J. Internet Comput. Serv.
ISSN 1598-0170 (Print) / ISSN 2287-1136 (Online)
http://www.jics.or.kr

메타마스크와 연동한 블록체인 기반 사용자 인증모델☆

A Blockchain-based User Authentication Model Using MetaMask

최 낙 훈[1] 김 희 열[1*]
Nakhoon Choi Heeyoul Kim

요 약

본 논문은 사용자 인증과 개인정보의 관리를 위해 중앙집중화된 서버를 사용하는 서비스 제공자들의 개인정보 탈취와 그로인한 개인정보 도용 문제의 해결을 위한 새로운 인증 모델을 제안한다. 탈중앙화 플랫폼인 블록체인을 통해 사용자 인증과 정보저장공간을 제공해 중앙집중화 이슈를 해결하며, 사용자별 대칭키 암호화를 통해 정보의 기밀성을 보장한다. 제안 모델은 퍼블릭 블록체인인 이더리움과 웹기반의 지갑 확장프로그램인 메타마스크를 이용해 구현되었으며, 사용자는 브라우저상에서 메타마스크를 통해 이더리움 메인네트워크에 접속해 스마트 컨트랙트에 암호화한 개인정보를 저장한다. 향후 사용자는 새로운 서비스 이용을 위해 자신의 이더리움 계정을 통해 서비스 제공자에게 개인정보를 제공하며, 이 과정은 가입이나 새로운 인증과정 없이 사용자 인증과 개인정보를 제공한다. 서비스 제공자는 별도의 인증 방식 및 개인정보의 저장을 위한 비용을 절감하고 개인정보유출로 인한 문제를 방지할 수 있다.

☞ 주제어 : 블록체인, 이더리움, 인증, 메타마스크, 개인정보, 탈중앙화

ABSTRACT

This paper proposes a new authentication model to solve the problem of personal information takeover and personal information theft by service providers using centralized servers for user authentication and management of personal information. The centralization issue is resolved by providing user authentication and information storage space through a decentralize platform, blockchain, and ensuring confidentiality of information through user-specific symmetric key encryption. The proposed model was implemented using the public-blockchain Ethereum and the web-based wallet extension MetaMask, and users access the Ethereum main network through the MetaMask on their browser and store their encrypted personal information in the Smart Contract. In the future, users will provide their personal information to the service provider through their Ethereum Account for the use of the new service, which will provide user authentication and personal information without subscription or a new authentication process. Service providers can reduce the costs of storing personal information and separate authentication methods, and prevent problems caused by personal information leakage.

☞ keyword : Blockchain, Ethereum, Authentication, MetaMask, Personal Imformation, Decentralize

1. 서 론

현재 다수의 서비스 제공자는 사용자 인증 및 개인정보의 관리를 위해 서버-클라이언트 구조의 중앙형(Centralized) 모델을 이용하고 있다. 사용자는 서비스를 사용하기 위해 가입과정을 통해 개인정보를 해당 서비스 제공자에게 제공하고, 이들은 여러 방식으로 사용자의 정보를 저장, 관리한다. 하지만 서비스 제공자의 보안성은 높지 않으며, 서비스 제공자 자체를 신뢰할 수 없는 경우도 많다. 또한 서비스가 제공하는 사용자 인증과정은 필요 이상의 정보를 요구하거나 복잡한 경우가 많다.

위 문제 해결을 위해 google 등의 대형 플랫폼들은 자신의 신뢰성을 기반으로 OAuth[1] 인증을 제공한다. 하지만 OAuth 또한 이를 제공하는 플랫폼에 미리 가입되어 있어야 하며, 이 서버에 모든 정보가 저장되어 공격으로 인한 대규모 개인정보 유출에 취약하고, OAuth 서버에게 필요 이상의 개인 정보가 노출된다.

자기주권신원(Self-Sovereign ID)은 사용자의 개인정보를 특정 서버가 통제하는 것이 아닌 자신이 정보소유권

1 Department of Computer Science, kyonggi University, Suwon, 16227, Korea.
* Corresponding author (heeyoul.kim@kgu.ac.kr)
[Received 29 August 2019, Reviewed 04 October 2019, Accepted 02 December 2019]
☆ 본 연구는 경기도의 경기도 지역협력연구센터 사업의 일환으로 수행하였음. [GRRC경기2017-B04, 영상 및 네트워크 기반 지능정보 제조 서비스 연구]

을 가지고 요청에 맞추어 제공한다는 새로운 접근방식으로 설계되었다. 블록체인의 탈중앙화가 자기주권신원의 기반이 되며, Sovrin[2], uPort[3] 등이 개발되고 있다. 하지만 위 방식들은 별도의 ID 관리 및 인증 방식이 필요하며, 별도의 블록체인을 사용하는 구조로 구현상의 어려움이 있으며 사용자 접근성과 인지도가 떨어져 아직까지 확산되지 못하고 있다.

MetaMask One-click Login[4]은 OAuth 등의 개인정보 노출문제를 해결하기 위해 메타마스크에 등록된 계정을 기반으로 인증을 제공한다. 하지만 위 방식은 블록체인을 통해 개인정보를 제공할 수 없어 사용자가 서비스 등록과정에서 개인정보를 직접 제공해야 하며, 블록체인 Account의 소유만을 증명하는 문제가 있다.

본 논문의 제안모델은 서비스 제공자가 서버를 통해 인증을 제공하는 중앙화 모델의 신뢰성 문제와 정보관리의 문제점을 해결하고자 퍼블릭 블록체인을 사용해 사용자 인증과 개인정보의 저장을 대리수행한다. 알려진 퍼블릭 블록체인을 사용함으로서 사용자 접근성과 인지도를 확보하며, 기존 방식과 달리 별도의 ID의 관리와 번거로운 인증 방식이 불필요해 사용의 간편성을 보장한다. 또한 사용자별 대칭키 암호화를 통해 OAuth 등의 개인정보노출 문제를 해결한다. 제안모델을 통해 사용자는 자신의 개인정보와 인증정보를 서비스에 위탁하는 것이 아닌 자신이 직접 관리하고 새로운 서비스에 재차 인증을 수행하지 않는다. 각 사용자는 이더리움 계정(Account)의 소유로 블록체인에 자신의 개인정보를 등록해 이더리움 계정만으로 인증을 수행해 이를 자신의 신분증과 같이 사용할 수 있다.

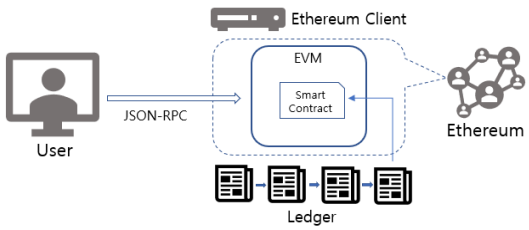
2. 관련연구

우리는 기존의 중앙화된 서비스 구조를 벗어나기 위해 블록체인의 탈중앙화(Decentralized) 특성을 이용하며, 블록체인에 대한 사용자의 접근성을 높이기 위해 메타마스크를 이용한다. 제안모델 설명에 앞서 먼저 설명한다.

2.1 이더리움(Ethereum)[5]

제안모델의 탈중앙성을 보장하기 위해 블록체인을 이용하며, 작업증명(PoW, Proof of Work)[6]으로 대표되는 합의알고리즘을 사용하는 퍼블릭 블록체인을 이용한다. 블록체인은 네트워크상의 서로 신뢰할 수 없는 참여자간의 합의로 블록에 포함된 정보의 무결성을 보장하고 블록 생성자를 선정해 관리자가 없는 탈중앙화된 네트워크를 지속한다.

우리는 제안모델에 퍼블릭 블록체인 플랫폼인 이더리움을 이용한다. 간단한 연산만 가능한 튜링 불완전한 스크립트를 지원하는 비트코인 등과 달리 이더리움은 ĐApp(Decentralized Application) 개발을 위한 스마트 컨트랙트(Smart Contract)[7]의 작성과 배포를 지원한다. 튜링 완전한 언어인 솔리디티(Solidity)로 작성되어 분산원장에 기록된 스마트 컨트랙트는 자동화된 실행과 결과의 무결성을 보장한다.



(그림 1) 이더리움 스마트 컨트랙트의 동작
(Figure 1) Operation of Ethereum Smart Contract

솔리디티로 작성된 스마트 컨트랙트는 컴파일된 후 이더리움 가상머신 바이트코드(EVM Byte Code)의 형태로 합의원장(Ledger)에 저장되어 계약내용의 무결성이 보장된다. 이더리움 클라이언트(Geth)는 스마트 컨트랙트의 바이트코드를 호출해 EVM(Ethereum Virtual Machine)환경에서 실행시켜 계약내용에 따른 자동화된 실행과 그 결과의 무결성이 보장된다. 아래에서 설명할 메타마스크를 통해 일반적으로 접근할 수 있는 웹사이트 상에서 이더리움 클라이언트 없이 스마트 컨트랙트를 동작시켜 인증과 정보저장을 탈중앙화된 방식으로 대체한다.

2.2 ECDSA[8]

이더리움은 계정 소유자에 대한 인증과 트랜잭션의 인증 및 무결성 검증을 위해 ECDSA(Elliptic Curve Digital Signature Algorithm) 전자서명을 사용한다. ECDSA는 타원곡선상의 이산대수문제인 ECDLP(Elliptic Curve Discrete Logarithm Problem)에 안전성을 기반하고 있으며, 현실적으로 개인키의 추측이나 전자서명의 위변조가 불가능하다. 또한 ECDSA와 같은 타원곡선암호(ECC)는 다른 암호화 알고리즘과 비교해 더 작은 bit수로 높은 보안

성을 제공하는 장점을 가진다.

이더리움에서는 타원곡선 secp256k1을 사용하며, 다음의 전자서명 방식을 사용한다.

- 키생성: KeyGen
  ① [0..n-1](n: order) 내의 임의의 개인키 d를 생성한다.
  ② 타원곡선 곱셈 연산을 통해 공개키 Q=dG (G: generator)를 계산한다.

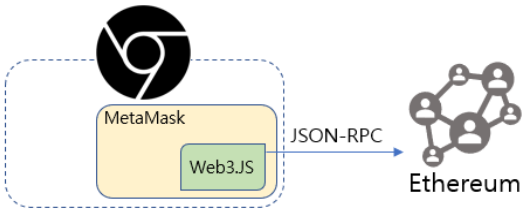
- 서명생성: Sign(m, k)
  ① 서명하려는 메시지 m의 해쉬값 h=hash(m)을 계산하고, h의 최하위 n비트가 z가 된다.
  ② [0..n-1] 내의 임의의 수 k를 생성한다.
  ③ R=kG를 계산하고, 포인트 R의 x좌표값 r을 획득한다.
  ④ $s=k^{-1}(z+rd) \bmod n$을 계산한다.
  ⑤ v=27+(y%2) (y: Q의 y좌표값)을 계산한다.
  ⑥ (r, s, v)가 전자서명 값이다.

- 서명검증: Verify(m, r, s, v)
  ① 메시지 m의 해쉬값 h=hash(m)을 계산하고, h의 최하위 n비트 z를 얻는다.
  ② $u_1=zs^{-1} \bmod n$, $u_2=rs^{-1} \bmod n$을 계산한다.
  ③ r과 v를 이용해 공개키 Q를 복구한다.
  ④ $(x,y)=u_1G+u_2Q$을 계산한다.
  ⑤ x=r mod n이면 올바른 서명이다.

2.3 메타마스크(MetaMask)[9]

블록체인 지갑(wallet)은 블록체인 계정의 잔고를 관리하는 것만이 아닌 사용자의 개인키를 저장하고, 사용자를 대신해 트랜잭션을 생성해 Broadcast할 수 있다. 이런 의미에서 지갑은 일반사용자의 블록체인 네트워크로의 가장 쉬운 접근방식이라 할 수 있다.

메타마스크는 브라우저에서 실행되는 가장 대표적인 웹기반의 지갑 확장프로그램으로 현재 Chrome, Firefox, Opera, Brave Browser를 지원하고 있으며, 이더리움 계정 생성 및 연동, 전자서명, 트랜잭션생성 등을 지원한다. 사용자는 브라우저상에서 메타마스크를 이용해 블록체인 상의 스마트 컨트랙트에 손쉽게 접근할 수 있다.

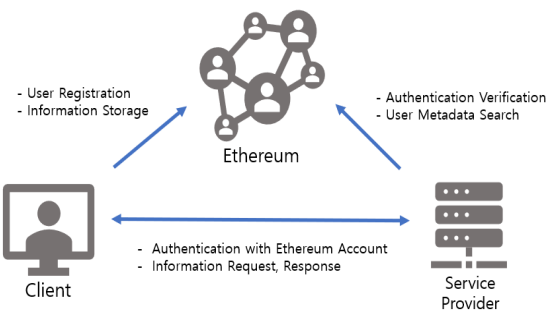


(그림 2) web3를 이용한 브라우저에서의 JSON-RPC 연결
(Figure 2) JSON-RPC Connection in Browser by web3

사용자가 이더리움 네트워크에 접근하기 위해서는 이더리움 클라이언트를 통해 블록체인 노드로서 네트워크에 참여해야한다. 이런 방식은 사용자 접근성을 낮추기 때문에 메타마스크는 지갑 사용자에게 네트워크 접속을 위한 클라우드 서비스인 Infura API[10]를 사용한다. 이를 통해 사용자는 직접 노드를 운영하지 않아도 이더리움 네트워크에 접속해 스마트 컨트랙트를 작동시키고 결과를 받아볼 수 있다. 이더리움은 웹어플리케이션에서 JSON-RPC를 이용해 이더리움 네트워크와 통신하기 위한 web3.JS 표준API를 지원한다. 메타마스크는 구동중인 브라우저에 이를 자체적으로 제공하고 있다.

3. 제안모델

3.1 제안모델 개요



(그림 3) 제안모델의 구성
(Figure 3) Composition of the Proposal Model

그림 3은 제안모델의 구성을 나타내며, 이더리움을 기반으로 사용자의 인증과 개인정보 제공이 수행된다. 제안모델에서 사용자는 자신의 이더리움 계정을 통해 스마트 컨트랙트에 접근해 자신의 암호화된 개인정보와 제공 가능한 정보의 항목인 메타데이터를 작성한다. 서비스

제공자는 사용자의 전자서명과 스마트 컨트랙트를 기반으로 인증을 수행하고, 사용자의 메타데이터를 확인해 사용자로부터 필요한 개인정보를 요청해 전달받는다.

제안모델을 통해 사용자는 이더리움 계정을 이용해 별도의 ID 생성이나 추가 인증방식 없이 브라우저를 통해 자신을 인증할 수 있으며, 블록체인에 저장된 개인정보로 서비스 제공자의 개인정보 요청을 처리할 수 있다. 또한 서비스 제공자들은 별도의 자체적인 인증 방식 없이 손쉽게 사용자 인증을 수행할 수 있다.

3.2 스마트 컨트랙트 구조

제안모델은 이더리움에 스마트 컨트랙트 기반의 정보저장공간을 구성하며, 인증 및 개인정보저장 트랜잭션의 서명에서 확인된 서명자의 이더리움 지갑주소를 식별자로 위 정보를 이더리움에 저장한다. 저장된 정보의 수정이나 삭제(미사용 처리)를 위한 트랜잭션 또한 서명을 확인해 해당 정보의 소유주임을 확인하여 실행된다. 스마트 컨트랙트 자체는 입력 및 저장데이터가 외부에 노출되어 정보의 기밀성이 보장되지 않는다. 우리는 개인정보를 사용자별 대칭키로 암호화한 후 저장해 블록체인에 저장되는 데이터의 기밀성을 보완한다.

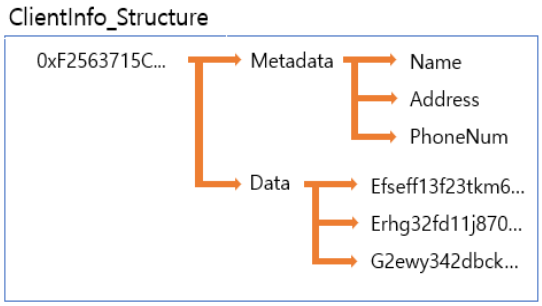
DAM(Decentralized Authentication Manager)

```
mapping (address => CIS) ClientInfo_Structure;
modifier accessControl(){…
function set_client(string memory _infoStructrue) public returns (bool){…
function modi_client_Data(address _client) public accessControl{…
function get_client_Metadata(address _client) public view returns (string memory)
```

(그림 4) 스마트 컨트랙트 DAM의 구조
(Figure 4) Structure of Smart Contract DAM

스마트 컨트랙트 DAM의 구조는 크게 그림 4와 같다. 사용자의 지갑주소를 식별자로 개인정보와 메타데이터가 저장되는 ClientInfo_Structure매핑 구조체와 정보수정의 접근제어를 위한 accessControl이 있다. 표면적으로 동작되는 함수로는 사용자의 정보 저장에 사용되는 set_client, 저장된 정보를 수정하기 위한 modi_client_Data, 그리고 서비스 제공자의 사용자 메타데이터 확인에 사용되는 get_client_Metadata로 구성된다.

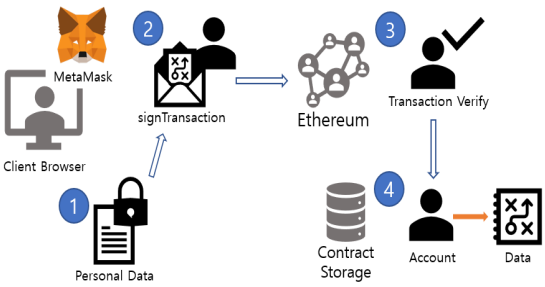
사용자는 그림 5와 같이 스마트 컨트랙트의 데이터 구조인 ClientInfo_Structure를 통해 개인정보의 메타데이터와 암호화된 개인정보를 작성한다. 메타데이터는 사용자의 인증 후 서비스 제공자가 정보제공을 요청하기 전 사용자정보의 존재확인에 사용된다. 사용자는 이를 통해 제공 가능한 정보의 범주를 공개한다.



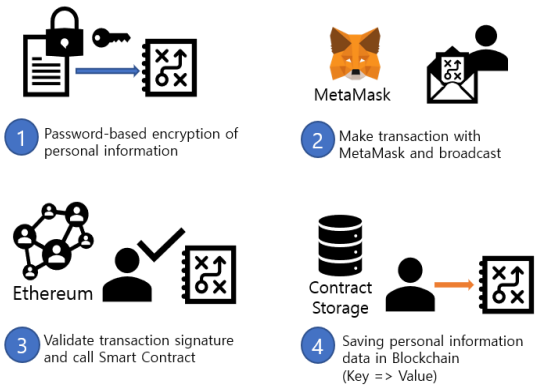
(그림 5) DAM에 저장되는 사용자 정보 구조
(Figure 5) Structure of User Data in DAM

3.3 사용자 등록 및 개인정보 등록

그림 6은 사용자가 메타마스크에 등록한 이더리움 계정으로 스마트 컨트랙트에 개인정보 및 인증정보를 등록하는 과정의 개요이며 그림 7은 그의 상세설명이다.



(그림 6) 사용자 등록 흐름도
(Figure 6) User Registration Flow



(그림 7) 사용자 등록 단계
(Figure 7) Steps of User Registration

① 블록체인상의 데이터 기밀성을 위해 사용자는 개인정보 암호화용 비밀번호를 입력하고 이를 SHA-3 해시하여 생성된 256Bit 데이터를 키로 하여 개인정보를 AES-256(Advanced Encrytion Strandard)[11]으로 대칭키 암호화한다. 저장되는 정보들은 사용자마다 다른 키로 암호화되므로 공격자의 무작위 대입공격(Brute Force)과 대규모 개인정보 유출에 면역력을 가진다.

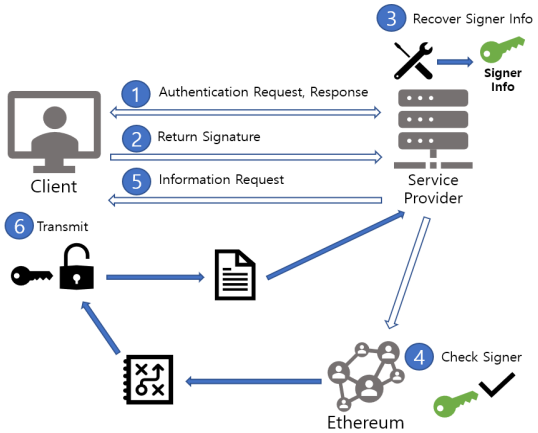
② 사용자는 스마트 컨트랙트 DAM의 set_client 함수 호출 요청과 앞서 생성한 암호화된 개인정보와 암호데이터의 메타데이터를 메타마스크를 통해 트랜잭션으로 생성해 이더리움 네트워크에 제출한다.

③ 제출된 트랜잭션이 포함된 블록이 채굴되어 블록체인에 포함될 때 이더리움은 트랜잭션과 서명의 유효성을 검증한다. 그리고 트랜잭션에 담긴 스마트 컨트랙트의 set_client의 호출을 수행한다.

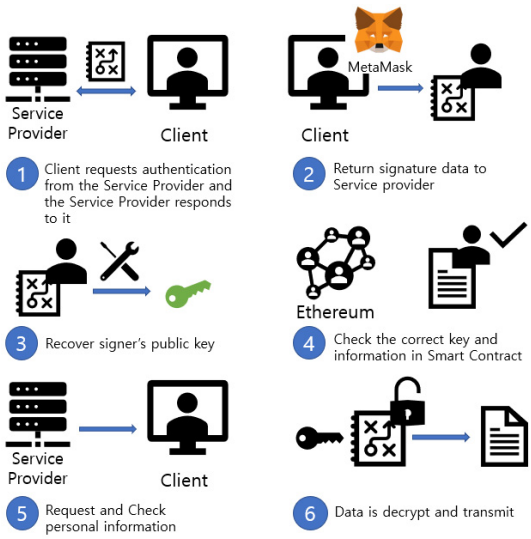
④ 스마트 컨트랙트 DAM 은 서명 검증과정에서 복원된 호출자의 지갑주소를 식별자로 전달된 정보를 그림 5와 같이 ClientInfo_Structure 에 저장한다. 저장된 정보의 수정을 위한 modi_client_Data 의 실행은 accessControl 에서 해당 요청 트랜잭션의 서명자의 지갑주소를 저장을 수행한 등록자와 비교해 권한이 부여된다.

3.4 사용자 인증 및 개인정보 제공

그림 8은 사용자 인증 및 정보 제공의 흐름을 나타내며, 그림 9는 각 단계의 상세 설명이다. 이 과정을 통해 사용자는 메타마스크와 연동해 이더리움 계정과 개인키, 전자서명을 통해 인증을 받는다. 서비스 제공자는 사용자 인증을 완료한 후 요청을 통해 사용자의 개인정보를 제공받는다.



(그림 8) 사용자 인증 흐름도
(Figure 8) User Authentication Flow



(그림 9) 사용자 인증과 개인정보 제공 단계
(Figure 9) Steps of Authentication and User Data Infomation Provision

① 사용자는 이더리움 계정을 통한 인증을 요청하고 이에 서비스 제공자는 리플레이 공격(Replay Attack)의 방지를 위한 임의의 넌스(Nonce)를 반환한다.

② 사용자는 메타마스크에 저장된 개인키로 서비스제공자에게 받은 넌스의 전자서명 (r, s, v)=Sign(Nonce, k)를 생성한다. 구체적으로는, 사용자의 브라우저에서 web3.personal.sign() 함수가 실행되면 메타마스크는 이를 감지해 사용자의 개인키를 이용해 서명을 생성한다.

③ 서비스 제공자는 사용자가 보낸 서명데이터(r, s, v)와 ①의 넌스를 입력해 Verify(Nonce, r, s, v)를 이용해 전자서명을 검증한다. 또한, 이더리움에서 지원하는 ecRecover를 통해 서명데이터로부터 사용자의 공개키 Q를 복원한다.

④ 서비스 제공자는 복원된 사용자의 공개키로 이더리움 지갑주소를 추출하고 스마트 컨트랙트 DAM의 get_client_Metadata함수를 호출해 블록체인 상에 사용자의 정보가 등록되어 있는지와 요구하는 정보의 항목이 메타데이터에 존재하는지 확인한다.

⑤ 스마트 컨트랙트에서 정보의 등록여부와 요구항목이 확인되면, 저장된 데이터의 복호화 요청과 필요한 정

보의 항목을 사용자에게 다음 시퀀스번호(Nonce+1)와 함께 요청한다. 사용자는 요청된 정보 항목을 검토하고 제공에 동의할지 결정한다.

⑥ 개인정보 제공을 위해 사용자는 스마트 컨트랙트에 저장된 암호데이터를 읽어와 저장과정에서 사용한 암호화용 비밀번호(AES 대칭키)로 데이터를 복호화한다. 이후 요청된 개인정보를 서비스제공자에게 시퀀스번호(Nonce+2)와 함께 전송한다.

4. 구현 및 실험

본 논문에서 제안한 모델의 동작을 확인하고 검증하기 위해 프로토타입을 구현하고 실험했으며, 세부사항은 아래에서 서술하고 있다.

4.1 개발환경

개발환경에서 사용자는 메타마스크가 설치된 Chrome에서 Web3.JS를 통해 이더리움 및 서비스 제공자 서버와 상호작용을 수행한다.

서비스 제공자는 일반적으로 작동하는 서버로 가정하며 Web3.JS를 통해 스마트 컨트랙트에 접근하며, 사용자와 통신을 통해 인증 및 개인정보 제공받는다. 스마트 컨트랙트는 솔리디티로 작성되어 컴파일되어 EVM바이트코드 형태로 이더리움 네트워크에 등록되며 EVM상에서 작동해 모델 전반에 걸쳐 사용된다. 사용자의 경우 메타마스크의 Infura를 통해 동작하며, 서비스 제공자는 이더리움 클라이언트를 사용하거나 Infura 서비스를 선택적으로 운용할 수 있다.

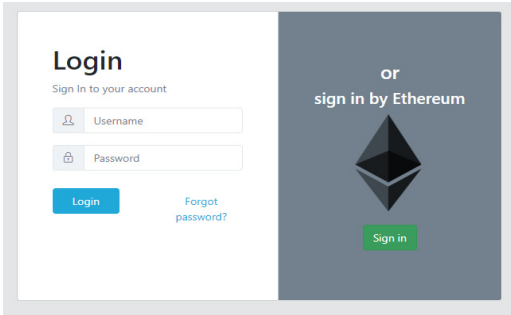
(표 1) 개발환경
(Table 1) Development Environment

User Web	Environment	Chrome Browser, MetaMask
	Language	JavaScript
Service Provider	Environment	Node Server, Windows
	Language	NodeJS
Smart Contract	Environment	Ethereum Ropsten Testnet
	Language	Solidity ^0.5.0

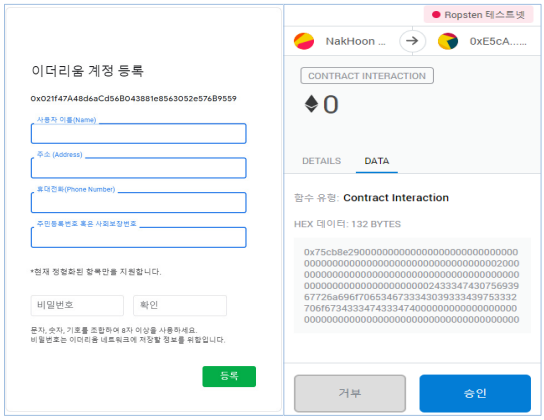
4.2 구현

4.2.1 계정 및 개인정보 등록

그림 10은 서비스에서 사용자의 로그인 화면이며, 메타마스크가 구동중인 브라우저 환경이다. 사용자가 해당 페이지에서 이더리움 로그인을 선택하면 이더리움 계정이 스마트 컨트랙트 DAM에 등록되어 있는지 확인한다. 등록되지 않은 계정이라면 계정등록과정으로 이동하게 되며, 등록된 계정이라면 서버로부터 넌스를 전송받아 로그인 과정을 수행한다.

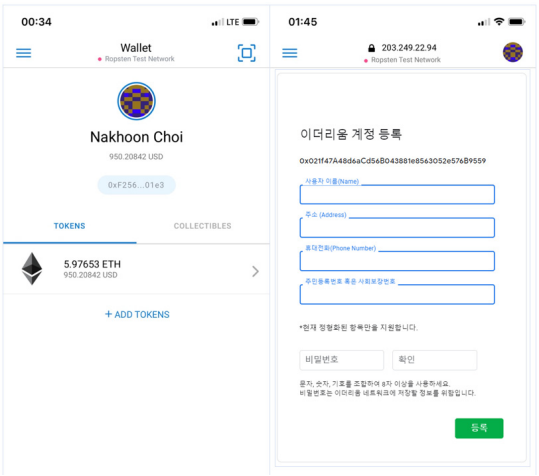


(그림 10) 사용자 초기화면
(Figure 10) User Initial Screen



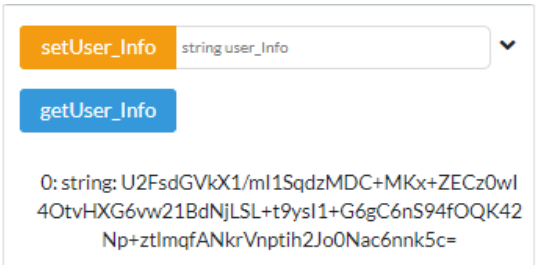
(그림 11) 이더리움 계정 등록(좌), 트랜잭션 생성(우)
(Figure 11) Ethereum Account Registration(Left), Create Transaction(Right)

그림 11은 사용자가 이더리움 계정을 이용한 로그인 기능을 사용하기 위한 계정등록 화면과 스마트 컨트랙트에 정보를 저장하기 위한 트랜잭션의 승인화면이다. 그림 11(좌)에서 입력하는 비밀번호는 저장하는 정보의 대칭키 암호화를 위한 비밀번호이며 복호화과정에서 같은 키를 사용한다. 그림 11(우)은 입력된 정보의 저장을 위한 트랜잭션 생성의 승인화면이다. 132Bytes 크기의 정보를 블록체인에 저장하는데 사용된 가스(Gas)는 65,000가량으로 한화 14원 정도이다.



(그림 12) 모바일 환경의 메타마스크
(Figure 12) MetaMask in Mobile Environment

모바일 환경에서의 동작을 확인하기 위해 메타마스크의 IOS Beta버전을 이용해 실험하였으며, 그림 12를 통해 모바일 환경에서 정상적으로 작동하는 것을 확인하였다.

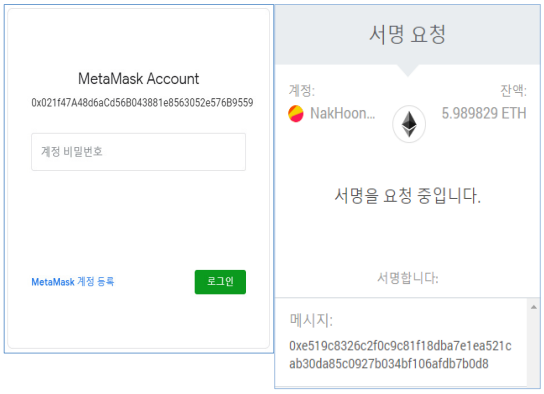


(그림 13) 리믹스 IDE
(Figure 13) REMIX IDE

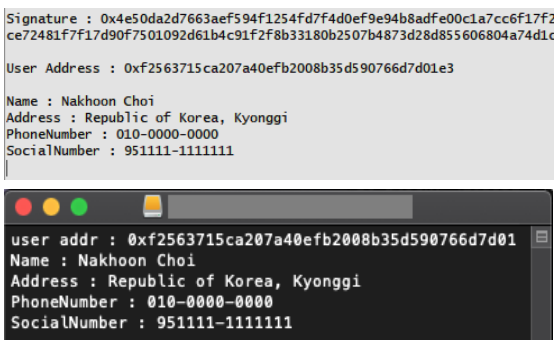
그림 13은 솔리디티 웹 IDE인 REMIX로 위 과정에서 등록한 정보가 스마트 컨트랙트에 정상적으로 암호화되어 저장된 것을 확인할 수 있다.

4.2.2 인증 및 개인정보 제공

그림 14(좌)는 사용자가 이더리움 계정 로그인을 요청한 후 생성되는 팝업으로 입력을 요구하는 계정 비밀번호는 등록과정에서 등록한 정보의 복호화용 대칭키를 말한다. 그림 14(우)는 인증과정에서 서비스 제공자가 사용자에게 전송한 넌스에 대한 서명을 승인하는 화면이다. 넌스에 서명되어 서비스 제공자에게 전송되면 서비스 제



(그림 14) 로그인 팝업화면(좌)과 넌스 서명(우)
(Figure 14) Login popup Screen(Left) and Signing a Nonce(Right)

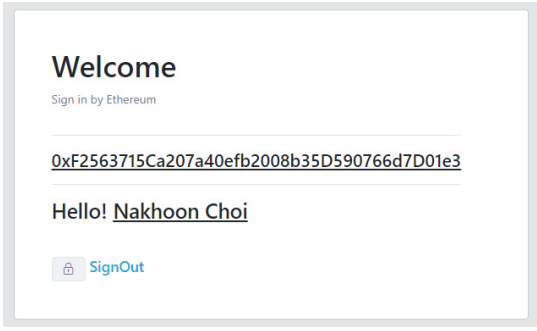


(그림 15) 서비스 제공자의 결과화면
(Figure 15) Service Provider Results screen

공자는 서명데이터로부터 공개키와 지갑주소의 복구와 함께 스마트 컨트랙트의 get_client_Metadata를 통해 사용자의 메타데이터를 확인하고 개인정보 요청을 수행한다.

서비스 제공자는 서명데이터를 기반으로 사용자의 지갑주소를 복구하고 바로 이어서 개인정보의 복호화 및 제공을 요청하고 제공받는 과정을 수행한다. 그림 15에서 Windows와 MacOS에서 운영되는 서비스가 Chrome을 이용하는 사용자 측에서 생성된 서명데이터(signature)를 받아 정상적으로 사용자의 지갑주소(User Address)를 복구한 것을 볼 수 있다. 또한 개인정보 제공과정을 통해 사용자의 개인정보를 제공받은 것을 볼 수 있다.

그림 16은 로그인 과정을 통해 서비스 제공자가 사용자의 정보를 제공받은 후 사용자가 확인할 수 있는 로그인이 완료된 모습이다.



(그림 16) 사용자 최종화면
(Figure 16) User final screen

4.3 실험

본 절에서는 메타마스크 지원 Browser인 Firefox, Opera, Brave Browser에서의 서명 알고리즘과 각 OS에 따른 ecRecover의 동작을 실험한다.

(표 2) ecRecover 정량평가
(Table 2) Quantitative evaluation of ecRecover

OS	Windows (x64)		
Browser	Chrome	Firefox	Opera
RSR*(%)	100	100	100
OS	macOS (Mojave)		
Browser	Chrome	Firefox	Opera
RSR(%)	100	100	100
OS	Ubuntu (18.04.3 LTS)		
Browser	Chrome	Firefox	Opera
RSR(%)	100	100	100

RSR(Recover Success Rate) = (match / total try) * 100

표 2는 500개의 시드(Seed)로 메타마스크 지원 브라우저에서 생성한 서명데이터와 각 운영체제에서 ecRecover를 통한 사용자 지갑주소 복구의 500회 실험결과이다.

실험결과를 통해 메타마스크의 ECDSA서명 알고리즘이 모든 브라우저에서 동일하게 동작하여 같은 서명데이터(r, s, v)를 출력하는 것을 확인할 수 있다. 또한 Windows, macOS, Linux 환경에서 ecRecover가 정상적으로 사용자의 지갑주소를 복구하는 것을 확인할 수 있다.

4.4 비교

기존의 분산형 ID(Decentralized Identifier)는 W3C의 DIDs 표준[12]에 따라 구현된다. 하지만 표준 정형화가 부족하고 Side-chain 등을 사용하는 구조로 구현상의 어려움으로 표준에서 벗어나 호환되지 않는 경우가 많다. 또한 개인정보의 저장을 위해 off-chain을 사용하는 경우, 사용자는 정보의 유지, 관리를 위한 블록체인 외부의 추가요소가 필요하다.

본 논문의 제안모델은 기존 DIDs 표준의 위와 같은 문제를 on-chain방식과 사용자별 대칭키를 통해 해결하였다. 최초 등록과정에서 개인정보를 블록체인에 암호화하여 저장하여 모델 사용에 있어 사용자 측면의 추가요소를 최소화한다. 인증을 위한 별도의 ID와 추가 인증방식을 사용자에게 부가시키지 않음으로서 사용자 접근성을 보장한다.

5. 한 계

본 논문에서 제안한 모델은 AES-256을 통해 사용자의 개인정보를 암호화하여 이더리움에 저장한다. 하지만 위 암호화 데이터는 향후 양자 컴퓨터(Quantum Computer)로 인한 컴퓨팅 성능의 향상으로 복호화 가능성이 존재한다. 양자내성암호화(Quantum-resistant cryptography)를 통해 이를 극복할 수 있지만, 표준으로 제정된 암호화 방식들은 아직까지 양자내성을 가지지 못하고 있다. 향후 양자내성암호화 방식의 표준화가 완료된다면 모델의 암호화 방식은 교체될 예정이다.

6. 결 론

지금까지 사용자들은 서비스의 이용을 위해 서비스 제공자의 서버에 가입정보와 개인정보를 저장해야 했다. 우리는 기존 중앙화 방식의 인증에서 벗어나 탈중앙화된 블록체인의 인증방식과 공유원장의 무결성을 이용한 사용자 인증과 개인정보저장 모델을 제시했다.

제안 모델을 통해 사용자는 새로운 서비스의 이용을 위해 별도의 가입과 인증 과정을 거치지 않고 이더리움 계정만을 통해 자신을 인증할 수 있으며, 개인정보 관리를 서버가 아닌 본인이 함으로서 서버 해킹으로 인한 개인정보의 대규모 유출을 방지할 수 있다. 또한 서비스 제공자는 개인정보의 저장 및 보안을 위한 비용을 절감하고 개인정보유출로 인한 문제를 방지할 수 있다.

우리가 제안하는 모델은 개인이 주도적으로 정보를 관리함으로써 사회적으로 개인정보 보안에 대한 의식을

함양하고 개인의 정보가치를 보존하며 개인정보도용의 문제를 해결할 수 있다.

참고문헌(Reference)

[1] Hardt, Dick. "The OAuth 2.0 authorization framework." , 2012.

[2] The Sovrin Foundation. "Sovrin: A Protocol and Token for Self-Sovereign Identify and Decentralized Trust." https://sovrin.org/library/sovrin-protocol-and-token-white-paper. Jan. 2018

[3] Lundkvist, C et al. "Uport: A platform for self-sovereign identity", blockchainlab.com/pdf/uPort_whitepaper_DRAFT20161020.pdf. Oct. 2016,

[4] One-click Login with Blockchain: A MetaMask Tutorial https://www.toptal.com/ethereum/one-click-login-flows-a-metamask-tutorial

[5] The Ethereum Foundation. "White Paper: A Next-Generation Smart Contract and Decentralized Application Platform" https://github.com/ethereum/wiki/wiki/White-Paper

[6] NAKAMOTO, Satoshi, et al. Bitcoin: A peer-to-peer electronic cash system. 2008.

[7] Szabo, N. "Smart contracts: building blocks for digital markets," EXTROPY: The Journal of Transhumanist Thought, 1996

[8] Danial R. L. Brown. "SEC1: Elliptic Curve Cryptography," Standard for Efficient Cryptography, pp. 47-48, May. 2009

[9] METAMASK. https://metamask.io/

[10] The Infura Inc. INFURA. https://infura.io/

[11] STANDARD, NIST-FIPS. Announcing the advanced encryption standard (AES). Federal Information Processing Standards Publication, 2001, 197.1-51: 3.3.

[12] Decentralized Identifiers (DIDs) v0.13 https://w3c-ccg.github.io/did-spec/#generic-did-parameter-names

◐ 저 자 소 개 ◑

최 낙 훈(Nakhoon Choi)
2014년~현재 경기대학교 컴퓨터과학과 학사
관심분야 : 블록체인
E-mail : skrgns0411@naver.com

김 희 열(Heeyoul Kim)
2000년 한국과학기술원 전산학과(공학사)
2002년 한국과학기술원 전산학과(공학석사)
2007년 한국과학기술원 전산학과(공학박사)
2009년~현재 경기대학교 컴퓨터공학부 부교수
관심분야 : 정보보호, 블록체인
E-mail : heeyoul.kim@kgu.ac.kr