6. Introduction to Cryptography in Blockchain

Contents

1. 암호학 기초

암호학이란?

- **정의**
  - 데이터를 암호화하여 보안을 유지하고, 권한이 없는 사용자의 접근을 방지하는 기술

- **목적**
  - 기밀성 (Confidentiality)
  - 무결성 (Integrity)
  - 인증 (Authentication)
  - 부인 방지 (Non-repudiation)

- **블록체인에서 암호학의 역할**
  - 데이터 무결성 보장
  - 트랜잭션 서명과 검증
  - 네트워크 내 익명성 제공

2. 해시 함수

해시함수란?

- **해시 함수의 정의**
  - 입력 데이터를 고정된 크기의 해시 값으로 변환하는 함수
    - **비트코인의 SHA-256 :** 블록 헤더와 트랜잭션 데이터 무결성 보장
    - **이더리움의 Keccak-256 :** 스마트 컨트랙트와 트랜잭션 해싱

- **블록체인에서의 활용**
  - 트랜잭션 무결성 : 트랜잭션 데이터를 해싱하여 조작 방지
  - 블록 연결 : 블록 헤더의 해시 값으로 체인을 연결
  - Merkle Root 생성 : Merkle Tree 구조에서 트랜잭션 요약

2. 해시 함수

- **SHA-256 해시 계산**
  - 입력 : “Hello, Blockchain!”
  - 출력 : a591a6d40bf420404a011733cfb7b190d62c65bf0bcda32b58ba47b9d1731862

```
import hashlib

# 입력 데이터
data = "Hello, Blockchain!"

# SHA-256 해시 계산
hash_result = hashlib.sha256(data.encode()).hexdigest()

# 결과 출력
print(f"SHA-256 Hash: {hash_result}")
# SHA-256 Hash: a591a6d40bf420404a011733cfb7b190d62c65bf0bcda32b58ba47b9d1731862
```

- **해시함수 속성**

속성	설명
충돌 회피성	서로 다른 입력이 동일한 해시 값을 생성하지 않아야 함
역상 저항성	해시 값으로 원래 입력을 역산할 수 없어야 함
효율성	해시 계산이 빠르게 수행되어야 함

3. 디지털 서명

디지털 서명이란?

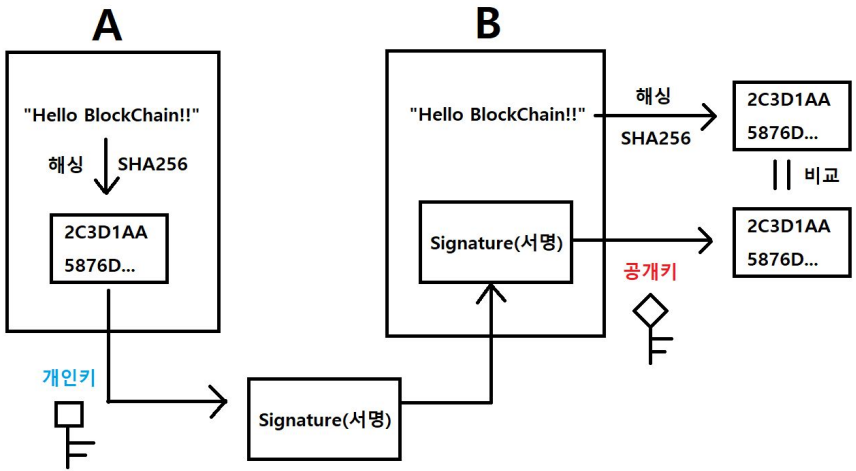
- **디지털 서명 정의**
  - 데이터를 암호화하여 발신자의 신원을 인증하고, 데이터가 변조되지 않았음을 보장

- **블록체인에서의 역할**
  - **트랜잭션 서명 :** 발신자의 트랜잭션 서명을 통해 권한 확인
  - **데이터 검증 :** 수신자가 서명을 검증하여 데이터 무결성 확인

3. 디지털 서명

디지털 서명 작동 원리

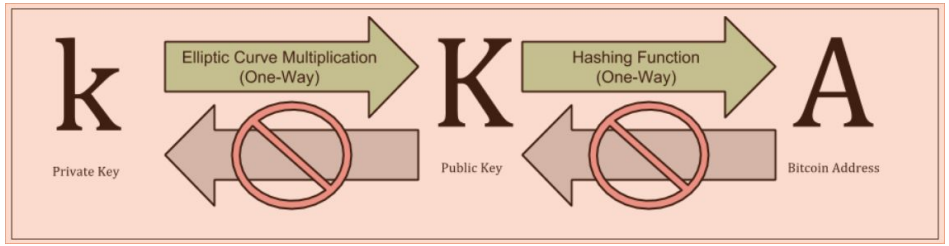
1. 발신자가 개인 키로 메시지를 서명
2. 수신자가 공개 키로 서명을 검증
3. 메시지가 변조되지 않았음을 보장



4. 공개 키 암호화

- **공개키 암호화 정의**
  - 개인 키와 공개 키를 사용하는 비대칭 암호화 방식
  - 발신자는 개인 키로 데이터를 암호화하고, 수신자는 공개 키로 복호화

- **블록체인에서의 활용**
  - 트랜잭션 서명
  - 키 쌍 관리 (공개 키 - 개인 키)



5. Merkle Tree

Merkle Tree란?

- Merkle Tree는 트리 형태의 데이터 구조로, 각 노드가 해시 값을 포함
- 트리의 가장 하위에 있는 데이터 블록(Leaf node의 해시 값을 계산한 뒤, 이를 반복적으로 조합해 최종적으로 **루트 해시(Merkle Root)** 생성

Merkle Tree의 이점

- **효율성 :** Merkle Proof를 통해 데이터 검증이 빠르고 경량화
- **확장성 :** 대규모 데이터도 효율적으로 검증 및 저장 가능
- **보안성 :** 데이터가 손상되거나 변조되었는지 빠르게 확인

5. Merkle Tree

Merkle Tree의 주요 역할

- **데이터 무결성 검증**
  - Merkle Root를 통해 전체 데이터의 무결성을 검증 가능
  - Merkle Proof를 사용하면 특정 데이터가 트리에 포함되어 있는지 효율적으로 증명 가능

- **데이터 경량화**
  - Merkle Proof를 통해 전체 데이터를 다운로드하지 않아도 특정 데이터의 유효성을 확인 가능
  - 경량 노드(Light Node) 또는 SPV(Simple Payment Verification)에서 활용 됨

사용 예

- **Bitcoin :** Merkle Tree 사용
- **Ethereum :** Merkle Patricia Tree 사용

5. Merkle Tree

비트코인에서의 **Merkle Tree** 사용

- 비트코인에서는 각 블록의 트랜잭션 목록을 Merkle Tree로 구성하여 Merkle Root 생성
- Merkle Root는 블록 헤더에 저장되어 블록의 트랜잭션 무결성을 보장
- SPV 클라이언트(경량 클라이언트)는 Merkle Proof를 활용해 전체 블록체인을 다운로드하지 않고도 특정 트랜잭션이 블록에 포함되어있음을 검증 가능

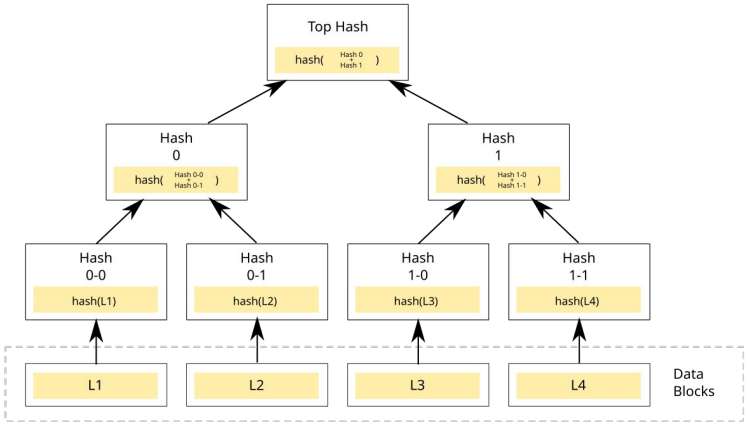
Merkle Tree 특징

- 데이터가 변경되면 루트 해시(Root Hash)가 변경되므로, 데이터의 무결성을 보장
- 루트 해시(Root Hash)를 사용하여 특정 데이터가 존재하는지 효율적으로 검증 가능

5. Merkle Tree

Merkle Tree 동작방법

- 데이터 삽입
  - 모든 데이터는 해싱(Hashing)을 통해 고정된 크기의 해시 값으로 변환 됨
  - **리프 노드(Leaf Node)**에 원본 데이터를 해싱한 결과를 저장
  - 리프 노드(Leaf Node)에서 시작해 상위 노드로 해시 값을 결합하여 트리를 구성
    - 두개의 자식 노드의 해시(Hash)를 결합하여 보무 노드의 해시를 생성
    - 이 과정을 반복하여 최종적으로 Merkle Root가 생성

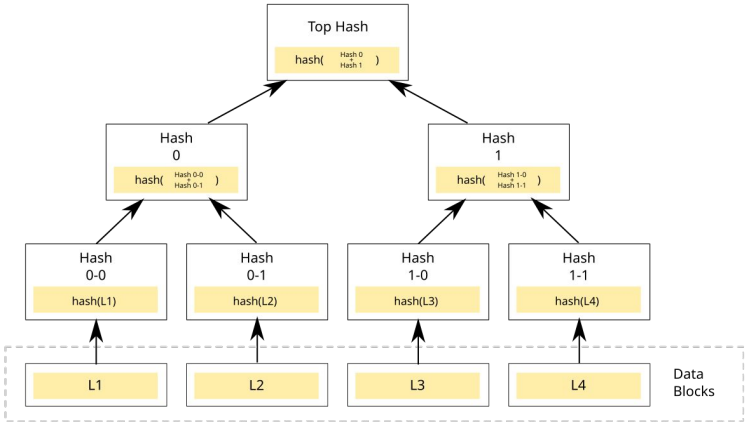


5. Merkle Tree

Merkle Tree 동작방법

- 데이터 검색
  - Merkle Tree는 검색보다는 데이터 검증에 중점을 둔 구조
  - 검색은 리프 노드에서 데이터를 직접 확인
  - 키(Key)가 없는 단순한 트리 구조이므로 특정 데이터의 위치는 미리 알고있어야 검색 가능

- 검색 예시
  - 특정 데이터 L2의 포함 여부를 확인하려면
    - 리프 노드에서 Hash(L2)를 찾음
    - 해당 노드가 존재하면, 데이터가 트리에 포함 됨

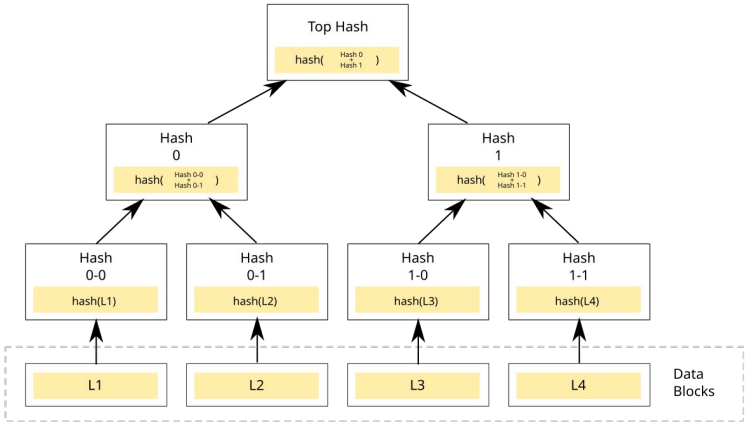


5. Merkle Tree

Merkle Tree 동작방법

- 데이터 검증
  - 특정 데이터가 트리에 포함되어 있는지 효율적으로 검증할 수 있음

- 검증 예시
  - 검증 대상이 L1라고 가정하면
    - Hash(L1)를 리프노드에서 가져옴
    - 형제 노드 Hash(L2)를 가져와 Hash(L1+L2)를 계산
    - 부모 노드의 형제 해시 Hash(L3+L4)를 가져와 Hash(L1+L2+L3+L4)를 계산
    - 최종 계산 결과가 Merkle Root와 일치하면 데이터가 트리에 포함되어 있음을 검증



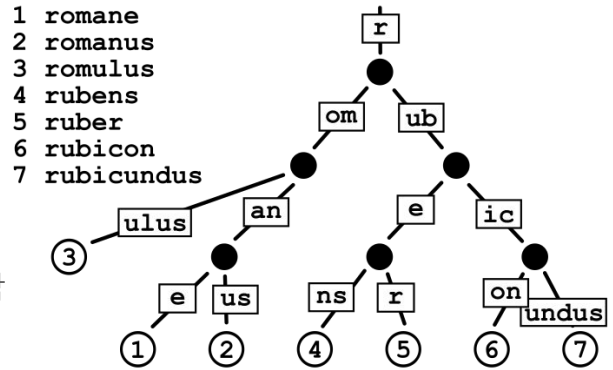
5. Merkle Tree

이더리움에서의 **Merkle Patricia Tree** 사용

- **계정 상태 저장 :** 각 계정 상태(잔액, 코드, 저장소 등)를 저장하고 검증
- **트랜잭션 기록 :** 블록의 트랜잭션 목록을 Merkle Patricia Tree에 저장
- **로그 저장 :** 이벤트 로그 데이터도 Merkle Patricia Tree로 관리

cf. Patricia Trie (a.k.a Radix Tree)

- Patricia Trie는 Trie 구조를 최적화한 버전
- 경로(Path)가 겹치는 노드들을 합쳐 공간 효율성을 높인 구조
  ex. 키 **abc** 와 **abd** 가 있으면, 공통경로 **ab**를 하나의 노드로 결합



5. Merkle Tree

Merkle Patricia Trie의 특징

- **데이터 무결성**
  - 해시 기반 검증
    - 모든 노드는 하위 노드들의 해시값으로 계산된 해시를 포함
    - 특정 노드가 변경되면 상위 노드와 루트해시(Root Hash)까지 모두 변경됨 → 데이터 변조시 즉시 감지

- **효율적 데이터 검색**
  - Patricia Trie의 경로 최적화를 통해 키를 기반으로 데이터를 빠르게 검색 가능

- **증명 가능성 (Proof of Inclusion)**
  - MPT는 특정 데이터가 저장되어 있는지 검증할 수 있는 Proof of Inclusion을 제공
    ex. 데이터가 포함된 경로를 따라가며 루트해시(Root Hash)와 비교하여 유효성을 검증

5. Merkle Tree

Merkle Patricia Trie의 동작 원리

- 데이터 삽입
  - 새로운 데이터를 삽입하면, 키를 기반으로 적절한 경로를 따라가며 Leaf Node에 데이터를 저장
  - 노드 추가 또는 변경이 발생하면, 영향을 받은 모든 상위 노드와 루트해시(Root Hash)가 업데이트

- 데이터 검색
  - 키를 기반으로 트리 경로를 따라가며 데이터를 검색
  - Patricia Trie의 최적화를 통해 키의 공통 경로를 빠르게 이동

- 데이터 검증
  - 특정 데이터가 트리에 포함되어 있는지 검증하려면
    - 루트해시(Root Hash)부터 시작해 경로를 따라 내려가면서 해시 값을 계산
    - 최종적으로 Leaf Node에서 데이터를 비교

6. 영지식 증명 (Zero-Knowledge Proof)

- **영지식 증명이란?**
  - 상대방에게 아무런 정보(민감한 정보)를 제공하지 않고도 자신이 특정 정보를 알고 있음을 증명

- **블록체인에서의 활용**
  - **프라이버시 보장**
    - 거래 금액, 거래 상대방 정보 등 정보를 노출하지 않고 거래 유효성 검증
      - ex. Zcash의 프라이버시 트랜잭션
  - **효율성 제공**
    - 대규모 데이터 검증 과정에서도 민감한 데이터를 전송하지 않아 네트워크 부하 감소
      - ex. zk-Rollup 기술

6. 영지식 증명 (Zero-Knowledge Proof)

영지식 구현 방식 예시

- **상황**
  - 친구1(검증자)은 두 공이 서로 다른 색(빨간색, 녹색)임을 확인하려 함
  - 친구2(증명자)는 공의 색을 구분할 수 있는 능력을 가지고 있지만, 이를 직접적으로 보여주지 않고 자신이 알고있음을 증명하려 함

- **방법**
  1. 친구1(검증자)는 두 공을 뒤로 숨기고 순서를 바꿀 수도 있고, 바꾸지 않을 수도 있음
  2. 공을 친구2(증명자)에게 보여주며 “순서가 바뀌었는지" 물어봄
  3. 색을 구분할 수 있는 친구2(증명자)는 공의 색 차이를 이용해 항상 정확히 대답
  4. 친구1(검증자)은 이 과정을 여러 번 반복하여 친구가 공의 색을 실제로 알고 있음을 확신

7. 블록체인 보안과 암호학

- **블록체인의 보안**
  - **51% 공격 방지 :** PoW, PoS 같은 합의 알고리즘으로 공격 방지
  - **Sybil 공격 방지 :** 노드 신뢰도 평가

- **암호학의 역할**
  - 데이터 암호화
  - 트랜잭션 무결성 보장

https://github.com/habit80/myBlockPanda

End