Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.29, NO.4, Aug. 2019 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2019.29.4.739

정형 기법을 이용한 하드웨어 AES 모듈 백도어 탐색 연구*

박 재 현,† 김 승 주‡
고려대학교 정보보호대학원

Study of Hardware AES Module Backdoor Detection through Formal Method*

Jae-Hyeon Park,† Seung-joo Kim‡
Center for Information Security Technologies(CIST), Korea University

요 약

임베디드 기기의 보안성이 주요한 문제로 부상하고 있다. 관련된 문제 중 특히 공급망 공격은 국가 간의 분쟁으로 이어질 수 있어 심각한 문제로 대두되고 있다. 공급망 공격을 완화하기 위하여 하드웨어 구성요소, 특히 AES와 같은 암호 모듈에 대한 CC(Common Criteria) EAL(Evaluation Assurance Level) 5 이상 고등급 보안성 인증 및 평가가 필요하다. 고등급 보안성 인증 및 평가를 위하여 암호 모듈에 대한 은닉 채널, 즉 백도어를 탐지하는 것이 필요하다. 그러나 기존의 연구로는 암호 모듈 그 중 AES의 비밀 키를 복구시킬 수 있는 정보가 유출되는 백도어를 탐지하지 못하는 한계가 있다.

따라서 본 논문은 기존의 하드웨어 AES 모듈 백도어의 정의를 확장하여 개선시킨 새로운 정의를 제안하고자 한다. 또한, 이 정의를 이용하여 기존 연구가 탐지하지 못했던 백도어를 탐색하는 과정을 제시한다. 이 탐색 과정은 Verilog HDL (Hardware Description Language)로 표현된 AES 모듈을 정형 기법 도구인 모델 체커(Model Checker) NuSMV를 이용하여 검증하는 것으로 백도어를 탐색한다.

ABSTRACT

Security in embedded devices has become a significant issue. Threats on the sup-ply chain, like using counterfeit components or inserting backdoors intentionally are one of the most significant issues in embedded devices security. To mitigate these threats, high-level security evaluation and certification more than EAL (Evaluation Assurance Level) 5 on CC (Common Criteria) are necessary on hardware components, especially on the cryptographic module such as AES. High-level security evaluation and certification require detecting covert channel such as backdoors on the cryptographic module. However, previous studies have a limitation that they cannot detect some kinds of backdoors which leak the in-formation recovering a secret key on the cryptographic module.

In this paper, we present an expanded definition of backdoor on hardware AES module and show how to detect the backdoor which is never detected in Verilog HDL using model checker NuSMV.

Keywords: hardware backdoor, formal method, model checker

Received(06. 05. 2019), Modified(07. 01. 2019), Accepted(07. 01. 2019)

* "본 연구는 고려대 암호기술 특화연구센터 (UD170109ED)를 통한 방위사업청과 국방과학연구소의 연구비 지원으로 수행되었습니다.

† 주저자, pjh224@korea.ac.kr
‡ 교신저자, skim71@korea.ac.kr(Corresponding author)

I. 서 론

사물인터넷 기반의 임베디드 기기 시장은 단순히 규모만 증가하는 것이 아니라 그 분야도 점차 다양화되고 있다. 사물인터넷 관련 시장조사 기관인 IoT Analytics는 2018년에 진행된 사물인터넷 관련 프로젝트 중 상위 10개의 분야를 조사 및 발표하였다. IoT 기반의 임베디드 기기는 스마트 홈 시스템과 같이 사용자의 생활환경을 개선하는 분야 외에 Fig 1과 같이, 스마트 시티나 생산 분야, 에너지 기술 등 다양한 산업분야에서 활용되고 있다[1].

하지만 사물인터넷이 접목된 임베디드 기기가 다양한 분야에서 활용됨에 따라 여러 가지 문제점도 대두되었다. 그 중 공급망 공격은 현재 확대되고 있는 대표적인 위협이다. 공급망 공격은 제품이 생산되고 소비자에게 발송되는 과정에서 악의적인 공격자의 개입으로 발생되는 위협이다. 공격자가 공급망에 개입하여 구성요소에 악성 코드 등을 삽입하고, 그 결과 제품이 정보 유출이나 기능 정지와 같은 결함 및 문제가 발생할 수 있다[2].

공급망 공격은 다음과 같은 두 사기 사례를 들 수 있다. 2016년 보안업체 Kryptowire는 화웨이에서 제작한 스마트폰에서 펌웨어 수준의 백도어가 삽입되어 있는 것을 발표하였다. Kryptowire에 따르면 이 백도어는 스마트폰 사용자의 문자 메시지, 연락처, 통화 기록, 위치 정보 등을 72시간 주기로 중국 서버에 전송하였다. 해당 백도어는 펌웨어를 공급하는 회사에서 공급망 공격으로 삽입된 것으로 추정되었다[3]. 2019년 영국의 국가 기관인 HCSEC(Huawei Cyber Security Evaluation Centre)에서 발표한 보고서에 따르면, [3]의 발표가 있은 지 3년이 지났으나, 보안 조치는 미비하고,

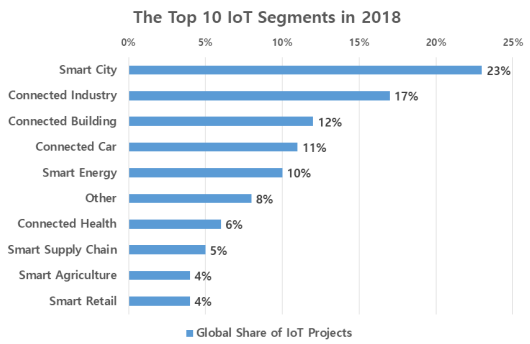

Fig. 1. The Top 10 IoT Segments in 2018

공급망 공격에 취약하여 개인정보 유출이 발생할 수 있으니 제품 사용을 자제할 것을 HCSEC는 권고하였다[4].

또 다른 예로 Bloomberg Businessweek에서 발표한 자료에 따르면 중국의 Supermicro사가 서버용 마더보드에 소형 마이크로 칩을 삽입하고 애플이나 아마존 등의 미국 기업에 납품하였다. 해당 마이크로칩은 마더보드에 전원이 공급되는 시기에 서버 내부 정보를 변조하여 정보 유출 및 기능 저하를 발생시켰다[5]. 이와 같이 하드웨어 수준의 공격은 소프트웨어 수준의 패치만으로 해결이 불가능하다.

상기 예시와 같은 공급망 공격은 지속적으로 확대되고 있는 실정이다. 2019년 시만텍에서 발간한 인터넷 보안 위협 보고서(ISTR, Internet Security Threat Report)에 따르면 공급망 공격은 2018년에는 전년도 대비 약 78%가 증가하여 지속적으로 위협이 되고 있음을 명시하였다[6].

이러한 공급망 공격을 방지하기 위하여 임베디드 기기에 대한 보안성 연구의 필요성은 커지고 있다. 공급망 공격을 방지하기 위하여 구성요소에 대한 인증 및 평가가 연구되며 수행되고 있다. 이 중 대표적인 방법론이 국제공통평가기준(Common Criteria, CC)이다. 국제공통평가기준은 개발 환경에서 해당 제품이 안전하게 개발되었는가를 평가하고 인증하기 위하여 활용되는 국제표준이다. 평가대상(Target Of Evaluation, TOE)의 신뢰도를 인증하기 위하여 평가자는 국제공통평가기준의 평가보증등급(Evaluation Assurance Level, EAL)을 활용하여 수준별 보안성을 보증한다[7].

평가보증등급을 획득하기 위하여, 설명서, 개발 환경 문서, 소스코드, 구현체 등 평가대상과 관련된 모든 자료가 필요하다. 특히 EAL 5 등급 이상의 고등급 평가에서 실시되는 은닉 채널(Covert Channel) 탐색을 위하여 소스코드 및 구현체는 반드시 제출되어야 한다. 은닉 채널은 백도어와 같이 사용자가 인지하지 못하게 내부 정보를 유출시킬 수 있는 통로를 의미한다. 은닉 채널은 높은 신뢰도를 인증받기 위하여 반드시 평가되어야 하는 요소로, 평가자들도 이를 탐색하기 위하여 다양한 기법들을 활용하고 있다[7].

공급망 공격, 특히 하드웨어 수준의 공격을 방지하기 위하여 국제공통평가기준을 적용하는 경우, 하드웨어 암호 모듈에 대한 은닉 채널이 존재하는 지를 확인해야 한다. 암호 모듈은 하드웨어의 보안성을 입

증하기 위한 핵심적인 기능이다. 그러므로 암호 모듈의 보안성이 우선적으로 입증되어야 한다. 또한, 하드웨어 암호 모듈에 은닉 채널이 존재한다면, 소프트웨어 패치로는 보완이 제한되기에 면밀한 분석이 필요하다.

하드웨어 암호 모듈 검증 연구 중 하드웨어 AES 모듈에 대한 백도어 탐색 연구는 현재까지 지속적으로 진행되었다. 기존 하드웨어 AES 모듈 백도어 탐색의 연구는 AES 비밀 키 유출에 한정되어 실시되었다. 그러나 AES 비밀 키 외에 암호화 과정에서 발생하는 중간 값을 통하여 공격자가 비밀 키를 복구할 수 있다[27],[28]. 그러므로 암호화 과정의 중간 연산 결과가 유출되는 백도어이 필요하지만, 기존의 연구로는 탐색이 제한된다.

따라서 본 논문은 백도어를 탐지하기 위하여 제시된 기존의 하드웨어 AES 모듈 백도어의 확장된 정의를 제안한다. 이 확장된 정의를 이용하여 AES 비밀 키 외에 암호화 과정에서 생성되는 중간 결과 값이 유출되는 백도어를 탐색하는 과정을 제시한다. Verilog HDL로 구현되어 있는 AES 모듈과 확장된 백도어 정의를 SMV 포맷으로 정형 명세하고 NuSMV에 입력하여 하드웨어 AES 모듈의 백도어를 탐색한다.

본 논문의 구성은 다음과 같다. 2장에서는 하드웨어 백도어의 정의와 정형기법을 이용한 하드웨어 AES 모듈의 백도어 탐색 연구에 대하여 서술하고, 3장은 백도어를 탐색하기 위하여 필요한 정보와 백도어를 탐지하는 프로세스에 대하여 서술한다. 4장은 해당 프로세스를 실시하는 과정과 그 결과에 대하여 서술하고, 마지막 5장에서는 결론과 향후 연구에 대하여 서술한다.

II. 관련 연구

2장은 하드웨어 AES 모듈의 백도어를 정형기법을 통해 탐색하기 위하여 하드웨어 백도어를 정의하고, 정형 기법을 활용한 하드웨어 AES 모듈의 백도어 탐색 연구에 대하여 서술한다.

2.1 하드웨어 백도어 정의

백도어의 정의는 백도어를 탐색하는 과정에서 백도어로 판단할 수 있는 기준과 특징을 구체화하기 위하여 내려졌다. 2000년에 Yin Zhang 외 1명이 작성한 논문에서는 백도어를 탐색하는 방법과 알고리즘에 대하여 서술하고 있다[8]. 해당 논문은 백도어를 컴퓨터 시스템에 인가되지 않은 접근을 실시하기 위하여 시스템 내부에 몰래(“surreptitiously”) 삽입된 매커니즘으로 정의하였다.

백도어에 대하여 정의하는 과정에서 Felix Schuster 외 1명은 백도어가 숨겨져 있는 것에 초점을 두고[9], Yan Shoshitaishvili 외 4명은 인가되지 않은 접근을 통하여 정보가 유출되는[10] 일부 특성에 집중하여 정의하였다. Ryan Williams 외 2명이 발표한 논문은 백도어에 대한 복잡도를 계산하기 위하여 수식을 혼합하여 백도어를 정의하였다[11].

2018년 Sam Thomas 외 1명은 백도어에 관련하여 정의, 부정 가능성(Deniability)과 탐지에 대해 조사한 논문을 발표하였다[12]. 해당 논문에서는 기존에 백도어와 관련된 연구를 종합하면서 구성요소와 특성을 반영하여 백도어의 정의를 내렸다.

본 논문에서 백도어의 정의는 다음과 같다. 백도어는 의도적으로 시스템 내부 정보를 유출시키기 위하여 포함된 추가 구조체로, 특정 값을 입력받아 백도어를 활성화시키는 Trigger와 백도어의 결과로 특정 정보를 유출시키는 Payload로 구성되어 있다.

하드웨어 백도어의 정의는 다음 연구들과 함께 진행되었다. Mohammad Tehranipoor 외 1명이 2011년 발표한 논문에서는 하드웨어 백도어를 물리적 특성, 활성화 방법에 대한 특성, 백도어의 기능에 대한 특성의 세 가지 기준으로 분류하였다[13]. 탐색 방법에 대하여 부채널 분석이나 Fuzzer와 같이 무작위 입력 값을 통해 백도어를 활성화시키는 방법에 대하여 서술하였다.

He Li 외 2명이 2016년에 발표한 논문에서 하드웨어 백도어은 특정 입력 값을 받았을 때 활성화되는 매커니즘 부분을 Trigger로, Trigger로 인하여 구동되거나 출력되는 결과를 Payload라고 명명하였다. 제안한 구조를 통하여 하드웨어 백도어를 탐지하고 방지하는 방법에 대하여 자체적인 프로세스를 제안하였다[14].

Nisha Jacob 외 3명은 하드웨어, 특히 IC (Integrated Circuit)의 개발과정 및 공급과정의 각 세부 단계에서 발생할 수 있는 백도어 삽입 방법 및 효용성에 대하여 연구하였으며[15], Julien Francq의 논문은 다양한 유형의 하드웨어 백도어를 탐색하는 방법과 복잡도 및 한계점을 서술하였다[16]. 다음 Table 2.는 이와 같은 관련 연구에서

정의한 하드웨어 백도어를 정리한 표이다.

Table 1. Definitions of Hardware Backdoor

Reference	Definition
[13]	mask theft and unauthorized overproduction as well as the insertion of malicious circuitry or alterations
[14]	A number of possible backdoors can be implanted in the design with varying activation mechanisms(triggers) and effects(payloads).
[15]	Hardware Trojans are built to gain access to secure devices and their data. The advantage of hardware Trojans is that access to a whole series or a batch of chips can be gained by manipulating the design or fabrication of an IC.
[16]	Malicious manipulation of hardware, referred to as Hardware Trogan Horses is quickly becoming an emerging threat.
[17]	It is desirable to design backdoors with rare triggers to avoid unintentional exposure during design validation or other benign testing
[18]	1) A functional input for an output is an input that is used by the specified normal functionality of this output 2) A trigger input for an HT-infected output is an input that is used in the condition under which the HT is activated. Note that, functional inputs can serve as trigger inputs for HTs. 3) The trigger condition is defined as the condition that the trigger inputs are driven with trigger values that activate the HT. 4) The nontrigger condition is defined as the condition that the trigger inputs are not driven by trigger values.
[20]	F : application I : set of input ports O : set of output ports ϕ : chip behaviour (analog or digital operation) f_n : subfunction (included in F) s_k : internal routing signal ϕ_n : valid internal state space i_n / o_n : inputs / outputs - $F = \{\phi, I, O\}$
	$F = f_1 \cup f_2 \cup f_3 \cup \cdots \cup f_N \cup f_t = \bigcup_{n=1}^{N} f_n \cup f_t$ $f_n = \{\phi_n, i_n, o_n, \phi_n\}$ $i_n \subseteq I \subseteq s_k,\ o_n \subseteq O \subseteq s_k,$ $n=1\cdots N,\ k=1\cdots K$ $f_t = \{\phi t, it, ot\}$: Backdoor $f_t = \{\}$ if ϕ_t =FALSE $\psi_t = f(i_n.i[\tau-m], o_n, i[\tau-m], \phi[\tau-m], \cdots)$
[21]	Backdoor consists of trigger & payload trigger : an optional part that monitors various signals and/or a series of events in the SoC, signals and events are rare conditions. payload : leak secret information / unauthorized access to a privilege system
[24]	R : register stores critical data O : output ports I : set of possible input patterns P : Design s_n : single bit of secret S_{N-1} : set of partial secrets $\exists i \in I, \forall s_0 \in \{0,1\}, s_{N-1} \in S_{N-1},$ $P \models (s_0 == 0) \vee (\neg s_0 == 0)$

상기 예시들을 참조하였을 때 하드웨어 백도어의 정의는 백도어의 구성요소인 Trigger와 Payload를 포함하여 공격자의 의도로 구현된 추가적인 회로 혹은 하드웨어 수준 코드로 정의할 수 있으며 대략적인 구성은 Fig 2.와 같다.

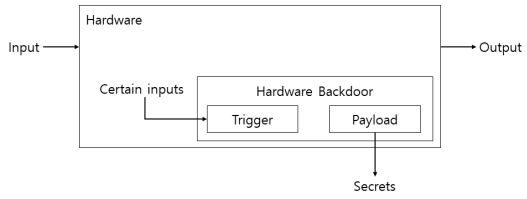

Fig. 2. Structure of Hardware Backdoor

2.2 정형기법을 이용한 하드웨어 AES 모듈의 백도어 탐색

2.1장에서 확인한 사항으로 백도어는 Trigger와 Payload로 구성되어 있는 추가 구조체이며 하드웨

어 백도어는 이를 하드웨어 수준에서 구현한 결과물을 의미한다. 기존의 하드웨어 AES 모듈의 백도어 탐색 연구는 백도어의 구성요소를 식별하는 것으로 진행되었다.

하드웨어 AES 모듈의 백도어 탐색을 위한 초기의 연구는 백도어를 활성화시키는 Trigger를 식별하는 것으로 백도어를 탐색하였다[17], [18], [19]. 해당 연구는 백도어는 전체 입력 값 중 드물게 발생하는 Rare Condition일 때 활성화된다고 가정하고, 이를 탐지하는 연구를 하였다. 하지만 백도어가 아닌 회로에 대한 False Positive가 많이 발생하고, Trigger가 항시 활성화되는 백도어를 탐색하지 못하였다.

이를 극복하기 위해 2010년대 중반 이후부터 최근의 연구는 Payload의 특징을 통해 백도어를 탐색하는 연구가 진행되고 있다. Adib Nahiyan 외 5명은 하드웨어 백도어를 Payload의 특성에 따라 2가지 유형으로 분류하고 각 유형별 백도어를 탐지할 수 있는 알고리즘을 제시하였다[21]. Payload의 경우, 특성에 대한 정형명세가 가능하여 Mainak Banga 외 1명[22]과 Xiaolong Guo 외 3명[23]의 연구와 같이 정형기법을 적용한 연구도 진행되고 있다.

하드웨어 AES 모듈 백도어를 정형 기법으로 탐색하기 위한 대표적인 연구로 Jeyavijayan Rajendran 외 3명은 공급망 공격에 대응하기 위해 SoC(System-on-Chip)의 설계를 정형 기법 도구를 이용하여 검증하고 하드웨어 백도어를 탐색하는 연구를 실시하였다[24]. 해당 연구는 백도어 탐색을 위해 모델 체커를 사용하였고, 이를 활용하기 위하여 백도어에 대한 정의를 재실시하였다.

해당 논문에서 백도어는 AES의 비밀 키의 유출 정도에 따라서 재 정의를 하였고, 해당 정의는 다음 Table 2.와 같다.

[24]의 연구는 최종적으로 AES의 비밀 키가 1-bit 이상 유출되는 경우와 AES의 비밀 키의 역이 1-bit 이상 유출되는 경우에 대하여 백도어로 판단하였다. 이 정의를 모델 체커에 입력하는 것으로 AES 비밀 키가 유출되는 백도어를 탐색하는 연구를 진행하였다.

기존의 정형 기법을 이용한 AES 백도어 탐색은 비밀 키의 유출을 탐색하는 연구가 진행되었다. 하지만 AES 암호화 과정에 영향을 끼칠 수 있는 정보는 비밀 키뿐만 아니라 Round Key 값과 같이 다양한 정보가 있다[29]. 해당 정보가 유출되는 경우, 비밀 키를 복구하거나 암호화에 영향을 끼칠 수 있으나 기존 연구는 키 이외의 값의 유출은 검증하지 않았다. 따라서 본 논문은 비밀 키 외에 AES 암호화 과정에 영향을 끼칠 수 있는 정보를 식별 및 정의하여 해당 정보가 유출되는 백도어를 탐색하는 연구를 진행하고자 한다.

Table 2. Hardware Backdoor Definition in [24]

Type	Property
Assumption	I : set of possible input patterns D : Hardware Design o : output port s : secret s_n : n-th single bit of secret S_{N-1} : (N-1)-bit length all possible values of secret
Initial	$\exists i \in I \ni D \vDash (s==0)$
Refinement 1	$\exists i \in I,\ s_x \in S_{N-1} \ni D \vDash (s_0==0)$
Refinement 2	$\exists i \in I,\ s_x \in S_{N-1} \ni D \vDash (s_0==0)$
Final	$\exists i \in I, \forall s_0 \in \{0,1\},$ $s_{N-1} \in S_{N-1}, \ni D \vDash (s_0==0) \vee (\neg s_0==0)$

III. 하드웨어 백도어 탐색

3장은 하드웨어 AES 모듈의 백도어를 탐색하기 위하여 백도어의 유출 대상 정보와 정보 유출을 정의하였다. 그리고 해당 정의와 모델 체커를 통하여 백도어를 탐색하는 프로세스에 대하여 서술한다.

3.1 백도어 탐색 정보

하드웨어 AES 모듈의 백도어를 통하여 유출될 수 있는 내부 정보를 비밀 키 외에도 존재한다. 본 논문에서 하드웨어 AES 백도어의 탐색을 위해 유출되는 정보의 특성을 고려한다. 기존의 백도어 탐색 연구는 Payload가 비밀 키의 일부 혹은 전체 값이 유출되는 경우로 한정하였다. 하지만 최근 연구에 따르면 AES 연산 간의 중간 값으로 비밀 키를 복구할 수 있는 연구 결과가 발표되고 있다. 이를 확인하였을 때, 하드웨어 AES 백도어는 비밀 키의 정보 외에 다른 정보에 대한 탐색이 필요하다.

[25]와 [26]의 논문은 AES 암호화 과정의 중간 값을 이용하여 비밀 키를 복구할 수 있는 수식화 부

채널 공격을 제시하였다. 해당 공격은 AES 암호화 연산, 특히 SubByte 연산과 MixColumn 연산 간에 발생하는 전력 소모량을 측정하고, 측정값들을 AES 논리식에 대입하여 비밀 키를 도출하는 방법이다. 해당 연구에서는 각 연산의 중간 결과 값을 부채널 공격을 통하여 획득하였다. 하지만 중간 결과 값을 출력하는 백도어가 있는 경우, [25], [26]의 공격을 통하여 AES 비밀 키를 복구하는 것이 가능하다.

[27], [28], [29]는 AES에 대한 공격 기법인 Square 공격, Impossible Differentials, Meet-in-the-Middle 공격을 통하여 비밀 키를 복구하는 연구를 실시하였다. 해당 연구들의 공통점은 특정 라운드 이하의 중간 연산 과정에서 각 라운드의 중간 입력 값, 중간 출력 값 그리고 라운드 키를 이용하여 비밀 키를 복구하는 공격을 실시하였다. [27]의 연구는 5라운드 이하의 값이 필요했으나, [29]의 연구에서는 8라운드의 중간 값을 통하여도 비밀 키를 복구한 결과를 제시하였다.

상기 연구를 참조하여 AES 백도어로써 AES의 암복호화 과정에 영향을 끼치거나 비밀 키를 획득하는데 기여할 수 있는 정보는 다음 Table 3.에 정리하였다. 해당 값들이 Payload를 통하여 외부로 유출되는 경우를 백도어가 발생하였다고 간주할 수 있다.

상기 유형의 정보를 유출시키는 백도어의 유무를 정형 기법을 통하여 탐색하기 위해서는 백도어와 상기 정보에 대한 정형화된 정의 2가지가 필요하다. 백도어 판단을 위한 정보 유출의 정의와 백도어의 유출 대상 정보의 정의이다.

백도어 판단을 위한 정보 유출의 정의는 다음과 같다. 하드웨어 AES 모듈을 $\mathbf{D}$, 유출 대상 정보를 $\mathbf{S}$, 백도어 포트를 $\mathbf{BP}$ 그리고 16-bit 길이의 S의 일부를 $\mathbf{s}$라고 명명하였을 때, 정보 유출의 정의는 수식 (1)과 같다.

$$\exists s \in S, D \vDash (BP == s) \vee (BP == \neg s) \quad (1)$$

[21]의 정의와 유사하나 s의 길이가 16-bit이며 그 이유는 다음과 같다. 본 논문에서 검증하고자 하는 하드웨어 AES 모듈은 128-bit를 암호화하는 대칭 키 블록 암호 알고리즘이다. NIST의 암호 키 관리 권고사항[30]에 따르면, 대칭 키 암호 알고리즘의 비밀 키는 최소 112-bit 이상의 보안강도를 유지해야 한다.

Table 3. Informations of AES

Information	Reference
Intermediate SubByte Input & Output Value Pair	[25], [26]
Intermediate Round Input / Output Value	[27], [28], [29]
Round Key	[27], [28], [29]

본 논문에서 지정한 유출 대상 정보들은 다양한 공격 기법을 통하여 비밀 키를 복구할 수 있는 정보이다. 예를 들어 유출 대상 정보가 16-bit 이상 노출되는 경우, AES 비밀 키를 16-bit 이상 복구할 수 있다. 이 경우 남은 AES 비밀 키는 112-bit 이다. 유출 대상 정보가 일부 유출되는 경우, 해당 과정을 반복하여 AES 비밀 키를 도출하거나 AES 보안강도를 저하시켜 전수조사 등으로 비밀 키를 도출할 수 있도록 유도하는 것이 가능하다. 따라서 본 논문에서는 유출 대상 정보의 전체가 유출되는 경우 외에 일부, 16-bit 이상이 유출되는 경우도 포함하여 백도어로 유출 가능하다고 판단하였다.

정리하자면 본 논문에서 '백도어를 통한 정보 유출'로 간주할 수 있는 속성은 비밀 키의 보안강도를 NIST 권고인 112-bit 이하로 저하시킬 수 있는 정보의 유출과 유출 대상 정보를 역연산한 결과가 일부 유출되어도 '백도어를 통한 정보 유출'로 간주하였다.

백도어의 유출 대상 정보의 정의는 Table 2.에 명시한 정보에 대하여 정형적으로 명세된 정의가 필요하다. 기존의 연구는 비밀 키의 유출에 대해서만 식별하였으므로, 비밀 키를 인지하고 있는 가정 사항으로 증명이 가능하다. 하지만 Table 3.의 정보는 중간 Round의 연산 결과의 유출을 탐색하므로, 백도어 검증 간 유출되는 값의 비교를 위하여 각 정보별 정의가 필요하다.

Round 연산 시 입력하는 메시지 $\mathbf{I}$, 키 스케줄링을 위하여 입력되는 Round Key $\mathbf{SK}$, 키 스케줄링을 위하여 사용되는 상수값 $\mathbf{rc}$로 명시한다. 그리고 k번째 Round에서 사용되는 Round Key $\mathbf{RK_k}$, 8-bit 단위 치환 함수(S-box) $\mathbf{S()}$, 최종 Round 연산 함수 $\mathbf{FR()}$, 1~9회 Round 연산 함수 $\mathbf{OR()}$, bit 연결자 연산자 $::$, bit 위치 식별 연산자 $[:]$로 명명하였을 때, 각 정보의 정의는 다음과 같다.

중간 SubByte 연산 출력값(ItS)

$$ItS = S(I[127:120]) :: S(I[119:112]) :: \ldots :: S(I[15:8]) :: S(I[7:0]) \quad (2)$$

Table 4. Comparison the definitions of Existing Studies with the definitions of this paper

	Existing Studies	This Paper
Leakage of Information	If Secret Information S or inverted S is leaked more than 1-bit, information leakage has occurred and there is a hardware backdoor. (Table 2.'s Final)	If Secret Information S or inverted S is leaked more than 16-bit, information leakage has occurred and there is a hardware backdoor. (Equation (1)
Information to be leaked	AES Secret Key (Assumption, not defined)	Intermediate SubByte Output (Equation (2))
		Intermediate Round Output (Equation (3), (4))
		Round Keys (Equation (5)

중간 Round 출력값(ItR)

$$1 \le k \le 9, ItR = OR(I, RK_k) \quad (3)$$

최종 Round 출력값(FnR)

$$FnR = FR(I, RK_{10}) \quad (4)$$

중간 Round Key 출력값(ItRK)

$$\begin{aligned} ItRK[127:96] &= K[127:96] \oplus rc \oplus \\ &\quad (S(K[119:96]) :: S(K[127:120])) \\ ItRK[95:64] &= K[95:64] \oplus ItRK[127:96] \\ ItRK[63:32] &= K[63:32] \oplus ItRK[95:64] \\ ItRK[31:0] &= K[31:0] \oplus ItRK[63:32] \end{aligned} \quad (5)$$

Table 4.는 기존에 하드웨어 AES 모듈 백도어에 대한 정의와 본 논문에서 정의하는 백도어에 대한 비교표이다. 기존의 연구는 AES 비밀 키 유출에 대해서만 백도어 탐색을 하였으나 본 논문은 유출 대상 정보의 범위를 확대하고 정형 기법을 활용하기 위해 정의하였다. 본 논문에서 탐색하고자 하는 백도어는 AES 비밀 키의 보안강도를 저하시켜 공격자가 비밀 키를 복구 가능하도록 유도하는 정보를 유출시킨다. 이 보안강도의 기준은 NIST의 112-bit 이하로, 유출되는 데이터의 양은 16-bit 이상으로 정의된다.

본 연구는 이러한 정의들을 통하여 하드웨어 AES 모듈의 백도어를 모델 체커로 탐색하는 연구를 진행하였다.

3.2 백도어 탐색 프로세스

하드웨어 백도어의 탐색은 Verilog HDL로 구현된 하드웨어 AES와 백도어의 Property를 모델 체커에 입력하여 검증하는 과정을 통하여 실시된다. Fig 3.과 같은 프로세스로 하드웨어 백도어를 탐색한다.

분석 대상인 백도어가 포함된 AES 모듈은 Verilog HDL로 구현되었으며, 3.1장에서 식별한 탐색 대상 정보를 유출시키는 백도어는 Trust Hub[31]와 같은 저명한 단체에서 제공하는 벤치마크가 존재하지 않아 별도로 구현하였다.

검증을 위한 모델 체커는 NuSMV[32]를 사용하였다. NuSMV는 카네기 멜론 대학을 포함한 4개 대학 및 기업에서 개발한 오픈 소스 모델 체커이다. NuSMV는 최초의 모델 체커인 SMV(Symbolic Model Verifier)의 지원이 종료되어 기존에 SMV를 이용한 연구를 지속시키기 위하여 개발되었다. NuSMV는 SMV 도구의 기능을 계승하고, 이전보다 확장하여 현재까지도 활용되는 모델 체커이다. NuSMV는 Verilog HDL과 유사하게 모듈 단위로 명세를 실시하고 모델을 생성하여 하드웨어의 검증에 용이하게 활용할 수 있다.

본 논문에서는 구현된 AES 모듈 및 백도어를 SMV 포맷으로 변경하고, 백도어의 특징을 검출하기 위한 Property를 수식형태로 명세한다. AES 모듈의 SMV 포맷 파일에 대하여 백도어 Property를 NuSMV로 검증하는 것으로 백도어의 존재 유무를 판단한다.

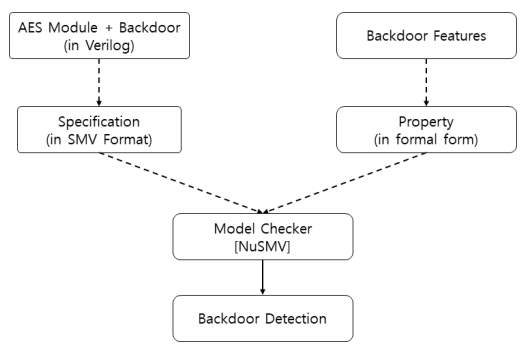

Fig. 3. AES Module Backdoor Detection Process

NuSMV를 이용해 AES 모듈의 모델을 생성하는 과정에서 유의할 사항으로 State Explosion이 있다. 모델 체커는 내부 변수의 변경에 따라 발생하는 State가 전환되는 Path나 Computation Tree 형태로 모델을 자동으로 생성하고 특정 Property를 검증한다. 이 과정에서 내부 변수의 경우의 수나 변수의 개수가 많을 경우, State의 수가 증가하여 모델 생성이 불가능할 수 있다. 따라서 NuSMV는 State가 전환되는 Transition 횟수를 제한한 BMC(Bounded Model Check) 기능을 지원하여 State Explosion을 해결한다. 본 논문에서는 Bound 범위를 5회 이하로 제한하여 검증을 실시하였다.

IV. 실험 결과

수식 (2) ~ (5)의 각 정보에 대한 백도어를 식별하기 위하여 수식 (1)의 정보 유출을 활용하여 SMV 포맷에서 활용 가능한 정형 명세를 실시하여야 한다. 수식 (2) ~ (5)를 SMV 포맷으로 정형 명세한 결과는 Table 5.과 같다.

Fig 4.의 코드를 NuSMV 상에서 검증하기 위한 SMV 파일 형태로 변환한다. 해당 과정에서 유출 포트인 wire 변수는 검증 단계에서 활용하기 위해 State의 변수로 변환한다. Fig 5.는 Fig 4.를

```
module aes_128(clk, state, key, out);
    input          clk;
    input  [127:0] state, key;
    output [127:0] out;
    reg    [127:0] s0, k0;
    wire   [127:0] s1, s2, s3, s4, s5, s6, s7, s8, s9,
                   k1, k2, k3, k4, k5, k6, k7, k8, k9,
                   k0b, k1b, k2b, k3b, k4b, k5b, k6b, k7b, k8b, k9b;

    always @ (posedge clk)
      begin
        s0 <= state ^ key;
        k0 <= key;
      end

    expand_key_128
        a1 (clk, k0, k1, k0b, 8'h1),
        a2 (clk, k1, k2, k1b, 8'h2),
        a3 (clk, k2, k3, k2b, 8'h4),
        a4 (clk, k3, k4, k3b, 8'h8),
        a5 (clk, k4, k5, k4b, 8'h10),
        a6 (clk, k5, k6, k5b, 8'h20),
        a7 (clk, k6, k7, k6b, 8'h40),
        a8 (clk, k7, k8, k7b, 8'h80),
        a9 (clk, k8, k9, k8b, 8'h1b),
       a10 (clk, k9,   , k9b, 8'h36);

    one_round
        r1 (clk, s0, k0b, s1),
        r2 (clk, s1, k1b, s2),
        r3 (clk, s2, k2b, s3),
        r4 (clk, s3, k3b, s4),
        r5 (clk, s4, k4b, s5),
        r6 (clk, s5, k5b, s6),
        r7 (clk, s6, k6b, s7),
        r8 (clk, s7, k7b, s8),
        r9 (clk, s8, k8b, s9);

    final_round
        rf (clk, s9, k9b, out);
endmodule
```

Fig. 4. Part of AES Module Verilog HDL Code

```
MODULE aes_128
VAR
        state   :       word[128];
        key     :       word[128];
        out     :       word[128];

        s0      :       word[128];
        s1      :       word[128];
        s2      :       word[128];
        s3      :       word[128];
        s4      :       word[128];
        s5      :       word[128];
        s6      :       word[128];
        s7      :       word[128];
        s8      :       word[128];
        s9      :       word[128];

        k0a     :       word[128];
        k1a     :       word[128];
        k2a     :       word[128];
        k3a     :       word[128];
        k4a     :       word[128];
        k5a     :       word[128];
        k6a     :       word[128];
        k7a     :       word[128];
        k8a     :       word[128];
        k9a     :       word[128];

        ex1     :       expand_key(key, 0uh32_01000000, k0a);
```

Fig. 5. Part of AES Module SMV Format

SMV 포맷으로 변경한 결과의 일부이다.

백도어의 유무를 판별할 때, AES 모듈이 동작하는 모든 상황에서 1회 이상 백도어 포트로 Table 3.의 정보가 유출되는 경우, 백도어를 탐색할 수 있어야 한다. 그러므로 이를 위하여 AES 모듈이 운용되는 Path 상에서 특정 조건이 만족하는지 여부를 확인할 수 있는 LTL(Linear Tempoal Logic)을 사용하여 백도어의 유무를 탐색한다.

LTL에 대한 검증 기능 중 F(Finally) 연산자를 이용한다. F 연산자는 운용하는 과정 중에서 특정 조건이 1회 이상 만족하는 여부를 확인하는 연산자이다. Table 3.의 조건들이 1회 이상 만족하는 경우, 백도어가 발생하였다고 탐색할 수 있다. 해당 명세를 모듈 단위로 구현하여 검증한 결과는 Fig 6.과 같다.

```
NuSMV > go_bmc
NuSMV > bmc_pick_state -i

***************  AVAILABLE STATES  *************

================= State =================
0) -------------------------
   state = 0ud128_889627103061277028662417274331420l5
   key = 0ud128_484630065291375013484742621849375427 91
   out = 0ud128_0
   s0 = 0ud128_48374043858445761544134292830676302456
                          :
NuSMV > check_ltlspec_bmc -k 5
-- no counterexample found with bound 0
-- no counterexample found with bound 1
-- no counterexample found with bound 2
-- no counterexample found with bound 3
-- no counterexample found with bound 4
-- no counterexample found with bound 5
```

Fig. 6. Result of Verification through Fig 5. Code

Table 5. Specification in SMV Format for Definitions to detect backdoor

Definition	Formal Form for Detecting in SMV Format
Equation (2)	ItS = S(I[127:120])::S(I[119:112]):: … ::S(I[15:8])::S(I[7:0]) BP = ItS[127:112] ∨ BP = ¬ItS[127:112] ∨ BP = ItS[111:96] ∨ BP = ¬ItS[111:96] ∨ BP = ItS[95:80] ∨ BP = ¬ItS[95:80] ∨ BP = ItS[79:64] ∨ BP = ¬ItS[79:64] ∨ BP = ItS[63:48] ∨ BP = ¬ItS[63:48] ∨ BP = ItS[47:32] ∨ BP = ¬ItS[47:32] ∨ BP = ItS[31:16] ∨ BP = ¬ItS[31:16] ∨ BP = ItS[15:0] ∨ BP = ¬ItS[15:0]
Equation (3)	1≦k≦9, ItR = OR(I, RKk) BP = ItR[127:112] ∨ BP = ¬ItR[127:112] ∨ BP = ItR[111:96] ∨ BP = ¬ItR[111:96] ∨ BP = ItR[95:80] ∨ BP = ¬ItR[95:80] ∨ BP = ItR[79:64] ∨ BP = ¬ItR[79:64] ∨ BP = ItR[63:48] ∨ BP = ¬ItR[63:48] ∨ BP = ItR[47:32] ∨ BP = ¬ItR[47:32] ∨ BP = ItR[31:16] ∨ BP = ¬ItR[31:16] ∨ BP = ItR[15:0] ∨ BP = ¬ItR[15:0]
Equation (4)	k=10, FnR = FR(I, RKk) BP = FnR[127:112] ∨ BP = ¬FnR[127:112] ∨ BP = FnR[111:96] ∨ BP = ¬FnR[111:96] ∨ BP = FnR[95:80] ∨ BP = ¬FnR[95:80] ∨ BP = FnR[79:64] ∨ BP = ¬FnR[79:64] ∨ BP = FnR[63:48] ∨ BP = ¬FnR[63:48] ∨ BP = FnR[47:32] ∨ BP = ¬FnR[47:32] ∨ BP = FnR[31:16] ∨ BP = ¬FnR[31:16] ∨ BP = FnR[15:0] ∨ BP = ¬FnR[15:0]
Equation (5)	ItRK = (K[127:96] ⊕ rc ⊕ (S(K[119:96])::S(K[127:120]))) :: (K[95:64] ⊕ K[127:96] ⊕ rc ⊕ (S(K[119:96])::S(K[127:120]))) :: (K[63:32] ⊕ K[95:64] ⊕ K[127:96] ⊕ rc ⊕ (S(K[119:96])::S(K[127:120]))) :: (K[31:0] ⊕ K[63:32] ⊕ K[95:64] ⊕ K[127:96] ⊕ rc ⊕ (S(K[119:96])::S(K[127:120]))) BP = ItRK[127:112] ∨ BP = ¬ItRK[127:112] ∨ BP = ItRK[111:96] ∨ BP = ¬ItRK[111:96] ∨ BP = ItRK[95:80] ∨ BP = ¬ItRK[95:80] ∨ BP = ItRK[79:64] ∨ BP = ¬ItRK[79:64] ∨ BP = ItRK[63:48] ∨ BP = ¬ItRK[63:48] ∨ BP = ItRK[47:32] ∨ BP = ¬ItRK[47:32] ∨ BP = ItRK[31:16] ∨ BP = ¬ItRK[31:16] ∨ BP = ItRK[15:0] ∨ BP = ¬ItRK[15:0]

Fig 6.에서 'no counterexample found'는 해당 수식이 False인 경우에 대하여 반례를 확인할 수 없다는 의미로, 해당 식이 True임을 의미한다. 즉, Table 3.의 백도어로 인한 정보 유출이 1회 이상 발생할 수 있음을 알 수 있고, 해당 식을 통하여 백도어를 식별하는 것이 가능하다.

상기 검증 과정의 확장으로 NuSMV의 기능을 이용하여 백도어가 발생하였을 시 AES 모듈의 구동을 확인할 수 있다. 원리는 NuSMV가 수식을 검증하는 과정에서 반례가 존재하는 경우, 반례에 대한 변수 값들을 도출해 주는 기능을 응용하는 것이다.

Table 3. 수식의 부정형을 입력하여 검증을 실시한다. 이는 해당 모듈은 백도어가 절대로 발생하지 않는다는 의미로 검증을 실시하는 것으로, 반례가 발생하는 것은 백도어가 존재한다는 의미와 같다. 해당 원리를 이용하여 백도어를 탐지하고 유출 상황에서 AES의 구동 및 정확히 어떤 값이 유출되는 가의 예시는 Fig 7.과 같다.

Table 6. Result of Detecting Backdoor

Definitions	Result
Equation (2)	Detectable
Equation (3)	Detectable
Equation (4)	Detectable
Equation (5)	Detectable

Table 6.는 수식 (2) ~ (5)의 각 유형에 대한 백도어 탐색 결과이며, BMC 모델에서 5회 이하 Bound에서 백도어를 탐색하는 것이 가능하다.

V. 결론 및 향후 연구

사물인터넷의 확대로 임베디드 기기의 보급이 활성화되는 가운데, 공급망 공격은 가장 중요한 보안위협으로 부상하고 있다. 공급망 공격으로 인하여 하드웨어 수준의 결함이나 백도어는 소프트웨어 수준의 패치로 완전히 개선하는 것이 불가능하다. 그러므로 하드웨어 모듈 단위의 검증이 필요하다.

본 논문은 하드웨어 AES 모듈에 대해 정형검증을 통하여 백도어의 유무를 탐색하는 연구를 진행하였다. 기존의 AES 모듈의 백도어는 비밀 키의 유출만을 백도어로 취급하였으나, AES에 대한 알고리즘 취약점, 부채널 공격 등으로 Round를 연산한 결과

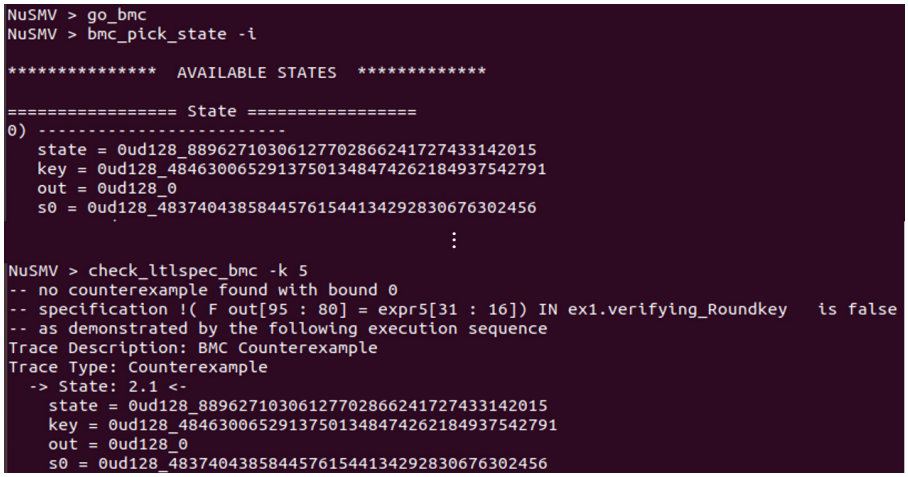

Fig. 7. Result of Verification to check module variables

를 통해서 비밀 키 추출 및 AES에 영향을 줄 수 있다. 따라서 본 논문에서는 비밀 키 외의 다른 정보에 대한 백도어를 정의하고 해당 특성을 모델 체커에서 활용하기 위한 Property로 명세하였다. 이를 통해 기존의 AES 백도어 연구에서 확인하지 못한 백도어를 모델 체커를 통한 정형 기법으로 탐색 가능하다.

본 연구의 한계로 첫째 탐지하고자 하는 백도어의 다양성이 부족한 것이다. 서술하였듯이 해당 유형의 백도어는 현재 저명한 단체에서 제공하고 있지 않아서 적절한 벤치마크로서 실험여부를 명백히 증명하기가 제한된다. 차후 연구로 해당 유형의 백도어를 다양한 방법으로 구현하고 이를 검증하는 과정이 추가적으로 필요하다.

두 번째로 모델 체커의 근본적인 한계로 인해 복잡한 수식이나 함수에 대해서는 검증이 제한된다. 또한, 하드웨어가 구현된 Verilog HDL 코드를 SMV 포맷으로 변경할 때, 지원하는 도구[33]의 미비함으로 일치성 검증의 문제가 발생할 수 있다.

[34]는 Theorem Prover와 모델 체커를 병행하여 사용하는 프로세스를 제안하여 상호 도구 간의 문제를 개선하고자 하였다. 이와 같이 향후 연구는 모델 체커와 Theorem Prover를 결합하는 것으로 하드웨어 모듈에 대한 정형 검증 방법론을 개선하는 것과 실제 Verilog HDL 구현체와 SMV 포맷 간의 일치성 검증에 대한 연구가 남아있다.

References

[1] IoT Analytics, "The Top 10 IoT Segments in 2018 - based on 1,600 real IoT projects", https://iot-analytics.com/top-10-iot-segments-2018-real-iot-projects/, Feb 2018.

[2] Iris Heckmannm, Tina Comes, Stefan Nickel, "A Critical review on supply chain risk - Definition, measure and modeling", Omega 52, April 2015.

[3] Matt Apuzzo, Michael S. Schmidt, "Serect Back Door in Some U.S. Phones Sent Data to China, Analyst Say", https://www.nytimes.com/2016/11/16/us/politics/china-phones-software-security.html?smid=nytcore-ipad-share&smprod=nytcore-ipad, The New York Times, Nov 2016.

[4] Huawei Cyber Security Evaluation Centre(HCSEC), "Huawei cyber security evaluation centre oversight board: annual report 2019", https://www.gov.uk/government/publications/huawei-cyber-security-evaluation-centre-oversight-board-annual-report-2019, HCSEC, March 2019

[5] Jordan Robertson, Michael Rilley,

Bloomberg Businessweek, "The Big Hack: How China Used a Tiny Chip to Infiltrate U.S. Companies", https://www.bloomberg.com/news/features/2018-10-04/the-big-hack-how-china-used-a-tiny-chip-to-infiltrate-america-s-top-companies?srnd=2018-the-big-hack, Oct 2018.

〔6〕 Symantec, "Internet Security Threat Report (ISTR) 2019", https://www.symantec.com/security-center/threat-report, Feb 2019.

〔7〕 Common Criteria, "Common Criteria for Information Technology Security Evaluation Part 3 : Security assurance components Version 3.1 Revision 5", April 2017.

〔8〕 Yin Zhang, Vern Paxson, "Detecting Backdoor", USENIX Security Symposium, Aug 2000.

〔9〕 Felix Schuster, Thorsten Holz, "Towards Reducing the Attack Surface of Software Backdoors", 2013 ACM SIGSAC conference on Computer & communications security, pp. 851-862, CCS'13, Nov 2013.

〔10〕 Yan Shoshitaishvili, Ruoyu Wang, Christophe Hauser, Christopher Kruegel, Giovanni Vigna, "Firmalice - Automatic Detection of Authentication Bypass Vulnerabilities in Binary Firmware", NDSS, Feb 2015.

〔11〕 Ryan Williams, Carla P. Comes, Bart Selman, "Backdoor to typical case complexity", IJCAI(International Joint Conference on Artificial Intelligence), volume 3, pp. 1173-1178, Aug 2003.

〔12〕 Sam Thomas, Aurélien Francillon, "Backdoors: Definition, Deniability and Detection", International Symposium on Research in Attacks, Intrusions and Defenses, pp. 92- 103, Springer, Sep 2018.

〔13〕 Mohammad Tehranipoor, Farinaz Koushanfar, "A Survey of Hardware Trojan Taxonomy and Detection", IEEE Design & Test of Computers, volume 27, pp.10-25, IEEE, Feb 2010.

〔14〕 He Li, Qiang Liu, Jiliang Zhang, "A survey of hardware Trojan threat and defense", Integration the VLSI journal, volume 55, pp.426-437, ELSEVIER, Sep 2016.

〔15〕 Nisha Jacob, Dominik Merli, Johann Heyszl, Georg Sigl, "Hardware Trojans: current challenges and approaches", IET Computers & Digital Techniques, volume 8, pp. 264-273, IET, Nov 2014.

〔16〕 Julien Francq, Florian Frick, "Introduction to hardware Trojan detection methods", 2015 Design, Automation & Test in Europe Conference & Exhibition(DATE), pp. 770-775), EDA Consortium, March 2015.

〔17〕 Adam Waksman, Matthew Suozzo, Simha Sethumadhavan, "FANCI: Identification of Stealthy Malicious Logic Using Boolean Functional Analysis", 2013 ACM SISGSAC conference on Computer & communications security, pp.697-708, CCS'13, Aug 2013.

〔18〕 Jie Zhang, Feng Yuan, Linxiao Wei, Yannan Liu, Qiang Xu, "VeriTrust: Verification for Hardware Trust", IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems, volume 34, pp.1148-1161, IEEE, July 2015.

〔19〕 Xuehui Zhang, Mohammad Tehranipoor, "Case Study: Detecting Hardware Trojans in Third-Party Digital IP Cores", 2011 IEEE International Symposium on Hardware-Oriented Security and Trust(HOST), pp. 67-70, IEEE, June 2011.

[20] Michael Rathmair, Florian Schupfer, Christian Krieg, "Applied Formal Methods for Hardware Trojan Detection", 2014 IEEE International Symposium on Circuits and Systems(ISCAS), pp.169-172, IEEE, July 2014.

[21] Adib Nahiyan, Mehdi Sadi, Rahul Vittal, Gustavo Contreras, Domenic Forte, Mark Tehranipoor, "Hardware Trojan Detection through Information Flow Security Verification", 2017 IEEE Internatioanl Test Conference (ITC), pp.1-10, IEEE, Oct 2017.

[22] Mainak Banga, Michael S. Hsiao, "Trusted RTL: Trojan Detection Methodology in Pre-Silicon Designs", 2010 IEEE International Symposium on Hardware-Oriented Security and Trust(HOST), pp. 56-59, IEEE, June 2010.

[23] Xiaolong Guo, Raj Gautam Dutta, Prabhat Mishra, Yier Jin, "Automatic RTL-to-formal code converter for IP security formal verification", 2016 17th International Workshop on Microprocessor and SOC Test and Verification (MTV), pp. 35-38. IEEE, Dec 2016.

[24] Jeyavijayan Rajendran, Arunshankar Muruga Dhandayuthapany, Vivekananda Vedula, Ramesh Karri, "Formal Security Verification of Third Party Intellectual Property Cores for Information Leakage", 2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID), pp.547-552, IEEE, Jan 2016.

[25] Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, "Algebraic Side-Attacks on the AES: Why Time alse Matter in DPA", International Workshop on Cryptographic Hardware and Embedded Systems, pp. 97-111, Springer, Sep 2009.

[26] Mohamed Saied Emam Mohamed, Stanislav Bulygin, Michael Zohner, Annelie Heuser, Michael Walter, Johannes Buchmann, "Improved Algebraic Side Channel Attack on AES", 2012 IEEE International Symposium on Hardware-Oriented Security and Trust(HOST), pp. 146-151), IEEE, June 2012.

[27] Niels Ferguson, John KelseyStefan Lucks, Bruce Schneier, Mike Stay, David Wagner, Doug Whiting, "Improved Cryptanalysis of Rijindael", International Conference on Information Security and Cryptology, pp. 39-49, Springer, Dec 2001.

[28] Eli Biham, Nathan Keller, "Cryptoanalysis of Reduced Variants of Rijndael", 3rd AES Conference, volume 230, April 2000.

[29] Huseyin Demirci, Ali Aydin Selcuk, "A Meet-in-the-Middle Attack on 8-Round AES", International Workshop on Fast Software Encryption, pp. 116-126, Springer, Feb 2008.

[30] Elaine Barker, "Recommendation for key management part 1: General (revision 3). NIST special publication, vol. 800-57, pp. 1-147, Jan 2016.

[31] Trust-hub, "Trojan Benchmarks", https://www.trust-hub.org/benchmarks/trojan, 2018.

[32] Roberto Cavada, Alessandro Cimatti, Charles Arthur Jochim, Gavin Keighren, Emanuele Olivetti, Marco Pistore, Marco Roveri, Andrei Tchaltsev, "NuSMV 2.6 User Manual", CMU and ITC-irst, Oct 2015.

[33] Ahmed Irfan, Alessandro Cimatti, Alberto Griggio, Marco Roveri, Roverto Sebastiani, "Verilog2SMV: a

tool for word-level verification", Proceedings of the 2016 Conference on Design, Automation & Test in Europe, pp. 1156-1159, EDA Consortium, March 2016.

[34] Xiaolong Guo, Raj Gautam Dutta, Prabhat Mishra, Yier Jin, "Scalable SoC trust verification using integrated theorem proving and model checking", 2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pp. 124-129. IEEE, May 2016.

〈저자소개〉

박 재 현 (Jae-Hyeon Park) 학생회원
2014년 8월: 홍익대학교 컴퓨터공학과 공학사
2017년 9월~현재: 고려대학교 정보보호대학원 석사과정
〈관심분야〉 보안성 평가/인증, 위협 모델링, 정형기법

김 승 주 (Seungjoo Kim) 종신회원
1994년~1999년: 성균관대학교 정보공학과(학사, 석사, 박사)
1998년~2004년: 한국인터넷진흥원(KISA) 팀장
2004년~2011년: 성균관대학교 정보통신공학부 부교수
2011년~현재: 고려대학교 사이버국방학과/정보보호대학원 정교수
2017년~현재: 고려대학교 사이버무기시험평가연구센터(CW-TEC) 부센터장
2004년~현재: 한국정보보호학회 이사
2007년: 국가정보원장 국가사이버안전업무 유공자 표창
2010년: 방송통신위원회 정보통신망 침해사고 민관합동조사단 위원
2011년~현재: (사)화이트해커연합 HARU 및 국제해킹대회 SECUINSIDE 설립자 및 이사
2012년: 선관위 디도스 특별검사팀 자문위원
2014년~2015년: 육군사관학교 초빙교수
2014년~2016년: 다음카카오 프라이버시 정책 자문위원회 위원
2015년~현재: 방위사업청 방산기술보호 자문관
2016년~2018년: 개인정보분쟁조정위원회 위원
2016년~현재: 산업통상자원부 전략물자기술 자문위원
2016년~현재: 한국카카오뱅크 정보보호부문 자문교수
2017년~현재: 국방보안연구소 정보보호분야 자문위원
2017년~현재: 여신금융협회 신용카드 단말기 시험 인증위원회 위원
〈관심분야〉 보안공학 및 SDL, 위협 리스크 모델링, 보안성 평가/인증, 암호학, Usable Security