블록체인과 IPFS 기반 IoT 시스템 인증 강화 기법

이병민O
O호서대학교 컴퓨터공학부
e-mail: leebyeongmn@gmail.comO

A Scheme for Better Authentication of IoT System based-on Blockchain and IPFS

Byeong-min LeeO
ODept. of Computer Engineering, Hoseo University

● 요 약 ●

사물 인터넷 시스템에서 서버와 클라이언트의 신뢰성 확보는 매우 중요하다. 대부분의 IoT 시스템에서 신뢰성 확보를 위해 인증서 기법이 사용되고 있다. 인증서 기법을 사용하는 IoT 시스템은 데이터(인증서,공개키) 탈취 및 분실 취약점이 있다. 이러한 취약점을 강화하기 위해 서버와 클라이언트는 서로 주고받는 데이터의 무결성과 신뢰성을 보장할 수 있는 환경이 구축되어야 한다.

본 논문에서는 이러한 취약점을 보완하기 위하여 IPFS와 블록체인을 결합한 인증 강화 기법을 제시한다. 제시한 기법의 기본 개념은 IPFS를 이용하여 CA와 서버의 인증서와 공개키를 분산 저장하고, IPFS에 저장한 인증서와 공개키의 Content-Address를 블록체인에 보관한다. 마지막으로 제시한 기법의 타당성을 검토하기 위하여 Mobius, nCube, Ethereum, IPFS를 결합한 IoT 시스템을 구축하고 SSL을 사용한 인증 과정을 실험한다.

키워드: IoT(Internet of Things), IPFS(Inter Planetary File System), 블록체인(Block Chain), 인증서(Certification)

I. Introduction

사물 인터넷(IoT:Internet of Things)은 통신 인프라의 발전과 새로운 정보통신 기술들의 등장과 함께 개념, 정의, 서비스 시나리오가 확장되고 형상이 구체화되고 있다[1]. IoT 기기와 서비스가 빠른 속도로 확대됨에 따라 개인정보 침해, 모바일 악성코드 등 보안 위협이 나타날 것으로 예상되며, 이에 대한 대비가 중요하다[2,3]. 이를 위해 서버와 클라이언트가 주고받는 데이터의 무결성과 신뢰성을 보장할 수 있는 환경이 구축되어야 한다.

서버와 클라이언트가 서로 신뢰성을 보장하기 위해 사용되는 기법에는 대표적으로 인증서 방식이 있다.

인증서 방식은 인증 과정이 간편하나 해킹이 쉽고, 사람의 개입 없이 사용되는 IoT 환경의 특성상 서버 관리의 문제점이 존재한다[4].

본 논문에서는 서버 플랫폼(Mobius)와 디바이스 플랫폼(nCube)으로 IoT 시스템을 구성하고, 인증 과정에서 필요로 한 데이터(인증서, 키)를 분산 P2P 파일시스템인 IPFS(Inter Planetary File System)와 블록체인(Ethereum)을 결합하여 데이터를 관리하고 제공하는 인증 강화 기법을 제시한다. 제시한 기법은 CA(Certificate Authority)와 서버의 데이터를 IPFS를 이용해 분산 하여 저장하고, IPFS에 저장된 데이터의 Content-Address를 Smart Contract를 이용해 블록체인에 저장 및 제공한다. 인증에 필요로 한 데이터를 분산화 함으로 분실 문제를 예방하고, 블록체인에는 허가된 개체만이 접근 할 수 있어 데이터 탈취를 예방하여 무결성과 기밀성을 제공한다.

II. Preliminaries

2. Related works

2.1 Mobius, nCube

Mobius(Server Platform)와 nCube(Device Platform)는 SKT와 전자부품 연구소가 IoT 국제 표준인 oneM2M을 기반으로 만든 IoT 플랫폼이다. Mobius는 윈도우 파일 탐색기와 같은 트리 형태의 리소스 체계를 가지고 이를 지원하고 있다. 리소스 체계는 파일 탐색기의 폴더를 접근하는 것처럼 각 리소스는 리소스를 접근할 수 있는 URI와 그대로 매핑된다[5]. Mobius는 Fig. 1.과 같이 디바이스와

어플리케이션을 연결하며, 디바이스와의 연결을 위해 HTTP, CoAP, MQTT, WebSocket 프로토콜을 지원하며, 리소스 저장을 위한 DB는 MySQL을 사용하여 구성된다[6].

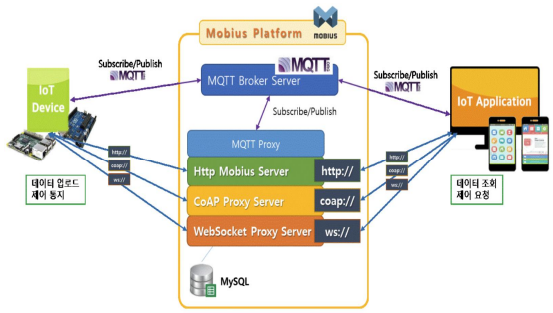

Fig. 1. Mobius Structure[6]

2.2 SSL/TLS

SSL(Secure Sockets Layser)은 컴퓨터 네트워크에 통신 보안을 제공하기 위해 설계된 암호 규약으로 Netscape사에서 만들었다. 이 규약은 인터넷 같이 TCP/IP 네트워크를 사용하는 통신에 적용되며, 통신 과정에서 전송계층 종단간 보안과 데이터 무결성을 확보해준다. SSL/TLS는 인증서와 공개키를 이용하여 CA와 서버를 인증하고 대칭키를 사용하여 서버와 클라이언트간의 비밀 통신을 제공한다.

2.3 Ethereum

블록체인의 사전적 정의는 '누구나 열람할 수 있는 디지털 장부에 거래 내역을 투명하게 기록하고, 여러 대의 컴퓨터에 이를 복제해 저장하는 '분산 형 데이터 저장기술'이다[7]. 블록체인은 합의 알고리즘을 이용해 장부 역할을 하는 블록이 체인처럼 연결되고 각 블록은 Fig. 2.과 같이 이전 블록의 해시 값을 가져 장부의 위조나 변조가 불가능하다.

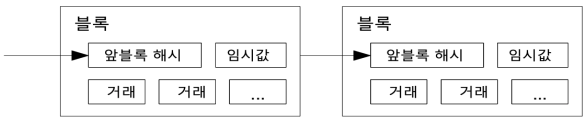

Fig. 2. Block Chain block structure[8]

이더리움은 2015년 Vitalik Buterin에 의해 제안되어 블록체인에 화폐 거래 기록뿐만 아니라 계약서(Smart Contract)의 추가 정보를 기록하도록 하여 SNS, 이메일, 전자투표 등 다양한 정보를 기록하는 시스템을 구축할 수 있다[9].

2.4 IPFS

IPFS는 HTTP Web 서버-클라이언트 환경에서 서버의 고장으로 인한 데이터 분실 및 삭제의 불안정성, 대용량 파일 다운로드의 요청으로부터 응답하는 비효율성, 고도의 중앙화 등 문제를 해결하려 만들어진 분산 P2P 파일 시스템이다. IPFS에 사용되는 기술에는 Bittorrent, DHT(Distributed Hash Tables), Merkle DAG이 있다. Bittorrent는 P2P File Exchange 프로토콜로 파일을 효율적으로 전달하고 DHT를 이용해 파일의 중복을 제거하며 파일을 검색하고 Merkle DAG을 통해 파일 데이터의 무결성을 보장한다.

III. The Proposed Scheme

3.1 Proposed Scheme

제시하는 기법을 실험하기 위해 IPFS와 블록체인을 결합한 IoT 시스템을 구성하고 Fig. 3.과 같이 설계한다.

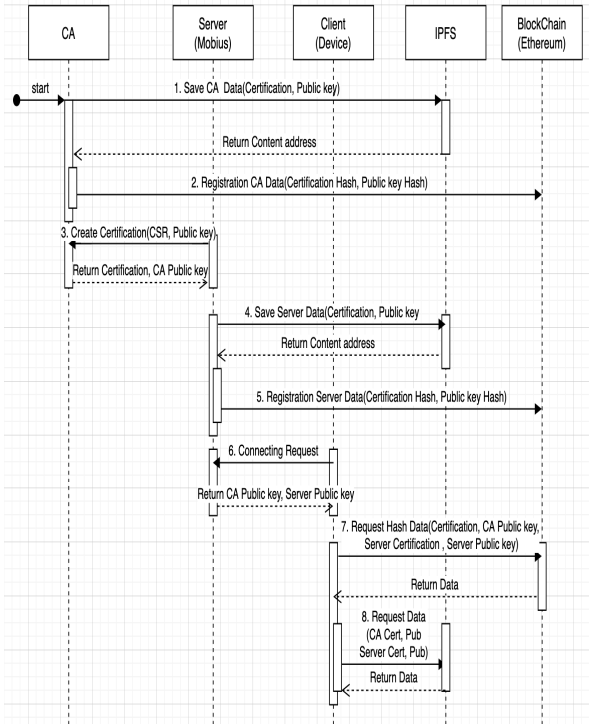

Fig. 3. Sequence Diagram of Proposed Scheme

Fig. 3.에서 CA는 인증서와 공개키를 IPFS에 저장하고 저장 위치를 찾을 수 있는 Content-Address를 받는다(①). CA는 Ethereum의 Smart Contract에 Content-Address를 등록한다(②). Mobius는 CA에게 인증서 생성을 위해 CSR(Certificate Signning Request)과 자신의 공개키를 보내 인증서 생성을 요청한다(③). CA는 Mobius의 인증서를 생성하고 자신이 인증한 인증서임을 보이기 위해 자신의 개인키로 암호화 해 반환한다. Mobius는 생성한 인증서와 자신의 공개키를 IPFS에 저장하고 Content-Address를 받는다(④). Mobius는 Content-Address를 Ethereum의 Smart Contract에 등록한다(⑤). nCube는 Mobius에게 연결 요청을 하면 Mobius로부터 CA와 Mobius의 Ethereum Account Address를 받는다(⑥). nCube는 Ethereum의 Smart Contract에 접근하여 Account Address를 이용해 CA와 Mobius의 인증서와 공개키 Content-Address를 획득한다(⑦). nCube는 Content-Address로 IPFS로부터 CA와 Mobius의 인증서와 공개키 원본을 획득한다(⑧).

3.2 Experiment and Result

제시한 기법의 타당성을 검토하기 위한 실험 환경은 Table 1.과 같이 구성한다.

Table 1. Experiment Environment

	IoT Server	IoT Device	IPFS	Blockchain
Utilities	Mobius (2.0)	nCube (Rosemary)	IPFS (0.4.17)	Ethereum (Ganache)
OS	Ubuntu 18.04	Raspbian	Ubuntu 18.04	Ubuntu 18.04
IP	192.168.0.8	192.168.0.12	–	–

Ethereum에 접속하기 위한 CA, Mobius, nCube의 Accounts와 Smart Contract를 이용하기 위한 Account는 Table 2.와 같다.

Table 2. Ethereum Accounts

	Account
CA	0x0d12888eC0F63d86fA9bCa51C801Fb3eFa6A2c92
Mobius	0xB136DAb3b7Cffa5B78eF547378Dea5ab8feD58De
nCube	0xf881DFC38D15EC2415f25a0e68f9E130D04b7A79
Smart Contract	0x5cead976e6bea64f85c9875779fe4d5c68fe3dcf

우선 CA가 개인키로 Mobius의 CSR을 서명해 Mobius의 인증서를 생성한다. CA와 Mobius의 인증서와 공개키를 각각 IPFS에 파일을 업로드 하고 Fig. 4.와 같이 Content-Address를 돌려받는 결과를 확인한다.

```
ca_crt hash : QmYcnL73wiuKY4bKGYVC4j6pwq9bnbPLj4gcVuvjkY2iVt
ca_key hash : QmdsVhbAuGiqEY7caEEnKZBAuccKw5V1f9qSjjdUm6cXb2
server_crt hash : QmPSJAgzG9R2GToRgxT2tX3QZzEeMPFPhgaRh7mWDuKzN8
server_key hash : QmWNYsjKPfEX8zPMxni7ku9xzNyqYqtQWkfsKaAD1NejzE
```

Fig. 4. CA,Mobius Certifiacation, Publickey Content-Address

CA와 Mobius는 Smart Contract를 이용하여 Fig. 5.와 같이 Content-Address를 Ethereum에 저장한다. CA와 Mobius의 인증서와 공개키 Content-Address가 Smart Contract를 이용해 블록체인에 저장된다. Table 2.의 Account 정보가 Fig. 6.처럼 보내는 쪽(from)과 받는 쪽(To) Account 정보와 일치하는지 비교하여 CA와 Mobius가 정상적으로 Content-Address를 블록체인에 저장함을 확인한다.

```
mapping (address=>CA) CAs;
mapping (address=>Server) Servers;

//regist CA Certification, publickey Content-Address
function regist_CA(string memory crt, string memory pub) public {
    CAs[msg.sender].ca_crt = crt;
    CAs[msg.sender].ca_pub = pub;
}

//regist Server Certification, publickey Content-Address
function regist_Server(string memory crt, string memory pub) public {
    Servers[msg.sender].server_crt = crt;
    Servers[msg.sender].server_pub = pub;

}
```

Fig. 5. CA, Mobius Register function in Smart Contract

```
{ transactionHash: '0xa34d9561ebb4ad7eaf07b807db90a05c27fdea8c5e96fccf5ef5793a0f984d1d',
  transactionIndex: 0,
  blockHash: '0xd2c370e31eb41f3f833b72bb8c59ac0ebaadf4c6c0a9b89a784703783f571ad5',
  blockNumber: 170,
  from: '0xb136dab3b7cffa5b78ef547378dea5ab8fed58de',
  to: '0x5cead976e6bea64f85c9875779fe4d5c68fe3dcf',
  gasUsed: 32397,
  cumulativeGasUsed: 32397,
  contractAddress: null,
  status: true,
{ transactionHash: '0x7f33839449a9691bb68bd6f10f4bf9f00c6b2cbc4154fbe7df8b1cca74e5d480',
  transactionIndex: 0,
  blockHash: '0x39504cdbe228cc608a5bada3c5e4aab13fcb173b8149f206a81df31f28256503',
  blockNumber: 171,
  from: '0x0d12888ec0f63d86fa9bca51c801fb3efa6a2c92',
  to: '0x5cead976e6bea64f85c9875779fe4d5c68fe3dcf',
  gasUsed: 32353,
  cumulativeGasUsed: 32353,
  contractAddress: null,
  status: true,
```

Fig. 6. Transcation send to Smart Contract

다음으로 nCube는 CA와 Mobius의 Ethereum Accounts를 Mobius로부터 얻는다. nCube는 Fig. 7.과 같이 Ethereum Accounts 정보를 이용하여 Content-Address를 Ethereum으로부터 얻고, CA와 Mobius의 Content-Address를 이용해 인증서와 공개키를 IPFS로부터 다운받는다.

```
//ethereum smartcontract get_Server_crt();
myContract.methods.get_Server_crt(serverAccount).call().then((result)=>{
        serverCrtHash = result;
}).then(()=>{console.log(`get_Server_crt() : ${serverCrtHash}`)});

//ethereum smartcontract get_Server_pub();
myContract.methods.get_Server_pub(serverAccount).call().then((result)=>{
        serverKeyHash = result;
}).then(()=>{
        console.log(`get_Server_pub() : ${serverKeyHash}`)

}).then(()=>{
        //ipfs, download files
        ipfsDownloadFile(name, serverCrtHash, serverKeyHash);
});


//ethereum smartcontract get_CA_crt();
myContract.methods.get_CA_crt(caAccount).call().then((result)=>{
        caCrtHash = result;
}).then(()=>{console.log(`get_CA_crt() : ${caCrtHash}`)});

//ethereum smartcontract get_CA_pub();
myContract.methods.get_CA_pub(caAccount).call().then((result)=>{
        caKeyHash = result;
}).then(()=>{
        console.log(`get_CA_pub() : ${caKeyHash}`)
}).then(()=>{
        //ipfs, download files
        ipfsDownloadFile(name, caCrtHash, caKeyHash);
});
```

Fig. 7. Get CA, Mobius Certification, Public Key

마지막으로 제시한 기법의 타당성을 검토하기 위하여 패킷 분석 툴(Wireshark)을 이용하여 nCube와 Mobius의 인증 과정을 확인한다.

192.168.0.12	192.168.0.8	TLSv1.2	576 Client Hello
192.168.0.8	192.168.0.12	TLSv1.2	267 Server Hello, Change Cipher Spec, Encrypted Handshake Message
192.168.0.12	192.168.0.8	TLSv1.2	477 Change Cipher Spec, Encrypted Handshake Message, Application
192.168.0.8	192.168.0.12	TLSv1.2	97 Encrypted Alert
192.168.0.12	192.168.0.8	TLSv1.2	97 Encrypted Alert

Fig. 8. Mobius, nCube SSL Communication

SSL 프로토콜을 사용하는 서버와 클라이언트는 연결 초기에 SSL Handshake를 수행해 인증 방식, 암호화 방식 등을 결정한다. Fig. 8.과 같이 SSL Handshake가 문제없이 이루어졌다면 인증과정이 정상적으로 수행됨을 확인할 수 있다.

IV. Conclusions

IoT 시스템에서 인증을 위한 SSL 통신과정 중 인증서가 탈취되거나 분실된다면 인증서안의 정보들이 쉽게 노출될 우려가 있다. 본 논문은 IoT 시스템에서의 인증 과정 중 발생할 수 있는 취약점을 강화하기 위하여 IPFS와 블록체인을 기반으로 한 기법을 제시하였다. 제시한 기법의 기본 개념은 CA와 서버의 인증서와 공개키를 IPFS에 분산 저장하고, 블록체인에서 Content-Address를 관리함으로써 허락된 개체만이 인증서와 공개키에 접근 가능하도록 한다.

향후 과제로는 IoT 시스템을 위해 블록체인의 운영비용을 감소시킬 수 있는 경량형 블록체인의 연구와 인증 프로토콜의 연구가 필요하다.

REFERENCES

[1] 한국정보통신기술협회, 정보통신용어사전 (URL:http://terms.tta.or.kr)

[2] D.H. Kim, S.W Yoon, and Y.P Lee, "Security for IoT Service, journal of The Korean Institute of Communication Sciences, vol. 30, no.8, pp.53-59, July 2013.

[3] J.U. Kim and S.H. Jin, "Internet (IoT) Security Technology for Security Threats in Hyper-Connection Environment," journal of The Korean Institute of Communication Sciences, vol. 34, no.3, pp.57-64, Feb. 2017.

[4] J.Y. Lee and S.P. Hong, "A study on the Authentication Method of IoT Devices Based on Blockchain," journal of Security Engineering, vol. 15, no.5, pp.299-308, Oct. 2018

[5][6] "IOTKETI/Mobius: oneM2M IoT Server Platform."Github. last modified Nov 16, 2020, accessed DEC 20 https://github.com/IoTKETI/Mobius

[7] "What innovation will the blockchain bring?."Electronics and Telecommunications Research Institute. last modified DEC 23, 2020, accessed DEC 23 https://www.etri.re.kr/webzine/20190329/sub01.html

[8] Satoshi Nakamoto, "Bitcoin: A peer-to-Peer Electronic Cash System", White Paper, Aug. 2008

[9] J.S. Park and J.Y. Park and S.M. Choi and J.T. Oh and K.Y. Kim, "Past, Present and Future of Blockchain Technology," journal of Electronics and telecommunications trends, vol. 33, no.6, pp.139-153, 2018