어 보안, 운영체제, 임베디드 시스템,

고 재 용 (Jae-Yong Lee) 학생회원
2016년 2월: 충남대학교 컴퓨터공학과 졸업
2019년 2월: 충남대학교 컴퓨터공학과 석사
2018년 11월~현재: 국가보안기술연구소 보안기술본부 연구원
〈관심분야〉 임베디드/IoT 보안, 정보보호정책, 운영체제 보안

이 철 훈 (Cheol-Hoon Lee) 일반회원
1983년 2월: 서울대학교 전자공학과(공학사)
1988년 2월: 한국과학기술원 전기 및 전자공학과(공학석사)
1992년 2월: 한국과학기술원 전기 및 전자공학과(공학박사)
1983년 3월~1986년 2월: 삼성전자 컴퓨터사업부 연구원
1992년 3월~1994년 2월: 삼성전자 컴퓨터사업부 선임연구원
1994년 2월~1995년 2월: Univ. of Michigan 객원 연구원
1995년 5월~현재: 충남대학교 컴퓨터공학과 교수
2004년 2월~2005년 2월: Univ. of Michigan 초빙 연구원
〈관심분야〉 운영체제, 임베디드 시스템, 고장허용 컴퓨터, 로봇 미들웨어