

블록체인을 활용한 불변성 암호화 모듈 구현

윤경섭*, 김민지^o

*인하공업전문대학 컴퓨터정보과,

^o인하공업전문대학 컴퓨터정보과

e-mail: ksyoon@inhac.ac.kr*, kei07001@naver.com^o

Implementing an Immutable Encryption Module using Blockchain

Kyung Seob Yoon*, Minji Kim^o

*Dept. of Computer Science, Inha Technical College,

^oDept. of Computer Science, Inha Technical College

요약

이 시스템은 블록체인의 분산 처리 방식을 활용하여 중앙 집중식 데이터베이스에서 사용자가 관리자의 보안 인프라에 의존할 수밖에 없는 점을 보완하고, 네트워크 독립성과 데이터의 불변성, 투명성을 보장한다. 본 논문에서는 기존의 중앙 집중식 데이터베이스의 단점을 보완하는 블록체인의 분산 데이터 저장 방식을 활용하여 블록체인의 해시 기법과 작업 증명, 탈중앙화, 합의 알고리즘을 구현한 암호화 시스템을 제안한다.

키워드: 블록체인(Blockchain), 암호화(Encryption), 작업 증명(Proof of Work), 해시(Hash), 합의 알고리즘(Consensus Algorithm)

I. Introduction

기존의 중앙 집중식 데이터베이스는 클라이언트 - 서버 네트워크 아키텍처를 사용한다. 데이터베이스의 통제권은 지정된 관리자에게 있으며, 클라이언트 자격을 인증한 후 데이터베이스에 접근을 허용한다. 데이터베이스의 관리 책임은 관리자에게 있으므로, 관리자의 보안에 문제가 발생하면 데이터베이스가 변경되거나 삭제될 수 있다. 이는 관리자가 원하는 방식으로 데이터베이스를 관리, 수정 및 제어할 수 있고 관리자 없이는 데이터베이스가 전혀 작동하지 않음을 의미한다[1].


본 논문에서는 이러한 중앙 집중화 데이터베이스의 단점을 보완하기 위해 블록체인(Blockchain)의 분산 데이터 저장 방식이 제공하는 특이성인 네트워크 독립성과 불변성을 활용하고자 하였다.

II. Preliminaries

2. Related works

2.1 블록체인과 데이터베이스의 차이

Fig. 1은 블록체인과 데이터베이스의 차이점을 나타낸 것이다[2].



Properties	Blockchain	Traditional Database
Operations	Only Insert Operations	Can perform C.R.U.D. Operations
Replication	Full Replication of block on every peer	<ul style="list-style-type: none"> Master-Slave Multi-master
Consensus	Majority of peers agree on the outcome of transactions	Distributed transactions (2 Phase Commit)
Invariants	Anybody can validate transactions across the network	Integrity Constraints

Fig. 1. Difference between a Blockchain and a Database

블록체인과 중앙 집중식 데이터베이스의 가장 큰 차이점은 데이터베이스는 중앙 집중적 형태인 반면, 블록체인은 분산 환경을 제공한다. 중앙 집중식 데이터베이스는 접근 권한만 있으면 누구나 데이터베이스를 파괴하거나 변경할 수 있으므로, 사용자는 데이터베이스 관리자의 보안 인프라에 의존할 수밖에 없다. 이에 반해 블록체인은 네트워크가 독립적으로 작동하고 중앙 집중식 제어 없이도 작동한다. 또한, 블록체인은 불변성을 지원하므로 한 번 기록된 데이터는 지우거나 대체할 수 없다. 기존 데이터베이스는 불변성을 나타내지 않으므로 불량 관리자 또는 타사 해킹에 의해 조작되기 쉽다[1].

2.2 암호화 방식

블록체인에서 가장 많이 채택하여 사용되고 있는 암호화 방식인 SHA-256은 어떤 길이의 값을 입력하더라도 256비트의 고정된 문자열을 반환한다. 입력값이 조금만 변해도 출력값이 완전히 달라지므로 출력값을 토대로 입력값을 유추하는 것은 불가능하다. 또한, 출력 속도가 빠르다는 장점을 가지고 있으며, 평문을 암호화했을 때 다시 복호화할 수 없는 단방향성의 성질을 띠고 있다. 대표적인 해시(Hash) 알고리즘인 MD5의 해시 값과 비교해 2배 긴 해시 값을 반환한다. 단순히 산술적으로 계산하면 256비트는 2의 256제곱만큼의 경우의 수를 만들 수 있고 이는 무차별 대입 방식을 사용하여 해당되는 값을 찾아내는 데 많은 시간이 소요되게 하므로 안전하다고 볼 수 있다[3].

본 논문에서는 해시 값을 도출하기 위해 SHA-256 방식을 사용하였다.

III. Design

3.1 블록체인 구조

```

{
  "chain": [
    {
      "index": 1,
      "timestamp": "1606852242717",
      "nonce": 100,
      "hash": 0,
      "previousBlockHash": 0,
      "id": "id",
      "password": "password",
      "email": "email"
    }
  ],
  "currentNodeUrl": "http://localhost:3001",
  "networkNodes": []
}
    
```

Fig. 2. Blockchain JSON Data Structure

Fig. 2에서 제시된 것과 같이 블록체인은 블록을 저장하는 체인과 현재 노드의 URL, 연결된 네트워크 노드들의 배열로 이루어져 있다. 각 블록은 블록의 index 값, 블록의 생성 시간, nonce(Nonce) 값, 해시 값, 이전 해시 값, 그리고 아이디, 비밀번호, E-Mail주소로 이루어져 있다.

블록의 해시 값은 블록의 nonce 값과 해시 값, 이전 해시 값, 아이디, 비밀번호, E-Mail 주소를 하나의 문자열로 만들어 암호화한 것이다. 해시 함수에 대입하는 문자열에 이전 해시 값을 사용함으로써 어떤 블록의 데이터가 훼손되면 해당 블록 이후에 존재하는 모든 블록의 해시 값 또한 완전히 달라지게 되고, 잘못된 데이터임을 모든 노드에서 알아차릴 수 있게 된다.

체인의 networkNodes 배열에는 현재 노드와 연결된 다른 네트워크 노드들의 URL이 저장된다.

회원가입 페이지에서 입력된 정보를 하나의 블록으로 만들어 체인에 추가하고, 로그인 화면에서 아이디와 비밀번호를 입력 받아 해당하는 블록이 존재하는지 체인을 탐색한다.

3.2 작업 증명(Proof of Work)

채굴은 컴퓨터 처리 능력을 사용하여 수행 기록을 남기는 과정으로, 작업의 결과로 블록의 구성요소인 nonce 값이 구해지는데, nonce 값을 찾는 행위를 작업 증명이라고 한다. 작업 증명의 목적은 블록 생성을 계산적으로 어렵게 만들어 공격자들이 블록체인을 조작하는 것을 방지하는 것이다. 블록체인에서 사용하는 SHA-256은 전혀 예측 불가능한 유사난수 함수(Pseudorandom function)로 설계되었기 때문에 유효한 블록을 생성하기 위한 유일한 방법은 nonce 값을 증가시키며 생성되는 해시 값이 조건을 만족하는지 확인하는 과정을 반복하는 것이다[4].

블록체인에 새로운 블록이 생성될 때, 현재 블록의 데이터와 이전 블록의 해시 값이 작업 증명을 수행하는 함수로 전달된다. 이 함수에서는 nonce 값을 0에서부터 1씩 증가시키며 해시 함수에 대입한다. 보통 해시 값이 n개의 0으로 시작하는 문자열일 때 nonce 값을 구한 것으로 처리한다. Fig. 3에서 제시된 것과 같이, n이 클수록 nonce 값을 구하는 데 더 많은 시간이 소요되므로, n의 값으로 난이도를 조절한다.

```

4번째 블록의 채굴이 성공하였습니다. (난이도 : 4)
-----
블록 번호 : 4
이전 해시 : 00007c7ae822d28a11a74eacfb02806fca291ce5b9213c1abe38a1be011c269
작업 난수 : 13193
블록 데이터 : fourth_data
블록 해시 : 00003538e43ab427e7bec618915792083d7843273fa001ee9677478c38f62426
-----
생성 시간 : 0.520

4번째 블록의 채굴이 성공하였습니다. (난이도 : 5)
-----
블록 번호 : 4
이전 해시 : 000000512d001d08d151eac6443c0e45ec98a5cf38412f42022d6a07fe1f683
작업 난수 : 740319
블록 데이터 : fourth_data
블록 해시 : 00000f72c9c680621d938f6c9f163c4c43a2ab9288527d9708fc6e0d795c
-----
생성 시간 : 6.088

4번째 블록의 채굴이 성공하였습니다. (난이도 : 6)
-----
블록 번호 : 4
이전 해시 : 0000007d2ce8122c99cdeab8363d415e4887add6953cda26e6f6ac31624cf
작업 난수 : 9979524
블록 데이터 : fourth_data
블록 해시 : 00000057256081d0909a42c01e0c35f0c53c462cf336ee757ed1bd1007f155
-----
생성 시간 : 264.124
    
```

Fig. 3. Mining time required according to difficulty

올바른 해시 값을 생성하기 위해서는 해시 함수를 아주 많은 횟수만큼 실행하여야 하고, 이는 많은 에너지와 컴퓨팅 능력을 소모시킨다. 누군가 블록체인의 과거 블록, 또는 블록 안의 데이터를 변경하고자 한다면 많은 에너지를 사용하여야 하므로, 대부분의 경우 이미 존재하는 블록을 다시 채굴하거나 생성하는 일은 불가능하다.

3.3 탈중앙화

블록체인 네트워크를 구성할 API 서버의 여러 인스턴스(Instance)를 생성한다. 생성된 인스턴스들은 서로 연결되어 블록체인 네트워크 노드 배열에 저장되고, 배열에 존재하는 네트워크 노드들은 동일한 체인을 갖는다.

Fig. 4는 'register-and-broadcast-node' 엔드 포인트를 구축한 것이다. 한 노드에 새로운 네트워크 노드의 URL을 데이터로 하여 새로운 노드를 등록하고 브로드캐스팅 하려는 요청을 보내면, 요청을 받은 노드는 새로운 URL을 자기 자신의 노드에 등록하고 전체 네트워크에 브로드캐스트 한다.

```

app.post('/register-and-broadcast-node', function(req, res){
  const newNodeUrl = req.body.newNodeUrl;
  if(member.networkNodes.indexOf(newNodeUrl) == -1)
    member.networkNodes.push(newNodeUrl);
  member.networkNodes.forEach(networkNodeUrl => {
    const requestOptions = {
      uri : networkNodeUrl + '/register-node',
      method : 'POST',
      body : { newNodeUrl : newNodeUrl },
      json : true
    };
    regNodesPromises.push(rp(requestOptions));
  });
});

```

Fig. 4. '/registers-and-broadcasts-node' Endpoint

3.4 합의 알고리즘

합의 알고리즘은 하나의 노드를 네트워크 내의 다른 모든 노드들과 비교하여 이 노드 안에 올바른 데이터가 있음을 확인할 수 있도록 한다. 본 논문에서는 “가장 긴 체인 규칙”에 의한 합의 알고리즘을 사용한다. “가장 긴 체인 규칙”은 노드에 저장되어 있는 체인의 길이를 다른 노드들에 저장되어 있는 체인들의 길이와 비교한다. 만약 더 긴 길이를 갖고 있는 체인이 발견되면 현재 노드의 체인을 발견된 체인으로 교체한다. 이는 수정, 삭제가 불가능한 블록체인 구조에서 가장 긴 체인이 마지막으로 갱신된 체인이기 때문이다[5].

Fig. 5는 ‘consensus’ 엔드 포인트를 구축한 것이다. 요청이 발생되면 먼저 블록체인 네트워크의 다른 모든 노드들로부터 각각의 체인 복사본을 가져와 현재 노드와 비교한다. 그런 다음 현재 노드의 체인보다 더 긴 체인이 존재하는지 확인하고, 만약 더 긴 체인이 존재하며, 유효한 체인인 경우 현재의 체인을 더 긴 체인으로 교체한다.

```

if (!newLongestChain || (newLongestChain &&
  member.chainIsValid(newLongestChain))){
  res.json({
    note : 'Current chain has not been replaced.',
    chain : chain
  });
} else {
  chain = newLongestChain;
  pendingTransactions = newPendingTransactions;
  res.json({
    note : 'This chain has been replaced.',
    chain : chain
  });
}

```

Fig. 5. Consensus Algorithm(The Longest Chain Rule)

IV. Implementation

회원가입 화면에서는 아이디와 비밀번호, E-Mail 주소를 입력받고 각 입력 조건을 만족하는지 검사한다.

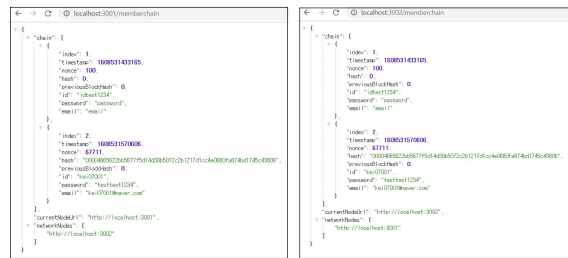
Fig. 6과 같이, 모든 입력 값이 각 항목에 대한 조건을 만족시킨 상태에서 회원가입 버튼을 클릭하면 회원 정보가 하나의 블록으로 체인에 등록된다.

Fig. 6. Register Page

Fig. 7의 (a)는 Fig 6에서 입력된 데이터를 하나의 블록으로 만들어 체인에 등록한 것이다. 네트워크 노드 배열에 등록된 노드들은 모두 같은 체인을 갖는다.

Fig. 7의 (b)에서 보듯이 Fig. 7의 (a)에서 등록된 블록은 연결된 네트워크 노드의 체인에도 동일하게 등록된다.

체인이 갱신된 시점 이후 노드가 추가되면, 새로운 노드는 최초의 블록인 체네시스 블록(Genesis Block)만을 가지고 있게 되는데, ‘consensus’ 엔드 포인트에 요청을 보냄으로써 기존의 블록들과 같은 체인을 가지게 된다.

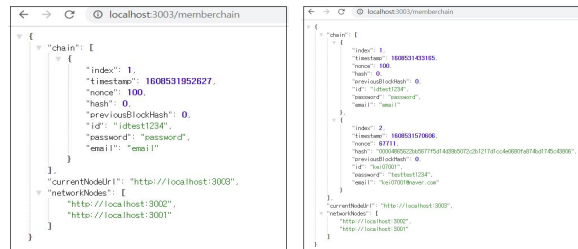


(a) http://localhost:3001 node' s chain

(b) http://localhost:3002 node' s chain

Fig. 7. Chain of Nodes in the API Server

Fig. 8의 (a)는 Fig 6 ~ Fig. 7의 작업 수행 이후 연결된 노드가 체네시스 블록만을 가지고 있는 상황이다. 이런 경우, ‘consensus’ 엔드 포인트에 요청을 보내면 Fig. 8의 (b)와 같이 새로 등록된 노드 또한 올바른 체인을 가지고 있는 것을 알 수 있다.



(a)

(b)

Fig. 8. http://localhost:3003 Node' s Chain

Fig. 9는 로그인 페이지로, 아이디와 비밀번호를 입력받아 로그인 성공 여부를 결정한다.

Fig. 9. Login Page

로그인에 성공한 경우 Fig. 10과 같이 로그인에 사용된 아이디와 로그인에 성공했음을 알리는 메시지를 출력하는 페이지로 이동한다.

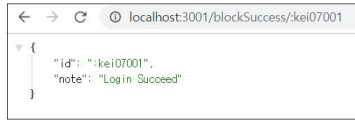


Fig. 10. Login Success Page

로그인에 실패하는 경우, 실패 원인을 판단하고 로그인 실패 페이지로 이동하여 로그인 시도에 사용된 아이디와 로그인 실패 사유를 출력해준다. Fig. 11의 (a), (b)는 각각 존재하지 않는 아이디가 입력된 경우, 아이디는 존재하지만 비밀번호로 입력된 값이 잘못된 경우 나타나는 페이지이다.

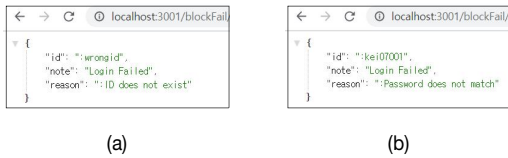


Fig. 11. Login Failure Pages

블록이 어떤 노드에서 생성되었는지와 관계없이, 합의 알고리즘이 수행된 모든 노드에서 같은 아이디와 비밀번호로 로그인할 수 있다. Fig. 12의 (a), (b)는 각각 Fig. 7의 (a)에서 생성된 블록의 아이디와 비밀번호를 가지고 연결된 네트워크 노드들에서 로그인을 시도했을 때 로그인 성공 페이지로 넘어가는 모습이다.



Fig. 12. Login Success Pages

V. Conclusions

본 논문에서 제시된 블록체인의 분산 데이터 저장 방식은 기존 데이터베이스의 중앙 집중 통제 방식에 의한 위험을 없애고, 보안성에 대한 강화 효과를 높일 수 있다. 제시된 알고리즘을 활용해 보다 트랜잭션 처리가 잦고 데이터 건수가 많은 다른 시스템에 대한 블록체인의 적용에 대해 향후 연구를 진행할 계획이다.

REFERENCES

- [1] “Blockchain vs Database: Understanding The Difference Between The Two”, <https://101blockchains.com/blockchain-vs-database-the-difference/>
- [2] Coin Revolution - Blockchain news, <https://coinrevolution.com/ko/what-is-the-difference-between-a-blockchain-and-a-database/>
- [3] What is SHA-256?, <https://m.post.naver.com/viewer/postView.nhn?volumeNo=15843055&memberNo=3270008>
- [4] James Ray, <https://github.com/ethereum/wiki/wiki/%5BKorean%5D-White-Paper#%EC%B1%84%EA%B5%B4>
- [5] Eric Traub, “Learn Blockchain Programming with JavaScript”