Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.35, NO.1, Feb. 2025 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2025.35.1.47

IoT 환경에서 안티디버깅 회피를 보완하기 위한 자동화된 디바이스 바인딩 적용 방법 설계 및 구현*

곽 동 규,[1†‡] 김 선 우,[2] 하 재 현[3]
[1,2,3]쿤텍 주식회사 (수석 연구원, 연구원, 선임 연구원)

The Design and Implementation of an Automated Device Binding Application Method to Complement Anti-Debugging Evasion in IoT Environment*

DongGyu Kwak,[1†‡] SunWoo Kim,[2] JaeHyun Ha[3]
[1,2,3]COONTEC Co., Ltd. (Chief Researcher, Researcher, Senior Assistant Researcher)

요 약

IoT 장치는 인터넷에 연결되어 데이터 수집, 전송 및 처리할 수 있는 장치로, 인공지능과 머신러닝 기술의 발전에 따라 다양한 분야에서 활용되고 있다. 그러나, 이러한 장치들은 서비스 개발 비용이 높으며, 공격자들에게 민감정보가 노출되어 보안 문제가 발생할 수 있다. 특히, IoT 장치의 소프트웨어는 디버깅을 통해 악의적으로 분석될 수 있어 안티디버깅 기법이 필요하다. 하지만, 기존의 안티디버깅 기법은 회피 가능성이 있어, 이를 보완하는 방법으로 디바이스 바인딩 기술을 사용할 수 있다. 디바이스 바인딩 기술은 특정 디바이스에서만 소프트웨어가 동작하도록 하는 방법으로, 고유값을 비교하여 인증한다. 본 논문에서는 LLVM Pass를 이용하여 디바이스 바인딩 기법을 자동으로 삽입하는 방법을 제안하며, 디바이스 DNA를 이용한 고유값으로 서명을 생성하고 검증하는 시스템을 보인다.

ABSTRACT

IoT devices are connected to the internet and capable of collecting, transmitting, and processing data. With advancements in AI and machine learning technologies, these devices are utilized in various fields. However, they handle sensitive information and have high service development costs, making them vulnerable to security issues, Particularly, the software of IoT devices can be maliciously analyzed through debugging, necessitating anti-debugging techniques. Existing anti-debugging methods can be bypassed, so device binding technology is helpful in defending against such attacks. Device binding ensures that software operates only on specified devices by comparing unique values for authentication. This parer proposes a method for automatically inserting device binding techniques using LLVM Pass, and describes a system that generates unique values using Device DNA and verifies signatures.

Keywords: Device Binding, IoT, Anti-debugging, LLVM Pass, Device DNA

Received(07. 11. 2024), Modified(1st: 10. 25. 2024, 2nd: 11. 29. 2024), Accepted(12. 05. 2024)
* 본 연구는 2024년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(No. RS-2024-00399373, 고신뢰 의료기기 위험관리를 위한 다차원 공격 표면관리 핵심기술 개발)
† 주저자, kwak@coontec.com
‡ 교신저자, kwak@coontec.com(Corresponding author)

I. 서 론

IoT 장치는 인터넷에 연결되어 데이터를 수집, 전송 및 처리할 수 있는 장치를 의미한다[1]. 최근 IoT 장치는 인공지능과 머신러닝 기술의 발전에 따라 다양한 개인 및 기업 분야에서 활용되고 있다. IoT 장치의 발전에 따라, 편리하고 효과적인 서비스를 제공하기 위해서 다수의 민감정보를 활용하고 있으며[2], 서비스 개발도 비용이 많이 소요된다. IoT 장치의 서비스 개발 비용은 하드웨어 개발 비용과 소프트웨어 개발 비용이 대부분을 차지하는데, 특히 소프트웨어 개발 비용이 많이 요구된다. 그리고 IoT 장치는 다양한 장소에서 손쉽게 설치하여 사용할 수 있는 특징을 가지고 있어 다수의 사용자에게 노출되어 있어 공격자가 쉽게 탈취할 수 있다. 이에 따라, IoT 장치의 민감정보에 대한 보안이나 개발된 응용 소프트웨어의 알고리즘 보호가 필요하다. IoT 장치의 민감정보나 소프트웨어 알고리즘을 악의적으로 탈취하는 공격자는 다양한 방법을 통해 소프트웨어를 분석하는데, 대표적인 분석 방법으로는 응용 소프트웨어를 디버깅하는 방법이 있다[3]. 이를 방어하기 위해 안티디버깅 기법을 적용하여 공격자의 디버깅을 방어한다.

하지만, 공격자는 안티디버깅 기법을 회피하여 공격할 수 있다. 하드웨어나 소프트웨어적 안티디버깅 기법은 일반적으로 해당 IoT 보드나 IoT에 설치되어 있는 OS에 적용하여 디버깅을 차단하는데, 해당 IoT 보드에서 응용 소프트웨어를 추출하여 디버깅이 가능한 IoT 보드에 설치하여 사용하면 안티디버깅 기법을 회피하여 디버깅할 수 있다. 이와 같이 안티디버깅이 적용된 환경을 회피하는 공격을 방어하기 위해서는 안티디버깅이 적용되어 있는 해당 IoT 장치 이외에 다른 장치에서는 실행되지 않도록 하는 디바이스 바인딩 기술이 필요하다.

디바이스 바인딩 기술은 해당 디바이스의 고유값을 실행 시 비교하여 허가한 디바이스에서만 동작하고 허가되지 않은 디바이스에서는 동작하지 않는 방법을 사용한다. 이 기술을 적용하기 위해서는 개발이 완료된 IoT 응용 프로그램에 고유값을 비교하는 소스 코드를 작성해야 하는데, 이를 작성하기 위해서 비용이 소요된다. 그리고 기존의 디바이스 바인딩은 많은 경우 저작권 보호를 위한 라이선스의 용도로 사용[4]하고 있어 민감정보 탈취나 응용 소프트웨어 알고리즘 보호에는 효과를 기대하기 어렵다. 왜냐하면 공격자가 사용하는 디버깅이 가능한 IoT 장치에서 라이선스를 할당받으면 디버깅을 사용할 수 있기 때문이다.

본 논문은 라이선스 방식이 아닌 디바이스 바인딩 기법을 자동으로 삽입하는 방법을 제안한다. 디바이스 바인딩을 응용 소프트웨어에 추가하기 위해 LLVM Pass[5]를 이용하여 자동화된 방법으로 디바이스 바인딩 인증 코드를 삽입한다. 또한, 디바이스 바인딩을 구성하기 위해서는 각 디바이스마다 고유값을 이용하여 인증해야 하는데, 이 고유값이 복제 가능한 경우 무력화된다. 이에 따라, 본 시스템에서는 복제가 어려운 디바이스 DNA (Device DNA)[6] 방법을 이용하여 디바이스 고유값을 생성한다. 그리고 ECDSA 서명 검증[7]을 통해 검증하여 장치를 인증한다.

본 논문 2장에서 관련 연구를 보이고, 3장에서 제안하는 시스템의 구조를 설명한다. 4장에서는 실험 결과를 보이고 5장에서 결론을 맺는다.

II. 관련 연구

본 장에서는 먼저 디버깅을 통한 IoT 장치 및 소프트웨어 분석 관련 연구와 기존에 활용되고 있는 디바이스 바인딩 기술을 살펴보고, 제안하는 시스템과 관련된 개념 및 활용 기술에 대해 설명한다.

2.1 디버깅을 통한 IoT 분석 관련 연구

IoT 보안이 강조되는 만큼, 보안 점검을 위해 취약점 분석이 필요하기 때문에 장치와 소프트웨어 분석에 대한 다양한 연구가 진행되어 왔다.

Omer Shwartz 등은 2018년 역공학 기법을 체계화하여 IoT 장치 보안을 평가하고 개선 방안을 제안하는 연구에서 IP 카메라 등 IoT 장치를 대상으로 UART 포트를 이용한 분석을 시도하였다[8].

Yejun Kim 등은 2021년 IoT 장치의 펌웨어를 추출하고 분석하는 과정을 제안하며 JTAG 등 디버깅 포트를 사용하는 방법과 디버깅 포트에 접근할 수 없는 경우 메모리 덤프를 통한 펌웨어 획득 방법을 설명하였다[9].

두 연구에서 공통적으로 최종 제품에서는 디버깅 포트를 제거하거나 불가피하다면 접근 제어 등 다른 보호 조치를 취할 것을 권고하고 있다. 공격자도 디버깅을 통해 취약점을 분석하거나 중요 데이터 및 알

고리즘을 탈취할 수 있기 때문에 디버깅을 제거하는 방법 뿐만 아니라 다양한 안티디버깅 기법을 적용하는 경우가 많다.

2.2 안티디버깅 기법

안티디버깅 (Anti-Debugging) 기법은 공격자가 디버깅을 이용하여 소프트웨어를 분석하거나 변경하는 공격을 방어하는 방법이다. 안티디버깅 기법은 하드웨어적 방법과 소프트웨어적 방법으로 구분할 수 있다.

하드웨어적 안티디버깅 방법으로는 시판되는 제품의 디버깅 포트를 제거하는 방법이 있다. 장치가 가지고 있는 디버깅 포트 (예 : TAP, Test Access Port)[10]에 하드웨어 디버깅 도구 (예 : TRACE32)[11]를 연결하여 IoT 장치가 사용하는 보드를 디버깅하기 때문에 포트를 제거하여 디버깅을 막는다.

소프트웨어적 안티디버깅 방법으로는 부모 프로세스가 디버거인지 확인하거나, 프로세스 상태값 변화를 관찰하는 등의 소프트웨어적 안티디버깅 기법[12]을 사용한다. GDB (GNU Debugger)로 대표되는 소프트웨어 디버깅 도구를 이용하여 디버깅 대상 프로그램과 함께 동작하며 프로그램을 분석하기 때문에 디버깅 도구가 함께 실행 중일 때 나타나는 변화를 탐지하는 것이다.

그러나 공격자는 인서킷 에뮬레이션 (In-Circuit Emulation, ICE)이나 JTAG 디버깅이 가능한 다른 하드웨어 장치를 이용한 방법을 사용하여 소프트웨어적 안티디버깅 기법을 무력화할 수 있다.

본 논문에서는 인서킷 에뮬레이션이나 JTAG 디버깅을 이용한 공격을 방어하기 위해 비인가 하드웨어에서 프로그램이 동작할 수 없도록 하는 방법으로 디바이스 바인딩을 제안한다.

2.3 서버를 이용한 디바이스 바인딩

디바이스 바인딩은 네트워크를 사용하는 디바이스의 인증 기능으로, 이를 통해 IoT 장치를 인증 등록으로 많이 사용되며 이는 서버와 클라이언트 간의 인증으로 구현한다[13]. 이를 위한 인증 절차는 다음과 같다.

① 장치 고유값 등록 : IoT 장치의 고유값을 서버에 등록을 위한 요청

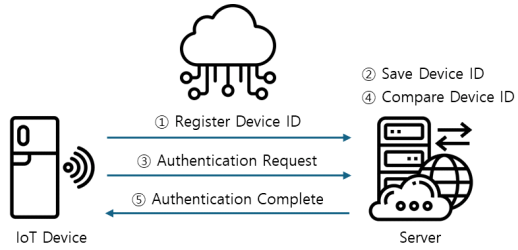


Fig. 1. Authentication Using Device Binding

② 고유값 저장 : IoT 장치의 고유값을 서버에 저장
③ 인증 요청 : 사용자 ID/Password를 통한 인증 요청
④ 고유값 비교 : 서버에서 생성된 공개 키를 이용해 IoT 장치의 고유값을 암호화 전송 후 비교
⑤ 인증 완료 : 인증이 유효함을 확인

위와 같은 인증은 IoT 디바이스의 고유값을 이용하여 인가된 IoT 디바이스를 보증해 준다. Fig. 1.은 인증 절차를 보인다.

2.4 라이선스를 위한 디바이스 바인딩

IoT 장치 개발사가 자사의 지적재산권을 보호하기 위해 디바이스 바인딩을 활용한다[4]. IoT 개발사는 개발된 IoT 응용 소프트웨어를 자사가 인가한 IoT 장치에서만 동작하도록 라이선스를 삽입한다. Fig. 2.는 라이선스를 위한 디바이스 바인딩의 절차를 보인다.

Fig. 2.와 같이 사용자의 요청에 따라 라이선스를 발급하고 해당 라이선스가 단일 환경에서 동작하도록 제한한다. 이런 경우 공격자는 자신의 장치 정보에 대한 라이선스 발급 요청을 하여 소프트웨어를 디버

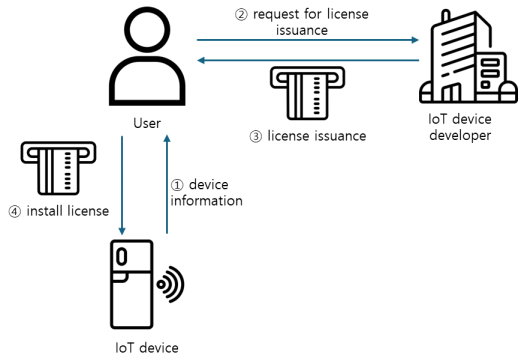


Fig. 2. Device Binding for Licensing

깅이 가능한 비인가 IoT 장치에서 실행할 수 있으므로 동적 분석을 막는 효과를 기대하기 어렵다.

2.5 LLVM Pass를 이용한 코드 삽입

LLVM[14]은 모듈식 재사용 가능 컴파일러 및 툴체인 기술의 모음인 프로젝트로써 LLVM을 기반으로 하는 Clang은 오픈소스 컴파일러로 GCC[15]와 더불어 많이 사용되고 있다. LLVM은 컴파일 과정에서 컴파일 대상인 프로그램의 효과적인 최적화를 위해 중간 언어인 IR (Intermediate Represen-tation)을 사용하는데, LLVM에서 IR을 사용한 최적화 과정의 추상적인 구조는 Fig. 3.과 같다.

Fig. 3.에서 보이는 것과 같이, 먼저 Front-end에서 다양한 고급언어 (C, C++, Go, Rust, Toy 등)를 중간 언어인 IR로 변경한 후 LLVM 최적화 도구(opt)를 통해 최적화를 한다. 최적화 과정에서는 대상 소스 프로그램을 최적화 규칙에 따라 소스를 분석하고 변형하는데, 이때 사용하는 모듈은 LLVM Pass[5]이다.

LLVM Pass는 IR 코드를 분석하여 변환하는 모듈로 특정 작업을 수행하며, 여러 Pass가 순차적으로 실행되어 IR 코드를 변환할 수 있다. 본 논문에서는 디바이스 바인딩 코드를 삽입하는 데 LLVM Pass를 이용한다.

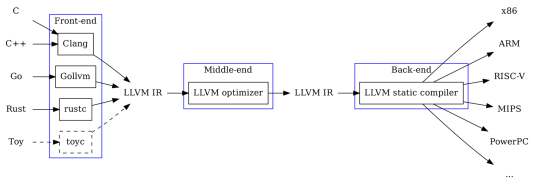


Fig. 3. Abstract Structure of LLVM Optimization

2.6 디바이스 DNA 기술

디바이스 DNA 기술은 각 디바이스마다 고유값을 생성하는 기술이다[6]. 디바이스는 고유의 물리적 특성이나 제조 과정에서 발생하는 미세한 차이점을 이용하여 고유값을 추출하고, 이는 다음과 같은 특성을 가지고 있다.

① 고유성 : 각 디바이스마다 고유한 값을 가지며, 이를 통해 디바이스를 식별할 수 있다.

② 복제 방지 : 물리적 특성을 기반으로 하여 복제가 어렵고 이는 하드웨어 수준의 고유한 값을 생성하기 때문에 소프트웨어적으로 복제하기 어렵다.

③ 보안 강화 : 특정 디바이스에만 소프트웨어가 동작하도록 설정할 수 있어 불법 복제나 무단 사용을 방지할 수 있다.

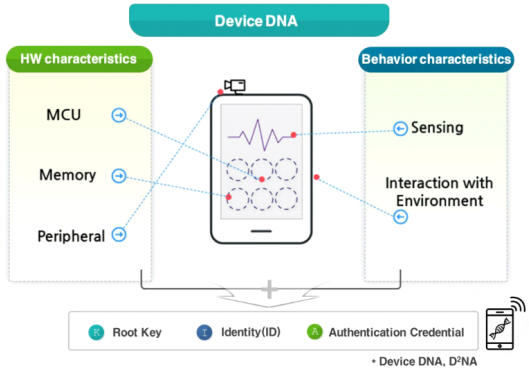


Fig. 4. Device DNA Extraction

Fig. 4.는 디바이스 DNA를 추출하는 구조를 보인다.

본 논문에서는 디바이스 DNA의 값을 이용하여 복제가 어려운 하드웨어 고유값으로 사용하고 이 값을 이용한 인증을 통해 비인가 디바이스에서 소프트웨어를 사용할 수 없도록 한다. Fig. 5.는 디바이스 DNA를 사용하기 위한 보정 데이터 생성과 난수 보정을 보인다.

플래시 메모리는 초기에 고유값을 가지게 되는데 이 고유값은 약 5% 정도 데이터가 고유성을 위배하게 된다. Fig. 5.는 고유성을 확보하기 위한 보정 데이터 생성과 보정 데이터를 이용한 디바이스 ID 생성을 보인다.

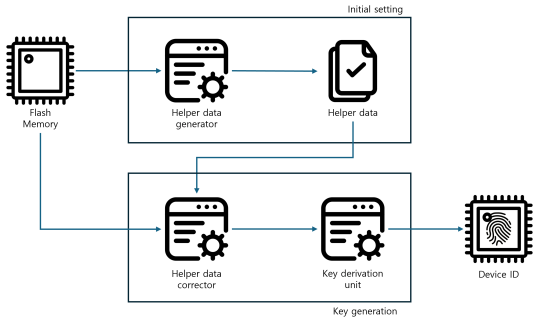


Fig. 5. Device ID Initial setting and generation

2.7 ECDSA 서명

ECDSA (Elliptic Curve Digital Signature Algorithm)는 타원 곡선 암호학(ECC, Elliptic Curve Cryptography)을 기반으로 한 디지털 서명 알고리즘으로 데이터의 무결성 및 송신자의 신원을 보장하는 데 사용된다[7]. ECDSA는 타원 곡선 연산을 통해 서명을 생성하고 검증하며 그 절차는 다음과 같다.

1. 공개 키와 개인 키 쌍을 생성한다. 개인 키는 서명자가 고유하게 보유하고, 공개 키는 검증자가 서명을 확인하는 데 사용된다.
2. 서명자는 서명하려는 메시지의 해시값을 계산하고, 개인 키와 난수(k)를 이용하여 두 개의 서명 값 (r, s)를 생성한다. 이 두 값은 타원 곡선 연산을 통해 구해진다.
3. 메시지와 서명을 받은 검증자는 서명 (r, s)와 공개 키를 사용하여 해당 서명이 올바른지 검증할 수 있다.

ECDSA는 RSA와 같은 기존 서명 알고리즘에 비해 짧은 키 길이로 높은 수준의 보안을 제공하므로, 리소스가 제한된 IoT 기기에서 사용하기 적합하다. 따라서 본 연구에서는 ECDSA 알고리즘을 사용하여 디바이스 고유값을 메시지로 서명을 생성하고 검증하는 방식으로 디바이스 바인딩 기능을 구현한다.

III. 시스템 구조

제안하는 시스템은 안티디버깅을 회피하는 공격인 인서킷 에뮬레이션이나 JTAG 디버깅이 가능한 다른 하드웨어 장치를 이용하는 방법을 방어하는 데 목적이 있다. 그러므로 디바이스 바인딩을 적용할 IoT 응용 소프트웨어는 안티디버깅이 선 적용된 응용 소프트웨어를 대상으로 한다. Fig. 6.은 안티디버깅과 디바이스 바인딩이 함께 적용되는 IoT 응용 소프트웨어 개발환경을 보인다.

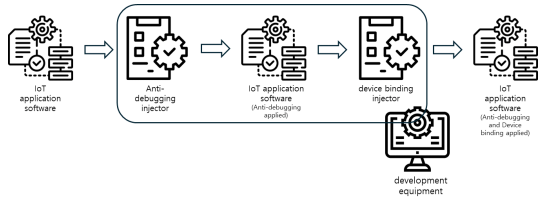


Fig. 6. Automatic Application of Anti-Debugging and Device Binding

안티디버깅이나 디바이스 바인딩 코드를 응용 소프트웨어에 추가하기 위해서는 응용 소프트웨어 개발자가 보안에 대한 지식을 가지고 개발해야 하는데 이는 많은 비용이 소요된다. 제안하는 시스템은 Fig. 6.과 같이 개발이 완료된 IoT 응용 소프트웨어에 자동으로 안티디버깅이나 디바이스 바인딩 코드를 삽입하는 구조를 갖는다. Fig. 7.은 디바이스 바인딩을 위한 장치 인증 코드를 삽입하는 과정을 보인다. LLVM API는 소스 코드를 분석, 수정 및 삽입하는 방법을 제공한다[5]. 본 시스템에서는 Pass를 이용한 장치 인증 코드 삽입기(device authenticator inject Pass)를 통해 장치 인증 함수 호출 코드를 삽입한다. 장치 인증 함수가 정의된 라이브러리로 제공되고 빌드 시 링크한다(Fig. 7.의 device authenticator library). 장치 인증 함수는 인자로 주어진 경로에서 공개 키와 서명 파일을 열기 때문에 인증을 위해 필요한 파일 위치 정보(device

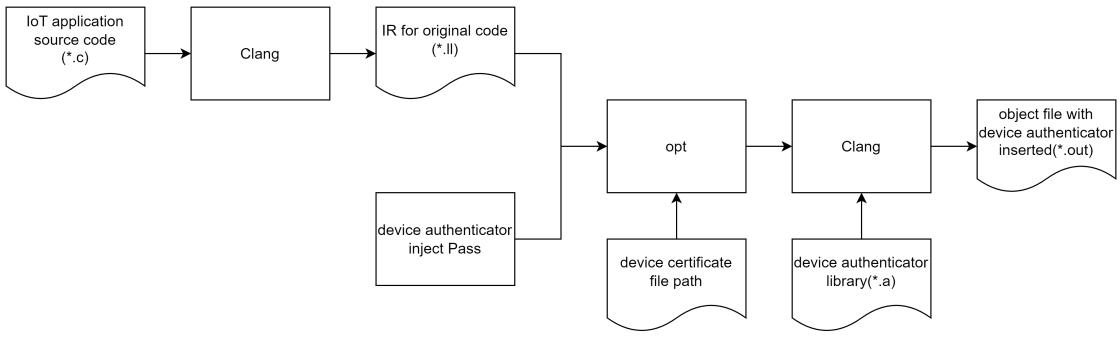


Fig. 7. Device Authenticator Insertion Structure

certificate file path) 입력이 필요하며, Pass는 장치 인증 함수 호출 코드 삽입 시 해당 입력값을 함수의 인자로 전달하도록 설정한다.

사용자가 개발한 IoT 응용 소스 코드를 IR로 변환하고 LLVM opt 도구를 이용하여 해당 IR 코드에 Pass를 적용한다. Pass를 적용한 IR 코드는 기존의 기능을 수행하기 전 장치 인증 함수를 호출하도록 변환된다. 이와 같이 자동으로 코드를 삽입하는 방법은 IoT 응용 소프트웨어 개발자에게 보안에 대한 부담을 경감시킨다.

Pass를 통해 변환된 코드로 빌드한 응용을 실행할 때는 인증용 키와 서명이 필요하므로, 코드 변환 시점에 키와 서명을 생성해야 한다. 키 쌍을 생성하고, 입력으로 받은 장치 고유값(디바이스 DNA)을 메시지로 하여 개인 키로 서명한다. 공개 키와 서명 파일은 Base64로 인코딩하며 소프트웨어를 실행할 장치에 저장하는데, 앞서 입력했던 파일 위치 정보에 해당하는 경로에 위치하도록 한다.

자동으로 삽입된 장치 인증기는 장치 고유값(디바이스 DNA)[6]을 이용하여 장비를 인증한다. 장치 고유값은 복제가 어려워 IoT 장치를 인증하는데 유용하다. Fig. 8.은 디바이스 바인딩이 적용된 IoT 장치 구조를 보인다.

장치 인증은 ECDSA[7] 공개 키와 서명 파일을 읽은 후 디바이스 DNA를 추출한 값에 해시 적용 후 서명 검증을 하여 장치를 인증한다.

Fig. 9.는 장치 인증 흐름도를 보인다. 프로그램이 실행되면 인증 함수가 호출되고 인증 함수는 먼저 장치 인증 코드 삽입 과정에서 입력한 설정 정보로 주어진 경로에서 공개 키와 서명을 읽는다. 이때 공개 키와 서명이 존재하지 않으면 프로그램은 강제 종료된다. 디바이스 DNA를 추출하여 해시를 적용하고 그 값을 메시지로 서명 검증을 시도한다. 서명 검증에 실패하면 프로그램을 강제 종료하고, 성공하면 인증 함수가 종료되고 응용의 원래 기능이 실행된다.

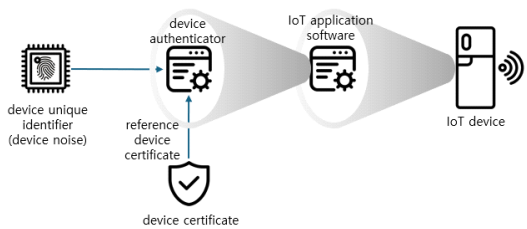


Fig. 8. Device Binding Applied IoT Device

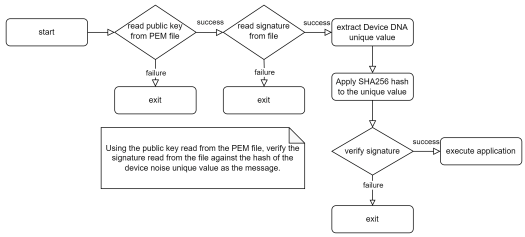


Fig. 9. Device Authentication Flowchart

IV. 실험 결과

본 논문에서 제안한 기술을 적용하였을 때 장치 인증이 정상 동작하는지 확인하고 성능 오버헤드를 측정하기 위해 실험을 수행하였다. 본 기술은 제한된 환경의 IoT 기기와 소프트웨어를 대상으로 적용할 수 있도록 설계되었다. Table 1.은 기술 적용이 가능한 환경을 나타내며, Table 2.는 실험 환경을 보인다.

장치 인증이 정상 동작하는지 확인하기 위해 인증 코드가 삽입된 실행 파일을 실행하여 결과를 비교하였다. Fig. 10.은 인증이 성공하여 프로그램이 정상적으로 실행되는 모습을 보인다. 반면 Fig. 11.은 디바이스 고유값이 일치하지 않는 경우의 실험 결과로, 프로그램이 강제 종료되는 것을 확인할 수 있다.

다음으로 인증 코드 삽입 시 성능 오버헤드 측정을 위해 장치 인증 함수의 실행 시간을 100회 측정

Table 1. Supported Platforms and Architectures

	Supported
CPU	x86, ARM
OS	Linux

Table 2. Environment Specifications

	Specification
Model	STM32MP157F-EV1
CPU	STM32MP 157 Arm-based dual Cotex-A7 800 MHz
RAM	2 * 4-Gbit DDR3L

```
test@deviceBindingTestBoard:~$ ./keyVerify -i ./
Verified Device
main function start
test@deviceBindingTestBoard:~$
```

Fig. 10. Experimental results (1)

```
test@deviceBindingTestBoard:~$ ./keyVerify -i ./
Failed to verify signature
return value of EVP_DigestVerifyFinal : 0
test@deviceBindingTestBoard:~$
```

Fig. 11. Experimental results (2)

하고 평균 소요 시간을 계산하였다. 실험 결과 장치 인증 평균 소요 시간은 52.9ms로 나타났다. Fig. 12.는 100회 인증 수행한 결과를 나타낸 그래프다.

실제로 프로그램에 인증 코드를 삽입하였을 때의 오버헤드 측정을 위해 간단한 알고리즘을 수행하는 프로그램에 기능을 적용하여 전후 실행 시간을 비교하였다.

기존 프로그램의 실행 시간과 상관없이 일정한 수준으로 시간이 증가하는 모습을 보이며, 평균 28.1ms의 오버헤드가 발생하였다. Fig. 13.은 기능 적용 전과 후 프로그램을 100회씩 실행하여 평균 소요 시간을 비교한 그래프다.

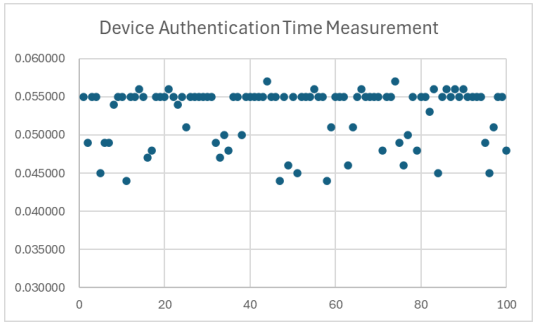


Fig. 12. Device Authentication Time Measurement

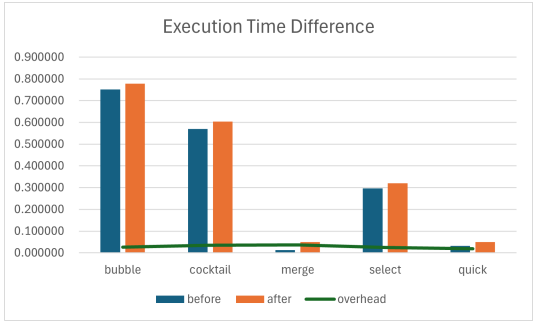


Fig. 13. Execution Time Difference Before and After

V. 결 론

최근 IoT 장치는 다양한 개인 및 기업 분야에서 활용되고 있고 서비스가 고도화됨에 따라 민감정보와 응용 소프트웨어 알고리즘 보호가 필요하다. 이에 따라 제품 개발사는 응용 소프트웨어 분석을 어렵게 하는 안티디버깅 기법을 제품에 적용하고 있는데, 이는 응용을 다른 IoT 장치에 설치하는 방법으로 회피할 수 있다.

본 논문은 안티디버깅을 회피하는 기법을 방어하기 위한 자동화된 디바이스 바인딩 기법을 제안한다. 제안하는 시스템은 디바이스 바인딩의 장치 인증 소스 코드를 추가하는데, LLVM Pass를 이용하여 자동으로 디바이스 바인딩 장치 인증 소스 코드를 삽입한다. 그리고 복제가 어려운 디바이스 DNA 기술을 이용하여 디바이스 고유값을 사용하였다. 이러한 기법을 적용한 소프트웨어는 지정된 장치에서만 실행이 가능하여, 공격자가 인서킷 에뮬레이션이나 JTAG 디버깅이 가능한 장치에서 프로그램을 실행하는 것을 막을 수 있다.

이와 같은 보안 기법은 단일 보안 기술만으로는 모든 경우를 방어하기 어렵다. 본 시스템의 코드를 분석하여 인증 코드를 제거하거나 변조하는 공격을 방어하기 위해서는 난독화와 무결성 검증 기술이 추가로 필요하다. 본 기술의 활용도를 높이기 위해 추후 다른 보호 기술에 대한 조사와 연구가 필요할 것으로 보인다. 또한 제안한 기술은 현재 제한적인 환경을 대상으로 설계되었다. 향후 연구에서는 다양한 환경에서 적용 가능한 방향으로 연구를 진행할 예정이다.

민감정보나 알고리즘 탈취 위협은 고도화되고 있으며, 이에 따라 다양한 보안 기법이 요구되고 있다. 본 논문에서 제안한 디바이스 바인딩 기법은 다양한 보안 기법과 더불어 사용할 때 악의적인 공격에 대응할 수 있는 솔루션이 되리라 기대된다.

References

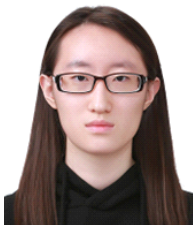
[1] Pradyumna Gokhale, Omkar Bhat, and Sagar Bhat, "Introduction to IOT," International Advanced Research Journal in Science, Engineering and Technology vol. 5, no. 1, pp. 41-44, Jan. 2018.

[2] Byung-Chul Oh, "A study on privacy protection in IoT environments," 2016 Naver Privacy Whitepaper, https://pri

vacy.naver.com/download/2016_NaverPrivacyWhitePaper_2.pdf, Dec. 2016.
〔3〕 Shang Gao and Qian Lin, "Debugging classification and anti-debugging strategies," Fourth International Conference on Machine Vision (ICMV 2011): Computer Vision and Image Analysis; Pattern Recognition and Basic Technologies, vol. 8350, pp. 729-734, Dec. 2011.
〔4〕 Shannon Davis, "SEMI anti-piracy server certification protocol for software license management reaches key milestone," https://www.semiconductor-digest.com/semi-anti-piracy-server-certification-protocol-for-software-license-management-reaches-key-milestone/, Aug. 2021.
〔5〕 LLVM compiler infrastructure, "LLVM Pass," https://llvm.org/docs/ WritingAnLLVMPass.html, Jun. 2024.
〔6〕 Dooho Choi, Yousung Kang, Mi-Kyung Oh, Sangjae Lee, and Taesung Kim, "The concept of device DNA for IoT security," Review of KIISC, 28(5), pp.15-19, Oct. 2018.
〔7〕 D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)," International Journal of Information Security, vol. 1, pp. 36-63, Aug. 2001.
〔8〕 O. Shwartz, Y. Mathov, M. Bohadana, Y. Elovici and Y. Oren, "Reverse engineering IoT devices: effective techniques and methods," IEEE Internet of Things Journal, vol. 5, no. 6, pp. 4965-4976, Dec. 2018.
〔9〕 Ye-jun Kim, Jeong-hyeon Gim, and Seung-joo Kim, "A study on systematic firmware security analysis method for IoT devices," Journal of the Korea Institute of Information Security & Cryptology, 31(1), pp. 31-49, Feb. 2021.
〔10〕 Sam Gallagher, "Introduction to JTAG and the test access port (TAP)," ALL ABOUT CIRCUITS:Technical Articles, https://www.allaboutcircuits.com/technical-articles/introduction-to-jtag-test-access-port-tap/, Nov. 2020.
〔11〕 MDSTECH, "TRACE32" https://trace32.com/trace32/solution/solution.php, Jun. 2024.
〔12〕 M.N. Gagnon, S. Taylor and A.K. Ghosh, "Software protection through anti-debugging," IEEE Security & Privacy, vol. 5, no. 3, pp. 82-84, Jun. 2007.
〔13〕 Ji-Hun Kim, Young-kil Kim, and Man-pyo Hong, "IP camera security using device unique identifier authentication," Proceedings of the Korean Institute of Information and Commucation Sciences Conference, pp. 82-85, May. 2018.
〔14〕 The LLVM Compiler Infrastructure, "LLVM," https://llvm.org/, Jun. 2024.
〔15〕 GCC the GNU Compiler Collection, "gcc," https://gcc.gnu.org/, Jun. 2024.

〈저 자 소 개〉

곽 동 규 (DongGyu Kwak) 정회원
2002년 2월: 서경대학교 응용수학과 학사
2004년 8월: 숭실대학교 컴퓨터학과 석사
2012년 2월: 숭실대학교 컴퓨터학과 박사
2022년 4월~현재: 쿤텍 주식회사 SS 연구개발팀 팀장
〈관심분야〉 임베디드 보안, 안티탬퍼링, 프로그래밍 언어, 컴파일러



김 선 우 (SunWoo Kim) 정회원
2024년 2월: 고려대학교 인공지능사이버보안학과 학사
2024년 2월~현재: 쿤텍 주식회사 SS 연구개발팀 연구원
〈관심분야〉 정보보호, IoT 보안

하 재 현 (JaeHyun Ha) 정회원
2022년 2월: 을지대학교 의료IT학과 졸업
2021년 12월~현재: 쿤텍 주식회사 SS 연구개발팀 선임 연구원
〈관심분야〉 정보보호, 소프트웨어 보안, 안티리버싱