Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.28, NO.2, Apr. 2018 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2018.28.2.287

BIL 비트스트림 역공학 도구 분석 연구*

윤 정 환,[1†] 서 예 지,[1] 김 훈 규,[2] 권 태 경[1‡]
[1]연세대학교 정보보호연구실, [2]국방과학연구소

A Study on the BIL Bitstream Reverse-Engineering Tool-Chain Analysis*

Junghwan Yoon,[1†] Yezee Seo,[1] Hoonkyu Kim,[2] Taekyoung Kwon[1‡]
[1]Information Security Lab., Graduation School of Information, Yonsei University
[2]Agency for Defense Development

요 약

FPGA (Field Programmable Gate Array) 는 개발자가 유연하게 프로그래밍 할 수 있다는 장점으로 인해 다양한 분야에서 사용되고 있다. 하지만 외부에서 구현된 시스템이 비트스트림 형태로 FPGA에 탑재 될 경우 오작동을 일으키거나 정보를 유출시키는 등의 하드웨어 악성 기능이 포함될 가능성이 있다. 이러한 이유로 비트스트림 역공학은 필수적이며, 따라서 이와 관련된 연구들이 진행되어 왔다. 본 논문에서는 FPGA 비트스트림 역공학 연구 사례 중 가장 대표적인 역공학 알고리즘을 활용한 BIL 비트스트림 역공학 도구에 대한 분석 실험을 진행하여 성능 및 한계점을 확인하였다.

ABSTRACT

Field Programmable Gate Array (FPGA) is widely used in a variety of fields because of its ability to be programmed as desired. However, when an externally implemented program is loaded on FPGA in the form of a bitstream, there is a possibility that hardware Trojans which cause malfunctions or leak information may be included. For this reason, bitstream reverse engineering is essential, and therefore related research has been conducted, such as BIL. In this paper, we analyze the BIL bitstream reverse engineering tool, which is the most representative algorithm, regarding its performance and limitations.

Keywords: FPGA, Cross-correlation Algorithm

I. 서 론

FPGA (Field Programmable Gate Array) 란 재프로그래밍 가능한 논리 소자의 집합으로, 개발자의 의도대로 유연하게 프로그래밍 할 수 있다는 장점으로 인해 자동차 산업, 항공, 국방 등 다양한 분야에서 사용되고 있다.[1] FPGA 기반의 시스템을 직접 구현하지 않고 외부 업체에 위탁하여 구현하는 경우, FPGA 칩에는 외부 업체나 IP (Intellectual Property) 제조사에서 구현한 시스템이 비트스트림의 형태로 탑재된다.[2] 이처럼 외부에서 구현된 시스템에는 임의의 공격자에 의해 오작동을 일으키거나 정보를 유출시키는 등의 하드웨어 악성 기능이 포함될 가능성이 있다[3]. 따라서 FPGA에 탑재된 비트스트림의 로직을 확인할 수 있는 단계까지 역공학 하여 시스템에 의도하지 않은 기능이 포함되어 있는지 검증할 필요가 있다.

이러한 이유로 최근 FPGA 비트스트림을 역공학

Received(10. 31. 2017), Modified(11. 28. 2017), Accepted(11. 28. 2017)
* 본 연구는 방위사업청과 국방과학연구소의 지원으로 수행되었습니다.(UD160066BD)
† 주저자, yjh1226@yonsei.ac.kr
‡ 교신저자, taekyoung@yonsei.ac.kr(Corresponding author)

하는 연구들이 진행되어왔다. 2008년 Note 등[4]은 비트스트림 내에서 내부 연결 정보를 추출하는 알고리즘과 역공학 도구 'debit'을 발표하였다. 2012년 Benz 등[5]은 'debit'[4]에서 제시된 알고리즘을 개선하여 내부 연결 정보에 대한 역공학 결과를 XDL (Xilinx Design Language) 형태로 출력하는 역공학 도구 'BIL'을 발표하였다. 2013년 Ding 등[6]은 비트스트림 내의 구성 정보에 대한 테이블을 구축하여 비트스트림을 NCD (Native Circuit Description) 파일로 복구하는 방법에 대해 제시하였다.

본 논문에서는 FPGA 비트스트림 역공학 연구 사례 중 가장 대표적인 역공학 알고리즘을 활용한 BIL 비트스트림 역공학 도구[5]를 분석하고, 분석 결과를 바탕으로 BIL이 가지고 있는 한계점을 제시한다.

II. BIL 비트스트림 역공학 도구

2.1 BIL 비트스트림 역공학 도구 개요

BIL 비트스트림 역공학 도구는 Xilinx 사 Virtex-5 제품군에 속하는 26개 모델[8]의 비트스트림을 대상으로 역공학을 수행한다. 비트스트림에서 로직 구현에 사용된 타일(tile) 간의 연결을 담당하는 pip (programmable interconnect point) 정보를 역공학 한다. 복구된 pip 정보는 일반 텍스트 형태로 로직에 대한 정보를 나타내는 XDL (Xilinx Design Language)[7] 파일 형식으로 출력된다.

$$pip \quad L \quad S \rightarrow E \tag{1}$$

식 (1)에서처럼, 복구된 pip 정보는 pip가 속한 타일의 종류와 위치(L), pip를 통해 연결되는 두 개의 선 start wire(S), end wire(E)로 표현된다[4].

BIL 비트스트림 역공학 도구는 2단계로 구성된다. 첫 번째 단계는 FPGA 분석 단계로 역공학 대상 FPGA에 대한 정보와 구성 데이터베이스를 생성하는 4개의 모듈로 구성되어 있다(Fig. 1).

- xdlrc_convert : FPGA 내부의 모든 리소스에 대한 정보를 일반 텍스트 형태로 기술한 XDLRC 파일을 분석하여 타일, 프리미티브 등의 정보를 추출한다.
- v5data_gen : Virtex-5 제품군에 속하는 26개 모델에 대한 리스트와 각 모델의 구성 데이터에 대한 주소 레이아웃을 생성한다.
- v5cfgmap_gen : FPGA 내 각 타일에 상응하는 구성 데이터 정보에 대한 테이블을 구축한다.
- bitcorrelate : 비트스트림과 이에 상응하는 XDL 파일에 상호상관 알고리즘을 적용하여 구성 데이터베이스를 구축한다.

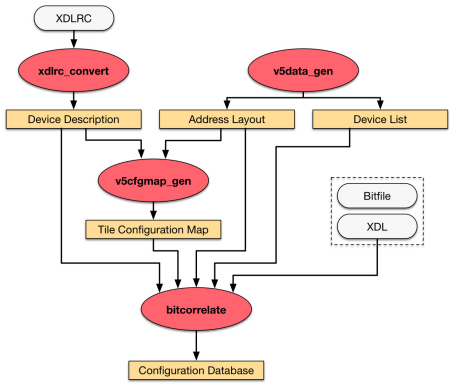


Fig. 1. BIL tool-chain for FPGA Analysis

두 번째 단계는 비트스트림 역공학 단계로 선행 단계를 통해 생성된 정보와 구성 데이터베이스를 활용하여 역공학을 수행한다. 3개의 모듈로 구성되며, 각 모듈은 다음과 같은 기능을 수행한다(Fig. 2).

- bit2xml : 입력받은 비트스트림의 내용을 분석하여 XML 파일로 나타낸다.
- bit_extract : 입력받은 비트스트림에서 구성 데이터를 추출하고, 구성 데이터 내 frame 정보를 텍스트 파일로 출력한다.

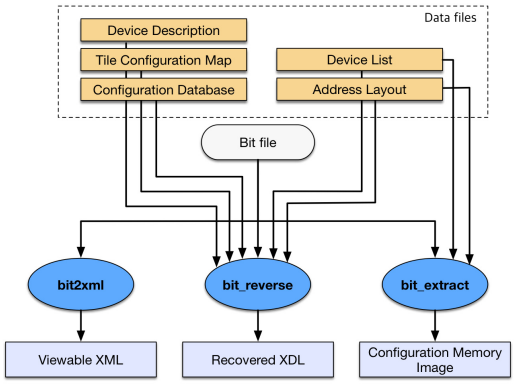


Fig. 2. BIL tool-chain for Bitstream Reversal

- bit_reverse : 역공학을 수행하는 모듈로써, 비트스트림에서 pip 정보를 복구하여 이를 XDL 파일 형태로 출력한다.

2.2 상호상관 알고리즘

상호상관 알고리즘은 비트스트림 내에서 XDL에 나타난 각 pip에 대응하는 비트 오프셋을 찾는 알고리즘이다. 알고리즘을 통해 pip에 대한 비트 오프셋이 구해지면 해당 pip는 'isolated pip'로 분류된다. 구해진 pip들은 데이터베이스에 저장되며 이것을 구성 데이터베이스라고 한다.

2.3 구성 데이터베이스

구성 데이터베이스는 Fig. 3.과 같이 'tile type', 'pip control set', 'bit offsets', 'bit values'의 계층 구조의 형태로 되어 있다. pip control set은 end wire가 같은 pip들의 집합을 의미하고 bit offsets는 pip control set 내에 있는 pip들과 관련된 비트의 오프셋들을 의미한다. bit values는 하나의 pip에 대응하는 비트 오프셋 값을 나타낸다. Fig. 3.의 'EL2MID1 → IMUX_B27: 0x00000005'를 예로 들었을 때, 16진수 bit value 0x00000005를 이진수로 변환하면 0000 0101이 된다. 이는 해당 pip의 값은 control set의 bit offsets 중 첫 번째(871)와 세 번째(1190)에 있는 오프셋의 비트가 1로 설정된 값이라는 의미이다.

Bit values는 'ISOLATED', 'NOT ISOLATED', 'ZEROED' 중 하나의 상태를 갖는다. 상호상관 알고리즘이 성공적으로 끝나게 되면 하나의 pip와 그에 대응하는 bit values만 남는데 이러한 경우는 'ISOLATED', 두 개 이상의 pip가 남

```
Tile type: INT
    PIP control set 1:
        Bit offsets: 871, 1189, 1190, 1254, 1255, 1317, 1319
        Bit values:
            EL2MID1 -> IMUX_B27: 0x00000005
            SW2MID1 -> IMUX_B27: 0x00000006
            ER2BEG2 -> IMUX_B27: 0x00000009
            EN2END1 -> IMUX_B27: 0x0000000a
            EL2END1 -> IMUX_B27: 0x00000014
            ER2MID1 -> IMUX_B27: 0x00000018
            BYP_BOUNCE_S0 -> IMUX_B27: 0x00000028
            FAN_BOUNCE6 -> IMUX_B27: 0x00000044
            BYP_BOUNCE1 -> IMUX_B27: 0x00000048
```

Fig. 3. Configuration database

```
Tile type : INT
    PIP control set 82:
        Bit offsets: 956, 958, 1082, 1083, 1086, 1087, 1149
        Bit values:
            LV18 =- LH0: ZEROED
            LV0 =- LH0: ZEROED
            EL2BEG2 -> LH0: 0x00000001
            EL2END2 -> LH0: 0x00000002
            ES2MID2 -> LH0: 0x00000005
            EN2END2 -> LH0: 0x00000006
                    ⋮
            NE2END2 -> LH0: 0x00000048
            NR2END2 -> LH0: 0x00000060
            LH18 =- LH0: NOT ISOLATED
            WS2MID_S0 -> LH0: NOT ISOLATED
```

Fig. 4. Representation of bit values (INT tile)

게 되면 'NOT ISOLATED', 하나의 pip가 남았지만 그에 대응하는 bit offsets 값이 전부 0으로 설정되어 있으면 'ZEROED'라고 표현한다. Fig. 4.는 이러한 상태들이 구성 데이터베이스 상에서 어떻게 표현되는지를 보여준다. bit values 내에 실선으로 표시된 부분은 'ZEROED', 점선으로 표시된 부분은 'NOT ISOLATED' 상태를 의미한다. 이외에 표시되지 않은 부분은 'ISOLATED' 상태를 의미한다.

2.4 역공학

역공학은 구성 데이터베이스를 이용하여 비트스트림에서 사용하고 있는 pip를 찾는 방식으로 진행된다. 역공학 결과로는 찾은 pip를 타일 타입, 타일 위치, pip 순으로 파일에 출력한다(Fig. 5).

```
INT_X11Y79 LV18 -> SR5BEG0
INT_X18Y79 WR2END0 -> SL2BEG_N2
INT_X20Y79 NE2MID0 -> IMUX_B2
INT_X20Y79 WS2MID2 -> IMUX_B3
INT_X20Y79 WS2MID2 -> IMUX_B39
INT_X20Y79 WS2MID2 -> IMUX_B45
INT_X20Y79 WS2MID2 -> IMUX_B9
INT_X20Y79 FAN_BOUNCE_N7 -> IMUX_B0
INT_X20Y79 NR2END0 -> IMUX_B1
INT_X20Y79 FAN_BOUNCE_N7 -> IMUX_B36
INT_X20Y79 CTRL_BOUNCE_N3 -> IMUX_B37
INT_X20Y79 FAN_BOUNCE_N7 -> IMUX_B42
```

Fig. 5. Reversed XDL

III. BIL 성능 실험 및 결과

3.1 실험 설계

본 연구에서는 Virtex-5 제품군 중 xc5vfx30t-ff665 모델을 대상으로 BIL의 성능 분석 실험을 진행한다. 실험은 역공학의 완전성과 정확성을 확인하는 것으로 나누어 진행하였다. 역공학의 완전성 확인은 여러 샘플을 통해 구성 데이터베이스를 직접 구축하고, 그 중 가장 잘 구축된 데이터베이스를 활용하여 역공학을 수행하는 방식으로 진행하였다. 정확성 확인은 역공학 수행 후 그 결과와 원본을 비교 분석하는 방식으로 진행하였다. 본 실험에서는 서로 다른 크기의 10개 로직에 대한 비트스트림 샘플을 사용하였다. 로직의 크기가 클수록 많은 pip가 사용된다.

3.2 역공학 완전성 실험 결과

Fig. 6.은 샘플 비트스트림과 그에 대응하는 XDL을 통해 구성 데이터베이스를 구축한 결과이다. Fig. 6.에서 볼 수 있듯이 대체적으로 해당 XDL에서 사용 중인 pip 개수가 많을수록 구성 데이터베이스에서 isolated pip의 비율이 증가했다. 하지만 샘플 ram_dp_sr_sw의 경우는 해당 INT 타일에서 사용하는 pip 개수가 ram_dp_ar_aw 보다 현저히 적지만 ram_dp_ar_aw와 거의 같은 수준으로 구성 데이터베이스를 구축하였다. 이는 구성 데이터베이스를 구축할 때 로직에 사용된 pip의 개수뿐만 아니라 사용된 pip의 다양성이 함께 고려되어야 함을 의미한다. 본 실험에서 가장 잘 구축된 데이터베이스는 샘플 jpegenc를 사용했을 때이며 isolated pip를

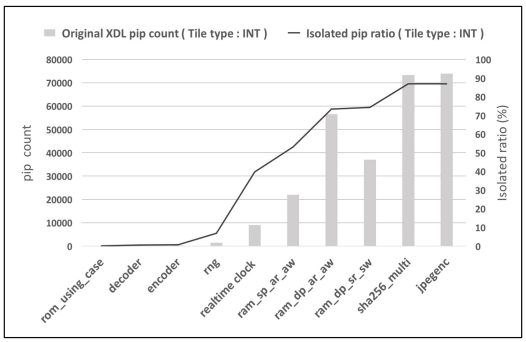


Fig. 6. Isolated pip ratio for INT tile

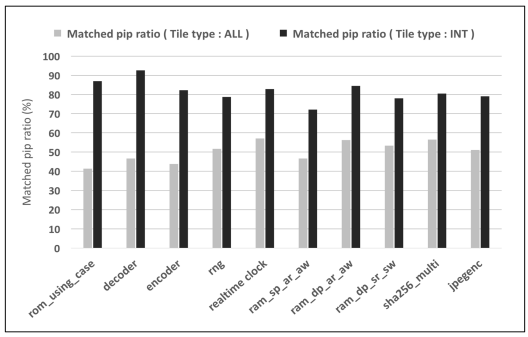


Fig. 7. Recovered pip ratio

87%까지 찾아냈다.

Fig. 7.은 jpegenc로 만든 구성 데이터베이스를 이용하여 비트스트림 샘플을 역공학한 결과이다. 회색 막대는 원본에 있는 모든 타일의 pip 개수 대비 복구된 pip 개수의 비율을 나타내고, 검은색 막대는 원본에 있는 INT 타일의 pip 개수 대비 복구된 pip 개수의 비율을 나타낸다. 모든 타일 타입을 대상으로 계산하였을 때는 평균 약 50%, INT 타일만을 대상으로 계산하였을 때는 평균 약 82%까지 복구에 성공하였다.

3.3 역공학 정확성 실험 결과

완전성 실험에서 구축한 jpegenc 구성 데이터베이스를 활용하여 역공학한 결과 원본 XDL에 없는 pip를 복구한 경우가 존재했다. 작은 로직의 경우에는 잘못 복구된 pip가 발견되지 않았고, 크기가 큰 로직일수록 잘못 복구된 pip 개수가 증가하였다. Table 1.은 각 샘플별 복구한 pip 개수 대비 잘못 복구한 pip의 비율을 계산한 오류율을 보여준다. Table 1.에서 볼 수 있듯 전체적으로 오류율이 상당히 낮은 것을 확인 할 수 있다. 이러한 결과를 통해 BIL 역공학 도구의 정확성은 매우 높다는 것을 확인하였다.

IV. 한계점 분석 및 토의

4.1 완전성

Benz 등[5]이 밝혔듯 상호상관 알고리즘은 다양한 타일 타입 중 INT 타일과 HCLK 타일에만 성공적으로 적용된다. 비록 INT 타일 내에 대부분의 pip가 존재하지만, 다른 타입의 타일에도 pip들이 존재

Table 1. Error rate for each sample

Sample / Count	rom_ using_ case	decoder	encoder	rng	realtime clock	ram_sp _ar_aw	ram_dp _ar_aw	ram_dp _sr_sw	sha256 _multi	jpegenc
Recovered pip	60	151	218	1,131	7,527	15,911	47,896	29,001	59,152	58,531
Correctly recovered pip	60	151	218	1,131	7,473	15,903	47,796	28,928	59,008	58,473
Incorrectly recovered pip	0	0	0	0	54	8	100	73	144	58
Error rate	0	0	0	0	.0071	.0005	.002	.0025	.0024	.0009

하기 때문에 BIL에서 사용되는 상호상관 알고리즘으로는 모든 pip를 복원할 수 없다.

또한 3.2의 실험 결과를 통해 알 수 있듯 구성 데이터베이스를 완전히 구성하는 것은 어렵다. 상호상관 알고리즘은 XDL에서 사용된 pip들에 대해서만 적용되기 때문에 완전한 데이터베이스 구축을 위해서는 FPGA 내의 모든 pip가 사용된 로직이 필요하다. 그러므로 구성 데이터베이스를 완전하게 구축하는 것에는 어려움이 따른다.

4.2 정확성

오류가 발생한 주된 요인은 Fig. 8.에서 볼 수 있듯 구성 데이터베이스에서 같은 end wire를 공유하는 pip 중 같은 비트 오프셋을 가지는 pip들이 존재하기 때문이다. 이로 인해 역공학 과정에서 원본에 있는 pip와 같은 bit values를 갖는 다른 pip를 복구하는 경우가 발생하였다.

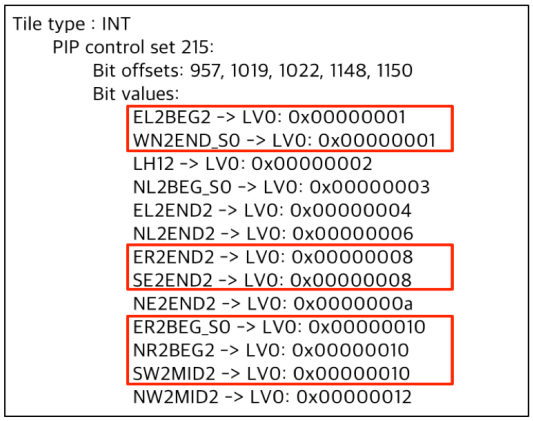


Fig. 8. Shared bit values

실제 역공학을 진행하는 과정에서는 오직 비트스트림만 주어지기 때문에 잘못 복구된 pip를 찾아내는 것은 어렵다.

V. 결 론

본 논문에서는 BIL 비트스트림 역공학 도구에 대한 분석 연구를 진행하였다. 분석 결과 BIL은 상호상관 알고리즘을 통해 로직 연결 정보인 pip를 일부 복구하는데 성공하였다. 그러나 FPGA 내 여러 종류의 타일 중 특정 타일에 대해서만 데이터베이스가 구축되어 역공학 되었고, 특정 타일에 대해서도 100% 데이터베이스를 구축하지 못하였다는 한계점이 존재했다. 이러한 한계점이 존재하는 이유는 크게 두 가지가 있다. 첫 번째는 상호상관 알고리즘을 수행할 때 한 개의 XDL 파일과 그에 대응하는 bit 파일 한 쌍만을 가지고 수행한다는 점이다. 그로 인해 XDL 파일에서 사용하지 않는 pip들에 대해서는 상호상관 알고리즘 자체가 수행되지 않는 문제가 있다. 두 번째는 상호상관 알고리즘 방식으로 찾을 수 없는 pip들이 존재한다는 것이다. 따라서 이 2가지 문제를 해결 할 수 있는 보완된 상호상관 알고리즘 연구가 필요하다. 이 외에도 특정 타일뿐만 아니라 모든 타일에 대해 구성 데이터베이스를 구축 할 수 있는 방법이 연구되어야 한다.

BIL은 연결 정보 이외에 구현된 로직에 대한 정보는 복구되지 않는다. 따라서 로직 구현에 사용되는 룩업테이블의 구성 정보를 추출하고 이를 연결 정보와 합성하는 방법 또한 연구되어야 한다.

References

[1] T. Todman, G. Constant S. Wilton, O. Mencer, W. Luk, and P. Cheung, "Reconfigurable computing: architectures and design methods," Computers and Digital Techniques, Vol. 152, No. 2, pp. 193-207, Mar. 2005.

[2] L. Sekanina, "Towards Evolvable IP Cores for FPGAs," In Proc. NASA/DoD Conference on Evolvable Hardware, pp. 145, Jul. 2003.

[3] S. Mal-Sarkar, R. Karam, S. Narasimhan, A. Ghosh, A. krishna, and S. Bhunia, "Design and Validation for FPGA Trust under Hardware Trojan Attacks," IEEE Transactions on Multi-Scale Computing Systems, Vol. 2, No. 3, pp. 186-198, Jun. 2016.

[4] J.B. Note and E. Rannaud, "From the bitstream to the netlist," In Proc. the 16th ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA), pp. 264-271, Feb. 2008.

[5] F. Benz, A. Seffrin, and S.A. Huss, "Bil: A tool-chain for bitsream reverse-engineering," In Proc. International Conference on Field Programmable Logic and Applications (FPL), pp. 735-738, Aug. 2012.

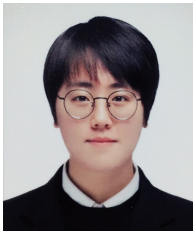
[6] Z. Ding, Q. Wu, Y. Zhang, and L. Zhu, "Deriving an NCD file from an FPGA bitstream: Methodology, architecture and evaluation," Microprocessors and Microsystems, Vol. 37, No. 3, pp. 299-312, May. 2013.

[7] C. Beckhoff, D. Koch and J. Torresen, "The Xilinx Design Language (XDL): Tutorial and use cased," In Proc. International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC), pp. 1-8, Jun. 2011.

[8] Xilinx Inc., "Virtex-5 Family Overview," https://www.xilinx.com/support/documentation/data_sheets/ds100.pdf, Aug. 2015.

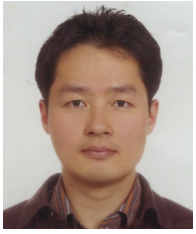
〈저자소개〉

윤 정 환 (Junghwan Yoon) 학생회원
2017년 2월: 부산외국어대학교 비지니스일본어학과 학사
2017년 3월~현재: 연세대학교 정보대학원 석사과정
〈관심분야〉 정보보호, 암호 프로토콜, 소프트웨어 보안 등



서 예 지 (Yezee Seo) 학생회원
2017년 2월: 덕성여자대학교 디지털미디어학과 학사
2017년 3월~현재: 연세대학교 정보대학원 석사과정
〈관심분야〉 정보보호, IoT 보안, 소프트웨어 보안 등

김 훈 규 (Hoonkyu Kim) 정회원
2015년 8월: 홍익대학교 컴퓨터공학과 박사
1987년 3월~현재: 국방과학연구소
〈관심분야〉 정보보호, 사이버 보안, 보안 OS 등



권 태 경 (Taekyoung Kwon) 종신회원
1992년 2월: 연세대학교 컴퓨터과학과 학사
1995년 2월: 연세대학교 컴퓨터과학과 석사
1999년 8월: 연세대학교 컴퓨터과학과 박사
1999년~2000년: U.C. Berkely Post-Doc.
2001년~2013년 8월: 세종대학교 컴퓨터공학과 교수
2007년~2008년: Univ. Maryland at College Park 교환교수
2013년 9월~현재: 연세대학교 정보대학원 교수
〈관심분야〉 암호 프로토콜, 네트워크 프로토콜, IoT 보안, Usable Security, HCI 등