https://doi.org/10.33778/kcsa.2021.21.2.029

디지털 서명과 암호화 기반 보안 UART의 설계와 구현★

김주현*, 주영진**, 허아라***, 조민경****, 류연승*****

이규호******, 장우현******, 유재관******

요 약

UART(Universal asynchronous receiver/transmitter)는 데이터를 직렬 형태로 전환하여 전송하는 하드웨어 장치로서 대부분의 임베디드 시스템에서 시스템 진단 및 디버깅 용도로 널리 사용되고 있다. 해커는 UART의 기능을 이용하여 시스템 메모리나 펌웨어에 접근할 수 있고 시스템의 관리자 권한 취득을 통한 시스템 장악도 가능하다. 본 논문에서는 UART를 통해 침투하는 해커의 공격을 방어하기 위한 보안 UART를 연구하였다. 제안한 기법은 약속된 UART 통신 프로토콜을 사용하는 인가된 사용자만이 UART 접속을 허용하고 비인가자의 접속은 불허한다. 또한, 스니핑을 통한 프로토콜 분석을 막기 위해 데이터를 암호화하여 전송한다. 제안한 보안 UART 기법을 임베디드 리눅스 시스템에 구현하고 성능검증을 수행하였다.

Design and Implementation of Secure UART based on Digital Signature and Encryption

Ju Hyeon Kim*, Young Jin Joo**, Ara Hur***, Min Kyoung Cho****, Yeon Seung Ryu*****

Gyu Ho Lee******, Woo Hyun Jang******, Jae Gwan Yu******

ABSTRACT

UART (Universal asynchronous receiver/transmitter) is a hardware device that converts data into serial format and transmits it, and is widely used for system diagnosis and debugging in most embedded systems. Hackers can access system memory or firmware by using the functions of UART, and can take over the system by acquiring administrator rights of the system. In this paper, we studied secure UART to protect against hacker attacks through UART. In the proposed scheme, only authorized users using the promised UART communication protocol are allowed to access UART and unauthorized access is not allowed. In addition, data is encrypted and transmitted to prevent protocol analysis through sniffing. The proposed UART technique was implemented in an embedded Linux system and performance evaluation was performed.

Key words : UART, Protection, Security, Communication protocol, Digital signature, Encryption

접수일(2021년 05월 11일), 게재확정일(2021년 06월 28일)

★ 이 논문은 2020년도 방위사업청과 국방과학연구소(계약번호 : UC190045ED)의 지원을 받아 수행된 연구임. 이 논문은 2018년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No.2018R1D1A1B07045375).

* 명지대학교 컴퓨터공학과 학사 졸업(주저자)

** 명지대학교 보안경영공학과 석사과정(공동저자)

*** 명지대학교 보안경영공학과 박사수료(공동저자)

**** 명지대학교 컴퓨터공학과 조교수(공동저자)

***** 명지대학교 보안경영공학과 교수(교신저자)

****** LIG넥스원 선임연구원(공동저자)

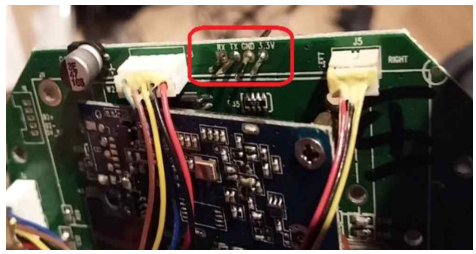
1. 서 론

UART는 범용 비동기화 송수신기(Universal asynchronous receiver/transmitter)의 약자로 병렬 데이터 형태를 직렬 데이터 형태로 전환하여 전송하는 하드웨어 장치이다. UART는 MCU(Micro Controller Unit)에서 기본적으로 제공하기 때문에 대부분의 임베디드 시스템 개발자들은 시스템 진단 및 디버깅 용도로 많이 사용하고 있다.

(그림 1)은 일반적인 컴퓨터 시스템의 PCB(Printed Circuit Board)에서 쉽게 볼 수 있는 UART의 예시이다. 그림에서 보듯이 UART는 PCB에 노출되어 있어 쉽게 접근 가능한데, 해커는 UART를 이용하여 시스템 메모리나 펌웨어에 접근하여 시스템 데이터를 유출하거나 시스템의 취약점을 파악하여 시스템의 관리자 권한 취득을 통한 시스템 장악을 할 수 있다[1].

최근 UART 해킹을 차단하기 위해 UART 포트를 제거하거나 비활성화하는 방법 등이 제시되었으나 포트의 제거는 시스템 진단 및 디버깅을 제한하는 문제점이 있으며 비활성화 방법은 해커의 공격으로 무력화되고 있다[1]. UART 통신 데이터의 스니핑을 막기 위해 데이터 암호화 기법이 제안되었으나 비인가자의 접속 여부를 확인하는 데는 한계가 있다[2].

본 논문에서는 UART를 통한 해커의 공격을 방어하기 위한 보안 UART를 제안하였다. 제안한 기법은 UART의 약속된 데이터 통신 프로토콜을 사용하는 인가된 사용자만이 UART 접속을 허용하고 비인가자의 접속은 불허하는 방법이다. UART 접속 시 일반적으로 터미널 에뮬레이터를 사용하며, 제안한 기법은 터미널 에뮬레이터와 시스템의 운영체제 간 디지털 서명 기반의 약속된 데이터 통신 프로토콜을 사용한다. 시스템의 운영체제의 UART 디바이스 드라이버에서 UART 통신이 약속된 프로토콜을 따르는지 검사하고 프로토콜에 맞지 않는 입력이 들어오면 해커로 인식해 접근을 차단한다. 또한, 스니핑을 통한 프로토콜 분석을 막기 위해 데이터를 암호화한다. 제안한 보안 UART 기법은 기 수행했던 선행 연구[3, 4]를 확장하고 실제 시스템에서 구현 및 성능검증을 수행한 연구 결과이다.



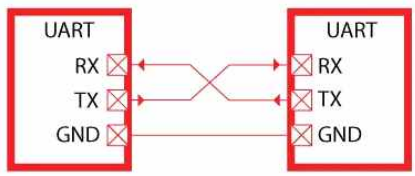
(그림 1) UART 포트의 예

본 논문의 구성은 다음과 같다. 2장에서는 UART 통신 방법을 설명하고 UART를 통한 해킹 사례를 살펴본다. 3장에서는 제안하는 디지털 서명과 암호화 기반의 보안 UART 통신 기법을 설명한다. 4장에서는 제안하는 보안 UART 통신 기법의 구현과 성능검증 결과를 기술하고 5장에서 논문의 결론을 맺는다.

2. 배경

2.1 UART 개요

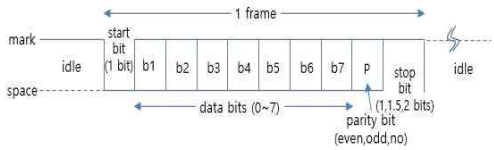
UART는 데이터를 한 비트씩 직렬 방식으로 전환하여 데이터를 전송한다. 송신 측에서 통신 데이터는 메모리 또는 레지스터에 들어 있으며 저장되어 있는 데이터를 비트로 읽어 직렬화하여 송신한다. UART는 RX(수신), TX(송신), Ground(그라운드), VCC(전압)의 4개 핀으로 구성되며 시리얼 통신을 위해서는 (그림 2)와 같이 RX와 TX가 서로 연결된다.



(그림 2) 통신을 위한 RX와 TX 연결

비동기 통신이므로 동기 신호가 전달되지 않으며 둘 간의 보율(baud rate)을 일치시켜주어야 한다. 또한, 수신 측에서 동기 신호를 찾아내어 데이터의 시작과 끝을 알 수 있도록 약속되어 있다. 이를 위해 데이터 필드는 최대 8비트가 기본 단위이고, 데이터 필드

이외에 시작 비트(start bit)와 정지 비트(stop bit)가 추가되며, 오류 검출용 패리티 비트(parity bit)가 추가되어 하나의 프레임으로 전송된다. (그림 3)은 UART 데이터 프레임 형태를 보여주고 있다.



(그림 3) UART 데이터 프레임

송신 측에서 데이터를 내보내기 직전에, 데이터 송신 시작을 알려주기 위하여 신호를 low 상태로 만들고 1비트 길이를 유지하는데 이를 시작 비트라고 한다. 그다음에는 양측에서 미리 약속된 길이만큼의 비트 데이터를 차례로 전송한다. 데이터 길이는 5~8비트인데 일반적으로 8비트이다. 데이터 길이는 그때그때 변경할 수 있는 것이 아니라 송수신 측에서 한번 길이를 정하면 계속 같은 크기의 데이터를 주고받아야 한다. 데이터 전송이 완료되면 오류 검출을 위하여 패리티 비트를 보낸다. 패리티 비트는 옵션 사항이므로 양쪽의 약속에 의하여 생략할 수 있다. 패리티 비트 다음에는 데이터 프레임의 끝을 알려주는 high 신호를 보내어 정지 비트를 전송한다.

수신하는 측에서는 신호를 계속 모니터링하고 있다가 신호가 high에서 low 상태로 변경되면 지금부터 데이터가 입력된다고 판단하여 데이터를 받을 준비를 하게 된다.

2.2 UART를 통한 해킹 사례

PCB에 있는 UART 핀의 식별과 접근 방법이 인터넷에 문서화되어 쉽게 구할 수 있고, UART 직렬 통신 방법이 매우 간단하기 때문에 UART는 해커의 시스템 공격 통로로 쉽게 이용된다.

일반적으로 시스템의 부트로더 및 운영체제는 부팅 과정에서 일련의 상태 정보를 시리얼 포트로 출력한다. 이 정보는 UART로 송신되므로 부팅 시 UART를 통해 부트로더와 운영체제 정보를 획득할 수 있으며 이를 통해 시스템 취약점 정보를 알아낼 수 있다.

또한, 어떤 시스템은 부팅 과정에 특수키가 입력되면 부트로더 명령어 모드로 진입할 수 있으며 명령어 모드에서는 시스템 권한을 취득할 수 있다. 이 경우, 해커는 부트로더 명령어를 사용하여 펌웨어를 다운로드하거나 펌웨어에 접근하여 저장되어 있는 인증서, 키, 각종 통신 프로토콜을 획득할 수 있다.

또한, 시스템 개발자는 시스템의 상태를 확인하기 위해 UART 통신으로 정보를 출력하고 확인하는데 프로그램의 동작 과정에서 미리 정의된 디버그 메시지 및 현재 상태에 대한 정보들을 포함한다. 만약 이런 정보를 해커가 취득하는 경우 시스템 취약점 분석에 활용되어 시스템 해킹에 이용된다.

부팅 후 셸(shell)을 실행하게 된다면 시스템의 동적 분석까지 가능하며 시스템 권한을 취득할 수도 있다.

<표 1> UART를 통한 해킹 사례

구분	공격 내용
부트 메시지	취약점 분석에 필요한 각종 정보 획득
디버그/오류 메시지	시스템, 응용프로그램 정보 획득
부트로더	펌웨어 획득, 새로운 펌웨어 퍼징
커널	시스템 취약점 분석에 필요한 각종 정보 획득
셸	바이너리 획득 및 시스템의 동적 분석

(그림 4) UART 데이터 프레임

3. 보안 UART 설계

3.1 개요

본 논문에서 제안하는 보안 UART 기법은 시스템

의 운영체제 디바이스 드라이버에서 UART의 약속된 데이터 통신 프로토콜을 사용하는 UART 접속은 허용하고 비인가자의 접속은 불허하는 방법이다. UART 접속 시 일반적으로 터미널 에뮬레이터를 사용하며, 제안한 기법은 터미널 에뮬레이터와 시스템의 운영체제 간 디지털 서명 기반의 약속된 데이터 통신 프로토콜을 사용한다. 시스템의 운영체제의 UART 디바이스 드라이버에서 UART 통신이 약속된 프로토콜을 따르는 지 검사하고 프로토콜에 맞지 않는 입력이 들어오면 해커로 인식해 접근을 차단한다. 또한, 스니핑을 통한 프로토콜 분석을 막기 위해 데이터를 암호화한다. 인가된 터미널 에뮬레이터를 사용하면 정상적인 접근과 통신이 이루어지며, 비인가된 터미널 에뮬레이터를 사용한다면 감지되어 접근을 차단하게 된다.

3.2 UART 통신 프로토콜

제안하는 UART 통신 프로토콜은 터미널 에뮬레이터와 시스템의 운영체제 디바이스 드라이버 간 약속된 프로토콜이다. 터미널 에뮬레이터는 프로토콜에 따라 데이터를 송신하며, 디바이스 드라이버는 이를 검사하여 올바른 경우 접근을 허용한다.

터미널 에뮬레이터가 보내는 메시지의 구조는 (그림 5)와 같이 <순서번호 + 데이터 + 디지털 서명>으로 구성되며, 메시지는 암호화되어 전송한다.

순서번호 (8비트)	데이터 (8비트)	서명 (16비트)

(그림 5) 메시지 구조

3.2.1 디지털 서명

송신하는 데이터마다 디지털 서명을 첨부하며 이는 순서번호와 데이터에 대한 해시 값이다. 해시 값은 해시 함수 SHA-1을 사용했으며, 구해진 해시 값의 앞부분 16비트만을 디지털 서명으로 사용한다.

일반적으로 암호학적 해시 함수들이 만드는 해시 값은 128비트 이상으로, UART 통신이 한 번에 보내는 8비트에 비해서 너무 크므로 앞 16비트만을 사용하였다.

수신 측(디바이스 드라이버)은 받은 데이터 및 순서번호의 해시 값을 계산하고 디지털 서명과의 일치 여부를 검사한다. 만약 불일치하는 경우 잘못된 통신으로 확인하여 접속을 차단한다.

3.2.2 순서번호

데이터가 같은 메시지들에 대해서는 디지털 서명이 같으므로 스니핑을 통해 데이터와 디지털 서명을 탈취한다면 프로토콜을 분석할 가능성이 있다.

본 연구에서는 메시지마다 순서번호를 순차적으로 증가하여 붙여주고, 디지털 서명을 계산할 때 데이터뿐만 아니라 순서번호도 포함함으로써 같은 데이터일지라도 디지털 서명을 다르게 만들었다. 즉, 보내는 메시지마다 다른 순서번호를 넣고 데이터와 순서번호를 함께 해시 값을 계산해서 디지털 서명으로 쓰면, 데이터가 같은 메시지들에 대해서도 다른 디지털 서명을 만들 수 있다.

3.2.3 암호화

(그림 5)의 메시지는 암호화하여 전송하며, 본 논문에서는 공개키 암호 시스템인 RSA를 사용한다.

RSA는 암호화 키가 (d,n)이고 복호화 키가 (e,n)일 때에 평문 M을 $C=M^d \bmod n$으로써 암호화하고 암호문 C를 $M=c^e \bmod n$으로써 복호화하므로 암호문과 평문은 0 이상 $n-1$ 이하로 계산된다. 그런데 UART 통신은 한 번에 8비트를 보내므로 UART 통신에 입력되는 평문은 0 이상 $2^8-1=255$ 이하고, 암호문이 255 이하인 평문으로 잘 복호화되려면 n은 $255+1=256$ 이상이어야 한다. 제안하는 프로토콜은 메시지를 n이 256 이상인 비밀키로써 암호화하고 공개키로써 복호화하였다.

메시지를 n이 256 이상인 키로써 암호화하는데, n은 소수 p와 q의 곱이나 256은 2^8으로서, $p \times q = 256$을 만족하는 소수 p와 q가 없으므로 n은 256이 될 수 없고 257 이상이어야 한다. 그런데 n이 257 이상이면 평문은 256 이상으로 암호화될 수 있으므로 암호문은 8비트에 표현될 수 없을 수 있다. 따라서 제안하는 프로토콜은 8비트의 평문를 암호화할 때 16비트 암호문으로 변환되며, 암호화된 메시지의 전체 구조는 (그림 6)과 같다.

암호화된 순서번호 (16비트)	암호화된 데이터 (16비트)	암호화된 서명 (32비트)

(그림 6) 암호화된 메시지 구조

3.2.4 비인가자 접속 감지

제안하는 보안 UART는 수신 측에서 비인가자 접속을 다음과 같이 감지할 수 있다:

① 송신 측에서 적어도 8 비트를 보내며 (그림 6)과 같이 이를 암호화한 메시지 크기는 $(8+8+16)\times(16\div2)=64$비트다. 따라서 받은 메시지가 64비트 미만이면 잘못된 데이터로 확인되어 비인가자의 접속을 감지하게 된다.

② 메시지의 각 8비트는 16비트로 암호화되므로 메시지의 크기는 16의 배수다. 따라서 받은 메시지의 크기가 16의 배수가 아니면 잘못된 데이터로 확인되어 비인가자의 접속을 감지하게 된다.

③ 순서번호는 차례대로 오므로 받은 순서번호가 차례와 다르면 비인가자의 접속을 감지하게 된다.

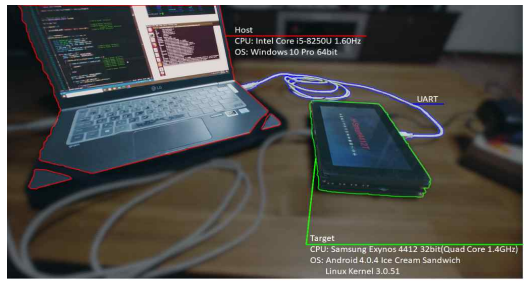
④ 디지털 서명은 순서번호와 데이터의 해시 값이다. 수신된 디지털 서명이 수신된 순서번호와 데이터의 해시 값과 다르면 잘못된 데이터로 확인되어 비인가자의 접속을 감지하게 된다.

4. 보안 UART 구현과 성능

4.1 개발환경

본 연구에서는 (그림 7)과 같은 임베디드 시스템과 랩톱 컴퓨터를 UART 케이블로 연결한 개발환경을 구축하고 임베디드 시스템의 리눅스의 UART 드라이버와 랩톱 컴퓨터의 PuTTY 터미널 에뮬레이터에 구현하였다.

실험용 임베디드 시스템은 안드로이드 4.0.4를 사용하는 태블릿 컴퓨터이며 운영체제는 리눅스 3.0.51을 사용한다. PuTTY 터미널 에뮬레이터를 설치한 컴퓨터는 윈도우 10을 사용하는 랩톱 컴퓨터이다.



(그림 7) 개발 환경

4.2 성능검증

제안하는 기법은 디지털 서명 적용 및 암호화 적용이 필요하며 전송하는 데이터 크기가 커지게 되어 전송시간에 부담을 준다. 본 절에서는 전송시간의 관점에서 성능을 평가하기 위하여 호스트의 터미널 에뮬레이터에서 임베디드 시스템의 리눅스 UART 드라이버로 메시지 송신에 걸리는 시간을 측정하였다. 송신하는 데이터 크기에 따른 시간 부담을 확인하기 위하여 데이터 크기를 변화시키며 전송 시간을 측정하였다.

<표 2>는 성능검증 결과를 보여주고 있다. 이때, 사용한 인자의 의미는 다음과 같다.

- a : 기법을 적용하지 않을 때 걸린 시간
- b : 디지털 서명까지 적용했을 때 걸린 시간
- c : 디지털 서명과 암호화까지 적용했을 때 걸린 시간

<표 2> 성능검증 결과 (단위: 초)

구분 / 데이터 크기	평문 전송 (a)	디지털 서명 적용 (b)	서명 + 암호화 적용 (c)	서명 적용 오버헤드 (b/a)	서명 + 암호화 적용 오버헤드 (c/a)
1 B	6.9E-5	0.001	0.002	18.4	22.4
32 B	0.002	0.005	0.027	2.11	12.31
1 KB	0.07	0.13	0.78	1.87	11.00
32 KB	2.28	4.36	25.02	1.91	10.99
1 MB	72.82	136.39	812.97	1.87	11.16
32 MB	2330.17	4281.24	26429.36	1.84	11.34

<표 2>에서 볼 수 있듯이, 1바이트를 전송할 때 암호화까지 적용하여 걸린 시간(c)은 약 0.002초이

다. 사람이 한 글자를 입력할 때 걸리는 평균 시간은 약 0.25초(약 240타/분)이므로, 1바이트 전송 시간(c)을 사람이 거의 느끼지 못한다. 1바이트 전송을 제외한 나머지 경우에는 전송되는 데이터 크기와 상관없이 서명 적용의 시간 오버헤드(b/a)는 약 2배이었으며, 암호화까지 적용했을 때 시간 오버헤드(c/a)는 약 11배로 나타났다. 전송 데이터의 크기가 클 때 전송 시간은 상당히 오래 걸린다.

그러나 UART를 이용한 터미널 에뮬레이터의 사용은 대부분 수십 바이트 이내의 명령어 입력을 위해 이루어진다. 위 표에서 32바이트를 전송할 때 암호화까지 적용하여 걸린 시간(c)은 약 0.027초로 사람이 인지하기 어려운 수준이다. 따라서 본 연구에서 제안한 보안 UART 기법은 일반적인 사용자 환경에서 시간적인 성능에 미치는 영향은 미미하다.

4.3 선행 연구와의 비교

UART 보안 기법에 대한 선행 연구는 많지 않다. [1]에서는 UART를 사용하지 못하도록 UART 포트를 물리적으로 제거하거나 점퍼(jumper) 설정으로 비활성화하는 방법을 소개하였고, [2]에서는 UART 통신 데이터의 스니핑을 막기 위해 통신 데이터 암호화 방법을 제안하였다.

UART 포트 제거 방법은 개발이 끝나고 제품 출시 전에 PCB 보드에서 물리적으로 포트를 제거하는 방법으로, UART 포트가 없으므로 이후에는 시스템의 진단이나 디버깅 소요가 생겼을 때 불가능해진다. UART의 비활성화 방법은 점퍼 설정을 변경하여 UART를 사용할 수 없게 만드는 방법으로, 이러한 점퍼는 접근이 용이하여 해커가 설정을 활성화하게 되면 UART 통신이 가능해진다.

UART 통신 데이터를 암호화하는 방법은 통신 스니핑을 막을 수 있지만, 비인가자가 UART 접속을 하더라도 막을 수 없고, 접속한 비인가자가 통신을 분석하고 시스템에 침투할 수 있는 여지를 허용한다.

본 연구에서 제안한 방법은 선행 연구의 단점을 해결하여 시스템 진단 및 디버깅이 가능하고 비인가자의 접속을 탐지하여 차단한다. 선행 연구와의 비교 분석을 요약하면 <표 3>과 같다.

<표 3> UART 방어 기법의 비교

구분 \ 기법	제안 기법	포트 제거	포트 비활성화	통신 암호화
시스템 진단 디버깅 가능	Yes	No	No	Yes
통신 스니핑 방지	Yes	-	-	Yes
비인가자 접속 감지	Yes	-	-	No

5. 결 론

대부분의 임베디드 시스템에서 시스템 진단 및 디버깅 용도로 널리 사용되는 UART 인터페이스는 해커에 의해 시스템 메모리나 펌웨어에 접근하기 위해 쉽게 이용될 수 있다.

본 논문에서 제안한 기법은 인가된 사용자만이 UART 접속을 허용하고 비인가자의 접속은 불허하는 보안 UART 기법이다. 제안한 기법은 디지털 서명 및 암/복호화 처리, 증가한 데이터 전송으로 인한 시간적인 부담이 늘어나지만, 일반적으로 사용자가 UART를 통해 전송하는 데이터의 양이 많지 않으므로 성능에 미치는 영향은 미미하였다.

참고문헌

[1] 새로운. "하드웨어 해킹 - (7) UART". 새로운 블로그, 2019년 12월 27일, https://blog.naver.com/wnswl316/221428720820, 2021년 2월 1일 접속.

[2] C. Sadashiva and S. Sunkari, “Data Encryption and Transition by AES Algorithm with UART”, International Journal of Scientific Engineering and Technology Research, Vol. 3, No. 35, pp. 6935-6938, 2014.

[3] 이신호 외, “임베디드 시스템 펌웨어 보안을 위한 UART 인증 프로토콜”, 한국차세대컴퓨팅학회 하계학술대회, 2020.

[4] 이신호 외, “안티탬퍼링을 위한 보안 UART 프로토콜 설계”, 한국인터넷정보학회 추계학술발표대회, 2020.

〔저자소개〕

김 주 현 (Ju-hyeon Kim)
2021년 2월 명지대학교 컴퓨터공학과 학사
email : hepha981214a@gmail.com

주 영 진 (Young-jin Joo)
2020년 2월 명지대학교 컴퓨터공학과 학사
2020년 3월 ~ 명지대학교 보안경영공학과 석사과정
email : jkevin0119@gmail.com

허 아 라 (A-ra Hur)
2015년 2월 대림대학교 국제사무행정과 학사
2018년 2월 명지대학교 보안경영공학과 석사
2021년 2월 명지대학교 보안경영공학과 박사수료
email : ara7494@gmail.com

조 민 경 (Min-kyoung Cho)
1999년 2월 서울대학교 컴퓨터공학과 학사
2010년 8월 Univ. of Maryland at College Park 전산학 박사
2018년 3월 ~ 명지대학교 교수
email : mkcho@mju.ac.kr

류 연 승 (Yeon-seung Ryu)
1990년 2월 서울대학교 계산통계학과 학사
1992년 2월 서울대학교 계산통계학과 전산과학 전공 석사
1996년 8월 서울대학교 계산통계학과 전산과학 전공 박사
2003년 3월 ~ 명지대학교 교수
email : ysryu@mju.ac.kr

이 규 호 (Gyu-ho Lee)
2004년 2월 인하대학교 컴퓨터공학과 학사
2012년 2월 인하대학교 정보통신공학과 석사
2007년 10월 ~ LIG 넥스원
email : kyuho.lee@lignex1.com

장 우 현 (Woo-hyun Jang)
2010년 2월 서강대학교 컴퓨터공학과 학사
2012년 2월 서강대학교 컴퓨터공학과 석사
2012년 2월 ~ LIG넥스원
email : woohyun.jang@lignex1.com

유 재 관 (Jae-gwan Yu)
2015년 2월 고려대학교 전자정보공학과 학사
2017년 2월 성균관대학교 플랫폼소프트웨어공학과 석사
2017년 2월 ~ LIG넥스원
email : jaegwan.yu@lignex1.com