-<이더리움 #7>

1. 이더리움과 비트코인 차이점 -> #7 번 프린트 보기

	이더리움	비트코인
사용 목적	스마트 계약 블록체인	결제용 블록체인
화폐	Ether 사용	Bitcoin 사용
블록체인 모형	계정 기반 (외부 계정, 계약 계정)	UTXO 모형
채굴 알고리즘	현재는 Ethash Proof-of-Work (PoW)로 블록 생성 (향후 Proof-of-Stake (PoS)으로 변경 예정)	SHA-256-기반 Proof-of-Work
블록 생성 시간	3 ~ 30 sec	~ 10 min
블록 크기	30KB (2KB/s)	1MB (1.67KB/s)

2.

3.

A. 계정기반 vs 거래기반(UTXO) 작업방식

4. 이더리움 블록체인의 모든 기능은 항상 EOA 가 날린 트랜잭션으로부터 시작된다.

5. 먼저 "소유계정"은 오직 거래만을 담당하고, 거래에 대한 기록은 "계약계정"에 저장된다.

   A. 이때 계약계정은 소유 계정에서 요구하는 조건을 수행할 수 있다. 예를 들어 "이더의 가격이 100 만원이 되면 B 는 C 에게 이더를 보내라 " 라는 조건이 있으면, 먼저 소유계정에서 보낸 이더가 pow 를 통해 블록으로 생성되고, B 에게 이더를 보낸다. 그리고 조건이 만족되면 B 는 C 의 외부계정으로 이더를 보내게 된다.

6. 계정 구조

   A. "계약관련 계정(CA,조건으로 거래내용이 저장되는 곳)"과 "소유금액 계정(EOA,실제 돈 거래가 이루어지는 계정)"으로 나눠진다.

   B. 소유계정 (EOA) or 외부계정

      i. 개인키를 사용해 거래를 생성하고 서명함

      ii. Ca 를 활성화 하기 위해서 eth 가 필요

      iii. 이더리움 사용자를 위한 계정

      iv. 주소와 연결하여 잔액정보를 기록

      v. 개인키로 제어되는 것으로서, 코드를 저장할 수 없다.

      vi. EOA 에서의 value 는 ether 를 뜻함

   C. 계약 계정 (CA)

      i. 계약서들을 저장하는 계정으로, 자신 스스로는 활성화 될 수 없다.

      ii. 계약 증명이 있는 계정으로, 프로그램의 일종이다.

      iii. 코드정보 및 잔액 정보를 기록

      iv. 스마트 컨트랙트 코드에 의해 제어되며, 특정 CA 코드를 저장할 수 있다.

   D. EOA -> EOA : 금액이 왔다갔다 할 때 사용

   E. EOA -> CA : 계약관련으로 일을 수행

7. 상태 트리

   A. 계정 정보, 이더 잔액 정보를 저장하는 곳

B. 상태 트리는 블록 외부에 보관하고, 각 블록에는 상태 트리의 루트 노드 값만 저장한다.

C. Nonce : 이중거래를 방지(얼마의 거래가 발생한지 알 수 있는 것) -> 동일한 값이 나오면 동일지불인 것이다. (중요함)

D. 블록 외부에 보관하고, 블록에는 상태트리의 루트 노드값만 저장

8. 이더리움의 스마트 계약

A. CA 가 EOA 로부터 명령을 받으면 EVM 에 의해 자동으로 수행되는 것을 "스마트 계약 " 이라고 한다.

B. 자율 에이전트와 같다 (자동적으로 기능을 수행하는 것)

C. 스마트 컨트랙트를 활성화 하기 위해서는 EOA 가 필요하다.

D. EVM 에 의해 스마트 컨트랙트(자율 에이전트) 가 실행된다.

E. 스마트 계약의 목적

i. 데이터 저장 및 유지 (CA 의 일종으로, 계약 내용 저장)

F. Ca 가 EOA 로부터 명령받으면 EVM 이 실행되고, EVM 은 스마트 컨트랙트가 실행되고 경쟁적으로 블록을 생성한다.

9. EVM (이더리움 버츄얼 머신)

A. 연산을 수행하는 곳

B. 블록들의 검증 절차로 실행되는 것(마이닝이 EVM 을 실행)

C. 마이닝이 EVM 코드를 실행하고, 경쟁적으로 블록을 생성

D. 생성된 이더리움 노드들은 생성된 블록들의 검증 절차의 일부로써 EVM 을 실행

E. EVM 이 contract code 를 EVM Bytecode 로 바꿔줌

F. EVM 은 코드기 때문에 무한루프가 생길수 있는데, 이때 gas 라는 개념을 통해 문제를 해결한다.

G. Solidity 특징 -> 튜링 완전 : 무한 반복이 가능한 것 -> 이것을 해결하기 위해 gas 를 사용

10. 이더리움 화폐 단위

A. 10^9 wei = 1gwei

B. 1Eth = 10^{18} wei

C. Gas 수수료 설정

i. 최대값 = gas price * gas limit

ii. Ex) 10^6gwei = 10^6 * 10^9 wei = 10^{15} wei = 10^{-3} Eth

11. 수수료 gas

A. 코드 명령어는 gas 를 필요로 한다. 이때 gas 부족하면 상태 변경이 취소되고, 사용된 gas 도 돌려받지 못한다.

B. 수수료 gas 는 채굴자의 이더리움 주소로 들어간다.

12. 이더리움은 Ghost 프로토콜을 사용한다.

A. 블록 생성자 삼촌 블록(전파시간보다 블록생성 시간이 더 빠를 경우 발생)에게도 보상을 준다. (fork 일어난 노드에 대해서도 보상을 지급)

13. POS(지분증명)

A. Eth 라는 금액에 대해 기간과 양에 따라 블록 생성 권한을 주는 것

B. 많은 Eth 를 가질수록 더 많은 블록을 생성할 수 있다. -> 중앙화가 될 수 있다.

14.

15. 블록체인의 취약점
    A. 트랜잭션 가변성 : 거래내용은 동일한데 사인하는 부분만 공격자가 다르게 바꿔 계속 돈을 요구할 수 있도록 만드는 것
16. Asic: 채굴기중 가장 좋은 성능을 내는 것
17. Dictionary attacks(아무 단어나 무작위로 대입해서 개인키를 알아내는 것)
18. Pool hopping : hash파워가 쌘 마이너들이 여러 마이닝 풀을 돌아다니면서 더 많은 보상을 얻는 행위
19. 마이닝 풀 공격
    A. Sabotage: mining pool을 망가트리기 위해 pool로 들어가 nonce값을 찾아도 보내지 않아 자원을 더 소모하게 하는 공격법
    B. Lie-in-wait : 자원을 더 요구해서 소비를 많이 한 것처럼 보이게 해서 더 많은 리워드를 받는 것
20. Blacklisting : 마이닝 풀 안에서 특정 지갑의 거래를 블록화 시키지 않을 수 있음 -> 그 지갑은 거래 못함
21. Selfish mining : 비트코인은 더 긴 블록을 선택하기 때문에, 블록을 하나 만들더라도 알리지 않고 빠르게 다음 블록을 생성해 더 많은 이득을 챙기는 방법
    A. 방어전략
        i. Block validation : 서명을 통해 뒤에 숨겨놓은 블록이 없다는 것을 증명하게 함
        ii. Fork-punishment : 분기에 대한 리워드는 없고, 그 뒤에 만든 block에 몰아서 보상을 주는 방법
        iii. Unforgettable timestamps : 위조가 불가능한 시간을 이용해서 이 사간 정보가 블록 생성에 사용되서 같은 경쟁을 하게 하는 것 -> 신뢰하는 제 3자가 있어야함

<#9>

1. BIP: 비트코인 기능개선을 위한 제안을 담은 문서
    A. 비트코인이 업데이트 될 때마다 BIP에 내용이 추가된다.
    B. 개선을 위해서는 소프트포크 수행을 통해 진행
2. 하드포크 vs 소프트 포크
    A. 하드포크 : 이전과 호환이 안되기 때문에 모두가 바꿔야 함
        i. 비공식적 : 예측할 수 없는 문제 발생시 사용
            1. 신규 기능 추가
            2. 블록크기 확장
            3. 해커 공격
            4. 새로운 코인 만들 경우
        ii. 공식적 : 예측 가능한 문제 발생시 사용
    B. 소프트포크 : 이전과 호환이 되기 때문에 천천히 바뀌어도 된다.
    C. 업데이트 block을 추가시키려면 50%이상의 hash 파워가 필요하다. 이때 50%를 넘기면 하드포크가 가능하고, 50%를 넘기지 못하면 하드 포크에 실패한다.
    D. 소프트포크는 파워와 상관없이 채택되기 위해서는 새로운 규칙에 동의하는 많은 채굴자들이 필요하다.
3. 합의 알고리즘 종류
    A. 종류 : BFT, FBA, Pow, Pos, PoET
    B. **BFT :** 동기식으로, 결함이 있어도 전체 1/3 만 넘지 않으면 정상동작 한다.
        i. PBFT : 비동기식으로, 5가지 절차로 구성됨 -> 5가지 절차에 따라 서로 합의를 진행한다.
            1. Request : client가 leader노드에게 요청을 보냄
            2. Pre-Prepare : leader 노드가 다른 노드에게 요청을 보냄
            3. Prepare: 각각의 노드가 요청을 수락하고, 자신을 제외한 다른 노드로 요청을 보냄
            4. Commit : 받은 요청중 가장많이 받은 요청을 다른 노드로 전파
            5. Reply : 해당 요청에 대해 수락한 것이 2/3가 넘으면 합의를 이룬것으로 결정한다.
    C. **FBA :** BFT를 보완한 것으로, 공격성공하기 위해서는 쿼럼 슬라이스에 공격을 위한 노드가 있어야 한다.
        i. 쿼럼슬라이스 : 어떤 결정을 하든 같은 결정을 내리는 소규모 집합(특정 결과를 합의하기 위한 그룹)
        ii. 쿼럼 : 합집합으로, 특정 노드가 집합의 합의 결과에 동의하도록 설득하는 노드 집합
        iii. 쿼럼은 쿼럼 슬라이스가 서로 의견을 주고받을 수 있어야한다. -> 거래에 대한 모든 슬라이스가 동의해햐 하기 때문에
        iv. 쿼럼 intersection : 여러 쿼럼 슬라이스의 공통된 부분 -> 쿼럼 슬라이스끼리 묶었을 때, 교집합을 "쿼럼 인터섹션" 이라고 한다. -> fork를 방지하기 위해 사용했다.
    D. Pow : 작업증명
        i. 블록을 생성하는 것

ii. 먼저 문제를 풀어서 블록을 만들어야 그 내용이 전파되고 확정된다.

iii. 문제를 풀어 특정 타겟보다 작은 hash값을 구해야 한다.

iv. 비트코인은 긴 블록을 선택하는 longest이다.

v. 이더리움은 계정기반 작업방식, 비트코인은 거래기반 작업방식(UTXO모델)이다.

vi. UTXO모델 : 받은 자산을 합치지 않고 쪼개서 저장 후 송금할 경우 새로운 UTXO를 생성해 전달한다.

vii.

	이더리움	비트코인
사용 목적	스마트 계약 블록체인	결제용 블록체인
화폐	Ether 사용	Bitcoin 사용
블록체인 모형	계정 기반 (외부 계정, 계약 계정)	UTXO 모형
채굴 알고리즘	현재는 Ethash Proof-of-Work (PoW)로 블록 생성 (향후 Proof-of-Stake (PoS)으로 변경 예정)	SHA-256-기반 Proof-of-Work
블록 생성 시간	3 ~ 30 sec	~ 10 min
블록 크기	30KB (2KB/s)	1MB (1.67KB/s)

E. POS(지분증명)

i. Eth 라는 금액에 대해 기간과 양에 따라 블록 생성 권한을 주는 것

ii. 많은 Eth 를 가질수록 더 많은 블록을 생성할 수 있다. -> 중앙화가 될 수 있다.

F. **PoET:** TEE로 보호되는 랜덤 리더 선출 모델 -> TEE를 이용해 랜덤시간을 생성하고, 그 중 가장 짧은 시간이 나온 사람에게 블록 생성권한을 준다.

<#10>

1. ICO : 블록체인 스타트업을 위한 자금을 모으는 수단 -> 기존의 크라우드 펀딩을 탈중앙화 방식으로 적용하여 쉽고 빠르게 자금을 모을 수 있다.
    A. 토큰과 코인을 구분한다. -> 토큰은 코인과 달리 어디에 종속되어 있어야 한다. 코인은 플랫폼을 가지고 있다 .
    B. 스타트업 개발자는 ICO 를 통해 자신의 산업과 관련된 토큰을 투자자에게 주고, 투자자는 가상화폐로 스타트업에 투자할 수 있게함
    C. 정부의 개입없이 자금을 조달하는 방법
    D. 백서 검증 절차 없이 자금 조달 가능
2. ICO vs IPO
    A. ICO : 규제가 거의 없지만 새로 생긴 기업 이여서 토큰의 가치를 신뢰할 수 없다.
    B. IPO : 규제가 심해서 사기 확률을 줄일 수 있다.
3. 토큰 : 잠재적 가치를 가지는 특수용도의 동전 같은 물건
    A. 물리적 토큰 : 일반적으로 하나의 기능만 갖고, 특정 조직으로 제한됨
    B. 블록체인 토큰 : 전 세계 시장에서 유통 가능
    C. 속성
        i. 대체성 : 값이나 기능의 차이 없이 다른 토큰으로 대체할 수 있는 경우
        ii. 거래상대방 위험 : 나와 거래자 사이에 스마트 컨트렉트를 이용해 간접적으로 거래가 된다.
        iii. 내재성 : 일부 토큰은 블록체인에 내재적인 디지털 아이템을 나타냄
    D. 토큰 이코노미
        i. 토큰은 한정된 곳에만 의미가 있어 미래 교환 가치가 확실치 않다. -> 블록체인을 통해 해결
            1. 중앙 주체가 필요없어 어디서든 확장과 범용성을 확보하여 사용할 수 있게된다.
            2. 코인과 달리 한정된 곳에서만 의미가 있다. -> 종속적이다.
4. IEO
    A. ICO 와 다르게 중간에 거래소가 있어서 해당 백서와 스타트업 팀을 보증하는 역할을 함

<#11>

1. 퍼블릭 블록체인은 모든 사용자에게 코드가 오픈되기 때문에 해커들에게 악용되기 쉽다.
2. 스마트 컨트랙트
   A. 불변성 : 배포된 코드는 변경할 수 없음. 따라서 신중히 배포해야 함
3. 스마트 컨트랙트 보안
   A. 오버플로우 : 최대 비트를 넘기면 0 이됨
   B. 언더플로우 : 0 에서 1 빼면 가장 높은 비트가 됨
   C. 외부 사용자에게 ether 를 송금하기 위해서는 smart contract 가 외부 계정에 대한 호출을 요청해야 한다.
      i. 외부 호출을 공격자가 악용할 수 있다.
      ii. 공격자가 smart cotract 의 callback 을 포함하여 대체 코드를 실행하도록 강제할 수 있다.
4. DAO -> 조직이름
   A. 이더리움 블록체인 환경 위에서의 Smart contract 코드를 이용한 조직 및 단체를 구성
   B. DAO 토큰을 발행 판매하여 자금으로 마련함 . 이때, 해커가 DAO smart contract 에서 ether 를 인출함
   C. Reentrancy(재진입) attack : 스마트 컨트랙트가 알 수 없는 주소로 ether 를 전송할 때 발생
   D. 이더 전송 방법
      i. Address.transfer() -> 가스가 소모됨
      ii. Address.send() -> 가스가 소모됨
      iii. Addres.call.value().gas().(함수이름) -> 가스가 필요하지 않아 공격에 사용됨
5. 폴백함수 : 이름이 없는 함수로 컨트랙트 실행시 무조건 한번 실행됨
   A. Require(msg.sender.call.value()()); 이 부분이 문제가 됨 -> 이것 때문에 컨트랙트가 해당 부분에서 멈추고 폴백함수가 계속 호출됨
6. 해결책
   A. Transfer() 함수 이용
   B. 이더를 보내기전 상태변수를 변경하는 로직을 실행
   C. 뮤텍스를 도입해 잠재적 재진입을 막는다.
      i. 하나의 메모리 공간을 누군가 사용하고 있다면 다른 사용자가 해당 메모리를 사용하지 못하도록 만드는 것
7. Parity : 해커가 한 번의 거래로 피해자의 지갑을 탈취할 수 있는 단순 공격
8. Solidity 는 디폴트로 public 이 지정된다.
   A. 가시성 종류
      i. External : 내용 공개를 의미하며, 계약서 외부에서 사용하는 인터페이스
      ii. Public : 내용 공개를 의미하며, 계약서의 외부, 내부에서 사용하는 인터페이스
      iii. Internal : 내용 비공개를 의미하며, 부모에서 파생된 컨트랙트에서만 볼 수 있음
      iv. Private : 내용 비공개를 의미하며, 선언한 컨트랙트 에서만 접근 가능
   B. 지갑같은경우는 private 나 internal 로 사용하는 것이 좋다.

<#12>

1. 스마트 컨트랙트 = 클래스
   A. 객체지향 프로그램에서 클래스와 유사한 독립적인 코드
   B. 탈중앙화 문제들을 풀기 위한 컨트랙트 규칙들을 구현함
   C. 이더리움 플랫폼 사용을 위해 솔리디티를 이용해서 스마트 컨트랙트를 구현한다.
   D. 스마트 컨트랙트는 비트코인 블록체인 프로토콜이 제공했던 기본적인 신뢰를 확장시키는 코드
   E. 따라서 스마트 컨트랙트는 블록체인의 프로그래밍을 가능하게 해준다.
2. Solidity
   A. 정적 타입의 언어
   B. 변수 타입은 컴파일시 검사 -> 컴파일 할 때 변수 타입 정해진다.
   C. Keccak256() : hash 값으로 변환해주는 함수
   D. 솔리디티의 함수는 컨트랙트와 연관되어 있는 코드 모듈이다.
      i. 함수들은 Function hash 로 서명된다.
   E. View : 어떤값을 변경하거나 무엇인가를 쓰지 않을 때 사용
      i. Function a() public view returns(string){}
   F. 폴백 함수 : 이름이 없고 인자와 리턴 데이터가 없는 함수
   G. 로컬변수는 storage 에 저장, 매개변수는 메모리에 저장된다.
   H. 상속 : is 키워드 이용해서 상속 가능 -> contract a is b -> b 를 상속받음

<#13, #14>

1. Geth
   A. Go ethereum 은 프로그래밍 언어인 go 에서 구현된 전체 이더넷 노드를 실행하기 위한 프로그램
   B. Geth 를 이용해 스마트 컨트랙트 실행
2. Ganache
   A. 블록이 자동 생성되는 것으로 "개인용" Dapp 개발에 사용된다.
   B. 트랜잭션과 블록을 mining process 없이 생성할 수 있음
3. Metamask
   A. 크롬 기반의 월렛
4. M2M (machine -to – Machine)
   A. iot 시대에 블록체인 기술이 활용되는 가장 큰 3 가지 요소
      i. 사물간 지급 결제(M2M payments)
      ii. 사물 보안(Security of Things)
      iii. 자동화 과정(Automated Process)
5. Swift 와 Ripple
   A. 국제간 송금할 때 이용하는 시스템
   B. Swift : 옛날에 이용했던 방법으로 절차가 복잡한 은행시스템
   C. Ripple : 최근 개발된 것으로, 여러 은행에서 실시간 자금을 송금한다. 저렴한 수수료와 빠른 송금을 지원한다.
6. 블록체인 이용 회사들
   A. Goldman sachs : 투자회사로, 비트코인 곤련 블록체인 ETF 추진
   B. Walmart : 농수산 공급에 블록체인을 적용시켜 추적할 수 있게 함
   C. Alibaba : 상품 위조사건 줄이기 위해 블록체인 적용
   D. Maersk : 최대 해운회사로, 하이퍼레저를 사용
   E. 크립토키티 : 이더리움 기반의 게임
7. DID
   A. 탈중앙화 신원증명으로, 개인정보를 개인 단말기에 저장해 개인정보 인증시 필요한 정보만 제출
   B. QR 인증 같은거
8. Deanonymization (탈 익명성)
   A. 블록체인은 기본적으로 익명성을 보장하지 않는다. = 탈 익명성
   B. 블록체인에서 사용자 익명성을 보호하기 위해 사용된 정교화한 암호화 및 개인 정보 보호기능이 필요하다.
   C. 따라서 개인정보를 보호하기 위해 대부분 블록체인은 가명정보(비트코인 계좌번호) 를 사용한다.
   D. 익명화의 목표는 실제 세계에서의 ID 를 가명정보와 연결하는 것이다. -> 익명화가 완벽하면 범죄에 사용됐을 경우 찾을 수 없기 때문에 어느정도 익명성을 벗겨야 한다.
   E.

F. 익명성을 벗기는 방법

i. Transaction Gragh Analysis : 그래프의 형태로 가명정보와 연결

ii. Clustering : 동일한 개체들에 대한 주소 클러스터

iii. Taint : 오염도라고 하며, 여러 개의 주소에서 받은 금액 중 한 개의 주소로부터 받은 금액에 대한 퍼센테이지를 나타냄

1. 시간이 많이걸리지만 좋은 정보를 얻을 수 있다.

G. 개인정보 보호를 위해 사용하는 이런 기능들은 범죄 요소를 수반하기 때문에 조심해야 한다.

9. Mixing

A. 추적하기 어렵게 만들어서 익명성을 최대로 높이는 것

B. 1BTC 를 입금했을때 Mixing 서비스 이용한다면, 다른 계좌에서 해당 BTC 를대신 입금하게 하는 방법 = 이런 mixing 된 코인을 "AltCoin" 이라고 함

C. 1 번의 mixing 을 통해 n 개의 peer 들을 구별할 수 없는 경우, anonymity set 의 크기는 n 이 된다. 그 다음은 n^2이 된다.

D. Anonymity set 이 커질수록 익명성이 크게 증가한다.

E. 하지만 Mixing 서비스가 믿을만 해야한다. 중앙화여서 거래장부를 만들수도 있음

F. 여러 개층의 AltCoin 으로 교환하여 자금 추적 어렵게 만듬

10. DASH

A. 네트워크에서 마스터노드(Mixing 하는 곳)가 privatesend(송금을 익명으로 하고 싶은 사람)로 익명 요청을 받게 될 경우, 3 개 이상의 전송 요청을 묶은 후 섞어서 보내는 방법으로, 추적을 어렵게 만듬

B. Privatesend 기능을 사용하여 거래가 익명 및 비공개인지 여부를 사용자가 선택할 수 있는 암호화폐

<#15> ERC-20 토큰

1. 이더리움 개선안 (EIP)
    A. 이더리움 품질 개선 문서로, 이더리움이 많이 나오기 때문에 표준으로 맞출 필요가 있다.
    B. 이때 표준으로 맞추기 위해 사용하는 문서가 EIP 이다.
    C. 종류
        i. Standard Track EIP : 구현에 영향을 미칠 수 있는 변경에 대한 제안
        ii. Informational EIP : 정보 제공을 위한 가이드 및 지침
        iii. Meta EIP : 프로세스 설명 또는 변경을 제안
2. ERC-20
    A. 이더리움 블록체인 네트워크에서 발행되는 토큰의 표준
    B. 토큰을 사용하기 위해 제공되는 표준 API 규격
    C. 스마트 컨트랙트내에 토큰에 대한 표준 API 를 EIP 라고 한다.
    D. 토큰을 전송하는 기본 틀(기능)을 제공하고, 토큰승인 및 on-chain-third party 에서 사용할 수 있도록 허용한다.
    E. 토큰은 컨트랙트 안에서 모든게 이루어 지기 때문에 저장소도 컨트랙트 안에 있어야 한다.
    F. 토큰에서 사용되는 주소는 실제 이더리움 주소가 아닌 ERC-20 안에서 사용되는 컨트랙트 주소이다.
3. ERC-20 의 함수 종류
    A. transferFrom : 토큰 전송시 approve 함수를 통해 전송할 수 있는 권리를 받은 사용자만 실행할 수 있어 투자자에게 토큰을 전송함
    B. approve : 투자자에게 이체를 실행할 수 있는 권한을 부여하는 것으로, 투자금액만큼 투자자에게 토큰권한을 부여하면 중간에서 ERC-20 이 투자자에게 토큰을 전송함
    C. balances : 전송시 한 개에서는 차감, 다른쪽은 추가되게 하는 함수로, 토큰 컨트랙트에서 토큰 소유자를 추적할 수 있게 된다.
    D. Allowances : 권한을 위임하는 함수
4. ERC-20 장점, 단점
    A. 장점
        i. 대체 가능성 : 서로의 토큰을 공유할 수 있다.
        ii. 유연성 : 많은 사람들이 자신만의 토큰을 쉽게 만들 수 있다.
    B. 단점
        i. 확장성 : 확장이 쉽지 않다.
        ii. 결제 수단 : 스마트 컨트랙트는 이더로만 결제할 수 있는데 ERC-20 에서 자신만이 사용하는 토큰을 주면 쓰레기 값으로 판단하여 토큰을 잃어버리는 경우가 생긴다.
5. ERC -721
    A. ERC-20 의 업그레이드 버전
    B. 우리의 실제 자산을 이용해 거래하는 방법
        i. Ex -> 물리적 자산 : 주택, 예술품

<# 16>

1. **NFT**
   A. 한 개의 토큰을 다른 토큰으로 대체하는 것이 불가능한 암호화폐
   B. 여기서 토큰은 디지털자산으로 인정받은 원본파일의 그림 또는 예술품 이다.
   C. 조작이 불가능하도록 디지털 자산의 소유권을 증명 보호 하는 기술
   D. NFT 로 기록하면 디지털파일의 원본에 대한 소유권을 증명할 수 있음
   E. 파일자체는 보관하지 않고 블록체인 증서만 보관된다. 원본파일은 외부저장소 IPFS 의 형태로 분산되어 저장된다.
   F. 원본과 복제본을 구분할 수 있는 중요한 요소 이다.
   G. 탈중화된 블록체인상에 저장되므로, 위조가 어렵고 추적하기 쉽다. 또한 소유권 분실에 대한 우려가 적다.
   H. 토큰을 분할하여 소유권을 부분적으로 유통 가능하도록 한다.
2. ERC 721
   A. 기본적으로 함수의 interface 를 제공하면, 회사들은 규격을 받아와서 자신들의 입맛에 맞게 만들어 나간다.
   B. 함수 balanceOf_= owner 에 대한 주소를 받고, 보유하고 있는 NFT 토큰들의 개수를 나타낸다.
   C. 함수 ownerOf = 소유권이 누구한태 있는지를 반환
   D. 함수 approve = 전송을 할 수 있도록 권한을 부여해주는 함수 -> 소유권자가 operator 에게 NFT 의 전송할 수 있는 권한을 부여, operator 가 NFT 토큰을 산 사람에게 권한을 넘겨줌
3. 팔때 token ID 와 smartcontarct Address 가 기록된다. 이 기록들은 IPFS 의 형태로 분산되어 저장된다.
4. **DeFi : 탈중앙화 분산 금융**
   A. 암호화폐를 담보로 걸고 일정 금액을 대출 받거나, 금액을 담보로 암호화폐를 대출 받는 방식
   B. 블록체인 네트워크상에서 스마트 컨트랙트를 활용하여 동작하는 탈중앙화 금융 서비스
   C. 기존금융은 특정 사용자만 거래기록을 볼 수 있는 반면, DeFi 는 모든 사용자가 거래 기록을 공유할 수 있다. -> 투명성이 중요
   D. 기술 필수 요소
      i. 탈중앙화 메인넷
      ii. 스마트 컨트랙트 코드 : 코드가 공개되어있어 로직을 확인할 수 있다. 임의의 행위로 인해 작동 중단이 불가능함(검열저항성), 국가 규제기관의 허락 없이 글로벌 금융서비스 제공이 가능함
      iii. 지갑
      iv. 거버넌스 : 오류발생이나 블록체인 상 정책이 바뀌었을 경우, 코드를 업데이트 할 수 있는 특수 권한을 작동시킬 수 있는 장치
      v. 탈중앙화 거래소

구분	기존 금융	De-Fi
허가	특정 고객	**네트워크상 존재하는 모든 고객**
운영 주체	중앙화	**탈중앙화**
중개인	신뢰 기관 필요	**네트워크 참여자가 대체**
투명성	특정 사용자만 접근 가능	**모든 사용자가 거래 기록을 공유**
검열 방지	검열 기관에 의해 특정 거래 삭제 가능	**하나의 주체가 특정 거래 기록 무효화 불가능**
프로그래밍 가능	독점 소프트웨어로 한정된 프로그래밍	**오픈 소스를 통한 자유로운 프로그래밍**

E. 기존 금융과의 차이점 중 투명성과 프로그래밍 가능 여부를 보기

F. CeFi : 기존금융과 디파이 사이의 것

G.

구분	디파이	씨파이	전통금융
이용 화폐	가상자산	가상자산, 법정화폐	법정화폐
중개자 및 관리주체	블록체인 네트워크 (거버넌스 토큰)	가상자산 거래소	전통금융기관
이용 화폐의 보관	지갑	가상자산 거래소	전통금융기관
금융서비스	예금, 대출, 가상자산 거래, 파생상품, 자산관리, 결제, 보험 등	가상자산 거래, 대출	예금, 대출, 증권거래, 환전, 파생상품, 자산관리, 결제, 보험 등
익명성	익명 거래	실명 거래	실명 거래
투명성	투명	불투명	불투명
국가 간 경계	없음	있음	있음
기업 사례	메이커, 에이브, 유니스왑, 컴파운드, 신세틱스 등	바이낸스, 코인베이스, 빗썸, 업비트 등	JP모건, 골드만삭스, 뱅크오브아메리카, HSBC 등

5. 이더리움 오라클

A. Oracle : 이더리움 외부에서 일어나는 문제들에 대한 정보를 알려주는 시스템

B. 필요성 : EVM 및 스마트 컨트랙트와 같이 동작하는 임의성을 위한 고유한 소스가 없음, 외부 데이터가 트랜잭션의 페이로드로만 유입 됨

C. 스마트 컨트랙트가 실행될 때마다 내부에서 랜덤값을 정한다면 다른 결과 상태가 나올 수 있기 때문에 탈중앙화된 합의가 불가능하게 될 수 있음 -> 이떄 외부데이터와 동기화될 수 있도록 오라클이 외부의 특정데이터 값을 알려준다.

D. 오라클은 데이터의 무결성(데이터가 틀렸을 수도 있기때문에)을 입증할 수 있는 오프체인 방식의 검증 방법이 필요하다.

i. 진위성 증명 : 스마트 컨트랙트는 체인상 진위성 증명을 검증함으로써 데이터를 처리하기 전에 데이터의 무결성을 검증

ii. TEE : 신뢰할 수 있는 실행환경으로 입증