Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.28, NO.5, Oct. 2018 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2018.28.5.1225

BIL 비트스트림 역공학 도구 개선 연구*

윤 정 환,† 서 예 지, 장 재 동, 권 태 경‡
연세대학교 정보보호연구실

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

Junghwan Yoon,† Yezee Seo, Jaedong Jang, Taekyoung Kwon‡
Information Security Lab., Graduation School of Information, Yonsei University

요 약

FPGA(Field Programmable Gate Array)를 이용한 시스템 개발은 개발 시간 단축 및 비용 절감을 위해 제3자에게 아웃소싱하는 형태로 발전하고 있다. 이러한 과정에서 악의적인 기능 및 오작동을 유발하는 하드웨어 악성기능(Hardware Trojan)이 시스템에 유입될 위협 또한 증가하고 있다. 하드웨어 악성기능의 탐지를 위해 다양한 방법들이 제시되고 있으나 FPGA에 탑재되는 비트스트림을 직접 수정하는 형태의 하드웨어 악성기능은 기존에 제시된 방법으로 탐지하기 어렵다. 이러한 유형의 하드웨어 악성기능 탐지를 위해서는 비트스트림으로부터 구현된 회로를 식별 가능한 수준으로 역공학하는 과정이 필요하며, 회로를 구성하는 여러 요소 중 특히 신호의 입출력 흐름을 파악할 수 있는 연결 정보를 역공학하는 것이 중요하다. 본 논문에서는 FPGA 비트스트림으로부터 연결 정보를 복구하는 도구인 BIL을 분석하고 이를 개선하기 위한 방법을 제시한다.

ABSTRACT

FPGA-based system development is being developed as a form of outsourcing that shortens the development time and reduces the cost. Through the process, the risk of letting the hardware Trojan, which causes malfunctions, seep into the system also increases. Various detection methods are proposed for the issue; however, such type of hardware Trojans is inserted by modifying a bitstream directly and therefore, it is hard to detect with the suggested methods. To detect the type of hardware Trojans, it is essential to reverse-engineer the electric circuit implemented by bitstream to a distinguishable level. Specifically, it is important to reverse-engineer the routing information of the circuit that can identify the input-output flow of the signal. In this paper, we analyze the BIL bitstream reverse-engineering tool-chain that uses the algorithm, which retrieves the routing information from FPGA bitstream, and suggest the method to improve the tool-chain.

Keywords: Hardware Trojan, Cross-correlation Algorithm, Reverse-engineering, XDLRC, PIP

I. 서 론

FPGA는 재프로그래밍 할 수 있고 초기 개발비가 저렴하다는 장점으로 인해 국방, 항공 등 다양한 분야에서 사용되고 있다. FPGA 개발단계는 HDL(Hardware Description Language)을 이용해 회로 설계를 한 후 합성, 맵핑, 배치 및 결선 단계를 거쳐 최종적으로 비트스트림 형태로 칩에 탑재된다.

FPGA 칩 설계는 외부 업체에 위임하는 경우가 많으며 이러한 경우 제3자에 의해 오작동 및 정보

Received(07. 16. 2018), Accepted(08. 14. 2018)
* 본 연구는 방위사업청과 국방과학연구소의 지원으로 수행되었습니다.(UD160066BD)
† 주저자, yjh1226@yonsei.ac.kr
‡ 교신저자, taekyoung@yonsei.ac.kr(Corresponding author)

유출과 같은 하드웨어 악성기능이 삽입될 수 있다. 하드웨어 악성기능 탐지를 위한 방법으로는 사이드 채널 분석 및 로직 테스팅이 있지만 사이드 채널 분석의 경우에는 골든 모델과 비교가 필요하며 매우 작은 사이즈로 삽입된 하드웨어 악성기능은 탐지하기 어렵고 로직 테스팅은 악성기능의 트리거 조건이 까다로울 경우 활성화시키기 어렵다는 한계점이 있다.

비트스트림을 직접 수정하여 삽입하는 하드웨어 악성기능[4]의 경우에는 배치 및 결선 중 생성된 로그 파일에 악성기능 삽입 흔적을 남기지 않기 때문에 탐지되기 어렵다. 따라서 비트스트림에 구현된 회로 정보를 확인 할 수 있는 수준으로 역공학하여 악성기능이 삽입되어 있는지 검증할 필요가 있다. 그 중에서도 회로의 연결 정보를 복구하는 것은 실제 신호의 입출력 흐름을 파악할 수 있기 때문에 역공학 연구에서 중요한 부분이다. 하지만 FPGA의 디자인 크기와 복잡성이 크게 증가하고 있다는 점과 제조사에서 비트스트림에 대한 모든 정보를 공개하고 있지 않다는 점에서 비트스트림 역공학은 어려운 문제로 여겨지고 있다. 이러한 이유로 비트스트림 형식 분석과 효율적인 재구성을 목표로 한 역공학 연구들이 진행되어 왔다. 2008년 Note 등[1]은 비트스트림 연결 정보 역공학 도구 'debit'을 발표하였다. 2012년 Benz 등[2]은 FPGA 내의 모든 리소스에 대한 정보가 포함된 XDLRC 파일을 이용하여 'debit'[1]에서 제시된 알고리즘을 개선한 'BIL'을 발표하였다. 2013년 Ding 등[3]은 분산형 컴퓨터로 구성 정보에 대한 맵핑 테이블을 구축하였고 이를 기반으로 한 비트스트림 역공학 도구 'Bit2NCD'를 발표하였다.

본 논문에서는 FPGA 비트스트림 역공학 도구인 BIL을 분석하고, 분석 결과를 바탕으로 개선 방법을 제시한다.

II. BIL 비트스트림 역공학 도구

FPGA는 Fig. 1과 같이 다양한 종류의 타일(tile)들의 2차원 배열로 되어있다. 그 중에 가장 대표적인 타일은 CLB (Configurable Logic Block) 타일과 INT (Interconnect) 타일이다. CLB 타일에는 사용자 회로 구현에 필요한 LUT (Look-Up Table), 게이트, 플립플롭, 멀티플렉서, 메모리 등이 배치되어 있고 INT 타일은 스위치 행렬로 되어 있으며 타일 간의 신호를 연결해준다. 이처럼 신호를 연결할 때 쓰이는 자원을 PIP

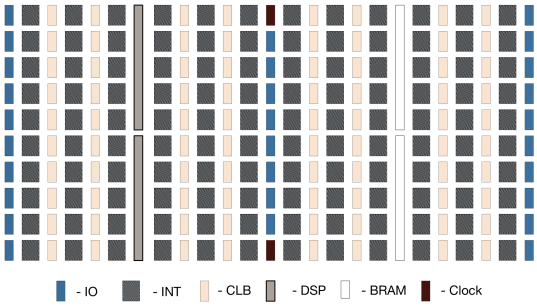


Fig. 1. Simplified FPGA tile layout

(Programmable Interconnect Point) 라고 한다. PIP는 다양한 종류의 타일 안에 존재하는 자원이며 그 중에서도 INT 타일에 가장 많은 PIP가 존재한다.

2.1 BIL 개요

BIL 비트스트림 역공학 도구는 비트스트림에서 회로 연결 정보인 PIP를 역공학 한다. 복구된 PIP는 특정 문법을 가지는 텍스트 파일인 XDL (Xilinx Design Language)[7]수준으로 출력된다.

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

식 (1)에서처럼, 복구된 PIP 정보는 타일의 종류와 위치(L), 신호의 흐름인 start wire(S), end wire(E)로 표현된다. 따라서 해당 PIP가 어떤 타일에 어디 위치에서 사용되는 자원인지 알 수 있다.

2.2 상호상관 알고리즘

상호상관 알고리즘은 비트스트림에서 XDL 파일에 사용된 각 PIP에 대응하는 비트 오프셋을 찾는 알고리즘이다. 알고리즘을 통해 하나의 PIP에 대응하는 비트 오프셋이 구해지면 구성 데이터베이스에 저장한다.

2.3 구성 데이터베이스

구성 데이터베이스는 각 PIP에 대해 상호상관 알고리즘이 적용된 결과가 저장되어 구축된 것이며 Fig. 2와 같이 'tile type', 'PIP control set', 'bit offset', 'bit values'의 계층 구조로 되어 있

```
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
        LV6 -> LH0: 0x00000008
        NL2BEG_S0 -> LH0: 0x00000009
        NL2END2 -> LH0: 0x0000000a
        SW2MID2 -> LH0: 0x00000010
        ER2BEG_S0 -> LH0: 0x00000014
                    ⋮
        LH18 =- LH0: NOT ISOLATED
        WS2MID_S0 -> LH0: NOT ISOLATED
```

Fig. 2. Configuration database

다. PIP control set은 같은 end wire를 갖는 PIP들의 집합이다. bit values는 각 PIP의 비트 오프셋 값이다. Fig. 2의 'EN2END2 -〉 LH0: 0x00000006'을 예로 들면, 16진수 bit value 0x00000006를 이진수로 변환하면 0000 0110이 된다. 이는 해당 PIP의 값이 control set의 bit offsets 중 두 번째(958)와 세 번째(1082)에 있는 오프셋의 비트가 1로 설정되어 있다는 것을 의미한다. bit values는 'Isolated', 'Not isolated', 'Zeroed' 세 개의 상태 중 하나를 갖는다. 하나의 PIP에 대한 상호상관 알고리즘이 끝난 후 PIP 대응하는 비트 오프셋이 남으면 'Isolated', 두 개 이상의 PIP에 해당하는 비트 오프셋이 남아 있으면 'Not isolated', PIP에 대응하는 비트 오프셋이 없는 경우에는 'Zeroed'라고 표현한다.

2.4 역공학

역공학은 비트스트림에서 구성 데이터베이스 내의 각 PIP 비트 오프셋을 비트스트림에서 찾고, 찾은 비트들이 1로 설정되어 있으면 해당 PIP를 복구하는 방식으로 진행된다.

III. BIL 분석 실험 및 개선 방법

본 연구에서는 Xilinx 사의 Virtex-5 xc5vfx30t-ff665 모델[8]에 탑재 가능한 15개의 샘플을 대상으로 실험을 진행하였다. 각 샘플은 OpenCores[5]에 공개된 소스코드를 통해 확보하였다.

3.1 BIL 분석 및 한계점

BIL 역공학 도구 분석 결과 세 가지 한계점이 존재하였다. 첫 번째는 상호상관 알고리즘으로 구성 데이터베이스를 구축할 때 한 쌍의 비트스트림 파일과 XDL 파일만 사용한다는 것이다. 이로 인해 XDL 파일에서 사용된 PIP들에서만 알고리즘이 적용되었고 XDL에서 사용되지 않은 PIP에 대해서는 비트 오프셋 값을 찾을 수 없다. 두 번째는 상호상관 알고리즘이 단일 PIP만 처리한다는 것이다. 따라서 다수의 PIP가 항상 같이 사용되는 경우에는 'Not isolated'로 되어 찾지 못한 것으로 판단한다. 세 번째는 비트스트림에 존재하지 않는 PIP들은 처리하지 못한다는 것이다. 이러한 PIP를 'Zeroed'라고 하며 실제 비트스트림에서는 나타나지 않기 때문에 상호상관 알고리즘으로는 찾을 수 없다.

3.2 제안하는 개선된 방법

3.2.1 구성 데이터베이스 구축 방법 개선

Fig. 3은 기존 BIL에서 구성 데이터베이스를 구축하는 방법과 본 논문에서 제안하는 방법을 비교한 흐름도이다. BIL에서는 한 쌍의 비트스트림 파일과 XDL 파일을 이용하여 구성 데이터베이스를 구축하였지만 제안하는 방법은 여러 개의 비트스트림 파일과 XDL 파일을 이용한다. Table. 1은 실험에 사용된 파일 개수와 수행시간을 나타낸 표이다. 개선된 방법을 사용한 경우 기존에 비해 데이터베이스 구축

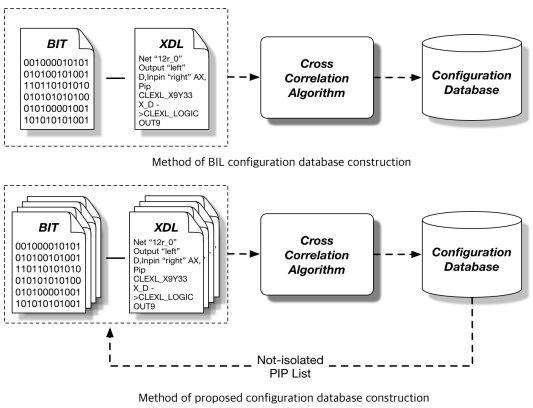


Fig. 3. Comparison between BIL method and proposed method

Table 1. Configuration database construction result

	Pair of bit and XDL files	Runtime (sec)	DB construction rate (INT tile)
BIL method	1	789.28	87.0%
Proposed method	314	6216.07	93.6%

율이 6.6%p 상승한 것을 확인 할 수 있다. 기존 방법보다 구성 데이터베이스를 구축하는 데에는 시간이 오래 소요되지만 역공학 시에는 사전에 구축된 구성 데이터베이스를 이용하므로 데이터베이스 구축 속도는 역공학 성능에 영향을 미치지 않는다.

3.2.2 Not isolated PIP 복구

BIL에서는 알고리즘 수행 후 단일 PIP에 대응하는 비트 오프셋이 남은 경우에만 찾은 것으로 판단하고 해당 PIP와 비트오프셋을 데이터베이스에 저장한다. 하지만 알고리즘 수행 결과 2개 이상의 PIP가 남는 경우에는 'Not isolated'로 표시되고 해당 PIP들과 비트오프셋은 저장되지 않는다. 따라서 이러한 경우는 복구 되지 않는다. 그러나 분석 결과 Not isolated PIP 중 항상 다른 PIP와 함께 사용되는 PIP가 존재하는 것을 확인했다. 따라서 이러한 특징을 갖는 PIP들과 이에 대응하는 비트 오프셋을 하나의 셋으로 구분하여 별도의 데이터베이스를 구축하였다. 실험 결과 총 87개의 셋을 확보하였고 이를 역공학에 활용하였다.

3.2.3 INT 타일 내의 Zeroed PIP 복구

Zeroed PIP는 비트스트림에 나타나지 않으므로 [1]에서 제시한 상호상관 알고리즘으로 복구하는 것이 불가능하다. 따라서 본 논문에서는 다른 PIP와의 연결 정보를 활용하여 Zeroed PIP를 복구하였다. Fig. 4는 한 개의 INT 타일에서 사용된 PIP들을 XDL 파일에서 추출한 결과이며 이를 통해 Zeroed PIP는 같은 타일 내의 다른 PIP와 연결되어 사용되는 것을 확인할 수 있다. 이러한 특성을 바탕으로 26개의 XDL 파일을 이용하여 Zeroed PIP와 연결되는 PIP를 찾아 별도의 데이터베이스를 구축하고 역공학에 이용하였다. 그러나 Fig. 5와 같이 Zeroed PIP들 중 start wire가 같은 PIP들이 존재하기 때문에 단순 연결로만 복구하는 경우 오류가 발생할 수 있다. Fig. 6은 이러한 오류를 제어하는 방법을 나타낸 그림이다. 오류가 발생하는 경우는 대부분 start wire가 같으나 end wire가 '_B#'과 '_BOUNCE#'로 나뉜 경우에 발생한다. 분석 결과 같은 타일 안에 특징이 비슷한 start wire들은 end wire 또한 비슷한 특징을 사용하는 경우가 대부분이었다. 이러한 특징을 이용하여 잘못 복구된 PIP들을 찾아 제거한다. 예를 들어, Fig. 6에서 start wire가 'FAN#'으로 시작하는 PIP의 수는 4개이고 그 중에서 end wire가 '_B#'으로 끝나는 PIP는 1개, '_BOUNCE#'으로 끝나는 PIP는 3개이다. 따라서 FAN5를 start wire로 갖는 PIP 중 다른 3개의 PIP와 다른 end wire를 갖는 PIP인

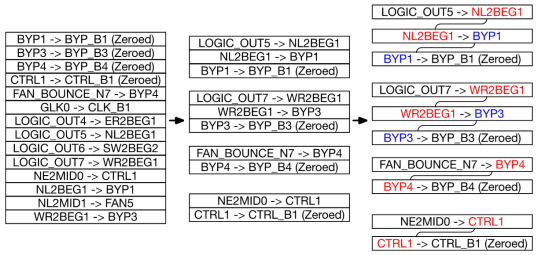


Fig. 4. Used PIPs in one of INT tiles

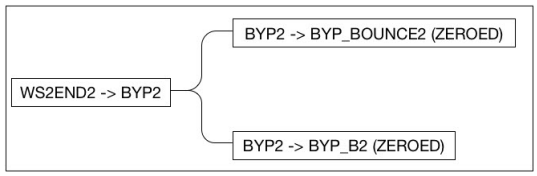


Fig. 5. Shared start wire

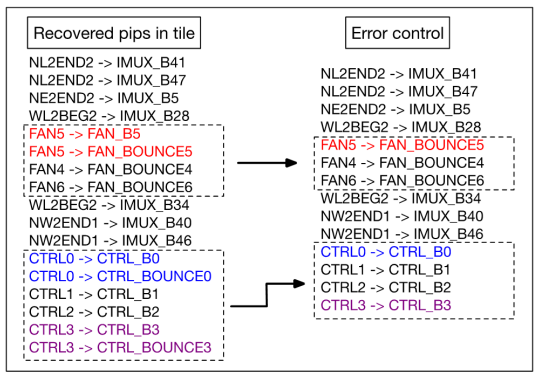


Fig. 6. Zeroed PIP error control

'FAN5 → FAN_B5'를 잘못 복구한 것으로 판단하여 제거한다.

3.2.4 INT 타일 외 다른 타일의 Zeroed PIP 복구

INT 타일 이외에 대부분 타일의 PIP들은 비트스트림에 나타나지 않는데 이는 해당 PIP들은 연결된 PIP의 사용 여부에 따라 자동으로 결선되기 때문이다. 예를 들어, INT 타일에서 PIP INT_X#Y# A -〉 B 가 사용되었다면 이와 연결된 CLB 타일의 CLBLL(M)_X#Y# B -〉 C 는 반드시 결선된다. 이러한 특성을 가지는 PIP는 비트스트림에 나타나지 않으며 따라서 오프셋 또한 존재하지 않는다. 따라서 이러한 타일들의 PIP를 복구하기 위해 Xilinx에서 제공하는 XDLRC 파일을 이용한다. XDLRC는 Xilinx FPGA에서 사용가능한 모든 자원에 대한 정보를 가지는 파일로써 Xilinx에서 제공하는 xdl 툴을 사용하여 생성 가능한 파일이다. 본 논문에서는 PIP 복구를 위해 자원정보 중 wire 연결정보를 활용하였다. wire 연결 정보는 Fig. 7에 나타난 클래스와 같이 나타난다. m_xOffset, m_yOffset은 해당 타일 기준 x, y 좌표 오프셋을 나타내고 m_tileTypeIndex는 해당 타일의 타입을 나타내는 인덱스 값이다. 따라서 해당 wire와 연결 가능한 wire들을 확보 할 수 있으며 이를 통해 연결 가능한 PIP 자원들을 알 수 있다.

```
class Wire {
      ……
private:
      std::string m_name
      WireConnections  m_connections
}

class WireConnection {
      ……
private:
      short m_xOffset;
      short m_yOffset;
      unsigned short m_tileTypeIndex;
      unsigned short m_wireIndex;
}
typedef std::vector<WireConnection> WireConnections;
```

Fig. 7. Wire & WireConnection class in BIL

3.3 개선 결과

Fig. 8은 기존 BIL 역공학 결과와 개선된 방법으로 역공학한 결과를 비교한 그래프이다. 비교를 위해 OpenCores에서 확보한 15개의 샘플을 사용하였다. 비교 분석한 결과 기존 BIL을 통한 PIP 평균 복구율이 54.4%였던 반면 개선된 방법을 사용한 경우 평균 복구율이 93.6%까지 증가하였다. 오류율은 기존 0.38%에서 0.44%로 소폭 상승 하였다. 분석 결과 복구율 증대에 가장 중요한 요인은 INT 타일 외의 다른 타일의 PIP 복구 개선에 있었다. 본 논문에서는 기존 BIL의 한계점을 분석 및 개선하였고 그 결과 모든 타일에서 사용된 PIP의 약 93%를 복구 할 수 있었다.

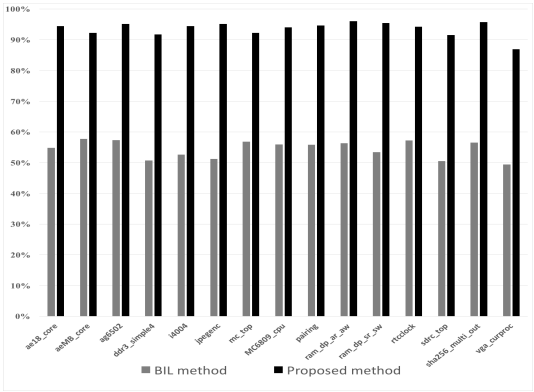


Fig. 8. Recovered PIP rate for each sample

IV. 결 론

본 논문에서는 BIL 비트스트림 역공학 도구에 대한 개선 연구를 하였다. 그 결과 기존 BIL이 복구하지 못한 INT 타일 내의 Zeroed PIP와 그 외 다른 타일 타입에 존재하는 Zeroed PIP까지 복구 가능한 것을 확인하였다. 이를 통해 PIP 평균 복구율이 기존대비 39.2%p 상승하였다.

본 논문에서는 회로 구현에 사용된 자원 간의 연결 정보는 복구하였으나 구현된 회로의 기능에 대한 정보는 복구되지 않는다. 이를 위해서는 회로 구현에 사용되는 LUT의 구성 정보를 추출하고 본문에서 복구한 연결 정보와 합성하여 전체 회로의 기능을 알 수 있는 회로의 형태로 역공학하는 방법이 연구되어야 한다.

References

[1] J.B. Note and E. Rannaud, "From the Bitstream to the Netlist," In Proc.

ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA), pp. 264-271, Feb. 2008.

〔2〕 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.

〔3〕 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.

〔4〕 R.S Chakraborty, I. Saha, A. Palchaudhuri, and G. Naik, "Hardware Trojan Insertion by Direct Modification of FPGA Configuration Bitstream," IEEE Design & Test, Vol. 30, No. 2, pp. 45-54, Apr. 2013.

〔5〕 OpenCores, "https://opencores.org," Jul. 2018.

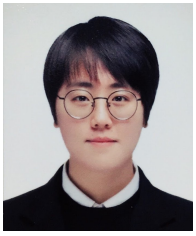
〔6〕 S. Bhunia, M.S Hsiao, M. Banga and S. Narasimhan, "Hardware Trojan attacks: threat analysis and countermeasures," Proceedings of the IEEE, Vol. 102, Iss. 8, pp. 1229-1247, Jul. 2014.

〔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.

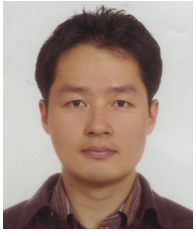
〈저자소개〉

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

장 재 동 (Jaedong Jang) 학생회원
2017년 2월: 성결대학교 컴퓨터공학과 학사
2017년 9월~현재: 연세대학교 정보대학원 석사과정
〈관심분야〉 정보보호, IoT 보안 ,웹 보안 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〉 암호프로토콜, Usable Security, 소프트웨어/시스템보안, 기계학습과보안 등