

< 암호 알고리즘 >

2018. 11. 09(금)

1. 암호의 종류

- 대칭키 알고리즘(암호화 키 = 복호화 키) : AES, DES, SEED 등
- 비대칭키 알고리즘(암호화 키 != 복호화 키) : RSA, ECC 등
: 개인키와 공용키를 이용해서 묶고, 푸는 것을 반대로 실시
(개인키로 암호화 하면 공용키로 복호화 / 개인키로 복호화 하면 공용키로 암호화)
- 해시 알고리즘(메세지를 일정 길이로 축약해주는 알고리즘) : SHA-1/2/3, MD5 등
: SHA나 MD5는 결국 압축해주는 방법(Digest)
: 중복되지 않은 값이 나옴

2. 비트코인코어 설치~기동까지

- ubuntu에서 git을 설치하고 bitcoin관련 git을 복사해 시작
- 관련 라이브러리 설치 및 업데이트
- 빌드(./autogen.sh) : GNU시스템에서 사용하는 빌드도구인 autotool을 이용해 configure(구성)
운영체제의 하나이자 SW의 모음집으로 GNU's Not Unix의 약자
전까지의 과정을 모두 묶어 한 번에 실행(script로 묶어서)

설치 완료되면 아래 명령어 실행해보기!!(백단에서 서버가 시작됨!)

최초 구동 시 버전정보로 인해 맨 뒤에 -deprecatedrpc = generate라는 옵션을 써줘야 함.

ps(Process Status 현재 구동중인 프로세스 보여줌)

cli(Command Line Interface) = 검정화면

종료? bitcoin-cli -regtest stop

```
guru@rhcsa:~/src/bitcoin$ bitcoind -regtest -daemon -deprecatedrpc=generate
Bitcoin server starting
guru@rhcsa:~/src/bitcoin$ ps
  PID TTY          TIME CMD
  1572 pts/0    00:00:00 bash
 20200 pts/0    00:00:00 ps
guru@rhcsa:~/src/bitcoin$ bitcoin-cli -regtest getbalance
0.00000000
guru@rhcsa:~/src/bitcoin$
```

< 이더리움 네트워크 기초 >

2018. 11. 12[월]

1. 복습

- 비트코인의 난이도는 bit와 문제를 푸는 nounce로 구함(0의 개수가 몇 개냐에 따라서 달라짐!)
- 이더리움은 비트코인처럼 코인의 개수가 한정적이지 않음.(연간 1800만개 만 한정)
- 15초에 1개 생성되는 난이도를 가짐(약 4단계 중 현재 2단계 진행중)

2. 블록체인 책 추천

- 블록체인혁명(검정페이지) / 블록체인 무엇인가?(빨간책) => 추천★ / 블록체인 현상(흰/보) 만화로 배우는 블록체인(너무 어려우면 만화책 읽어보기)
- 블록체인 에스토니아처럼.....은 잘 모르겠음!

3. 공개키 vs 개인키

- 비대칭키의 경우 공개키로 암호화 했으면, 개인키로 복호화!
개인키로 암호화 했으면, 공개키로 복호화!!(개인키는 file같은 것으로 암호화된 어떤 데이터를 가지고 해쉬 알고리즘을 돌려 얻은 값을 개인키를 통해 확인하고 얻은 값을 비교)

4. 비대칭키 기반 서명 알고리즘(ECDSA)

- Transactions의 서명을 검증하기 위해서는 공개키가 있어야함(개인키로 암호화 했단 뜻)
- 지갑이 하나 생성되면 **기본키 → 공개키 → address가 생성!!(주소체계)**

5. 이더리움 기초

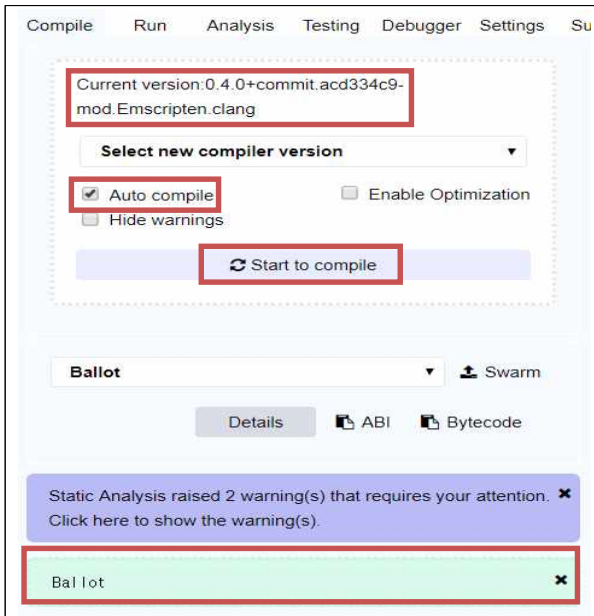
- 비트코인은 프로그램의 성질을 띄지 않음(PUT, GET 위주 / DDOS 공격 방지를 위해 반복문 없음)
- 이더리움은 이런 것을 해결하고자 함!!(이를 위해 잉클블록, Gas 등의 개념 추가)
- 이더리움(Network), 이더(이더리움에서 쓰이는 화폐의 단위)
ETH(이더), wei(위 : 모든 트랜잭션 및 네트워크 상에서 사용)라는 단위를 사용
- 책에 있는대로(p145) 설치하면 에러발생(→최신 버전으로 다운로드하고 make geth)
- 이더리움은 트랜잭션 없이도 수십초 간격으로 블록생성이 가능하며 보상이 ether 받을 수 있음!
-

6. 스마트 컨트랙트(myWallet에 가서 지갑 하나 만들어보기) : 개인키 보호 철저!!

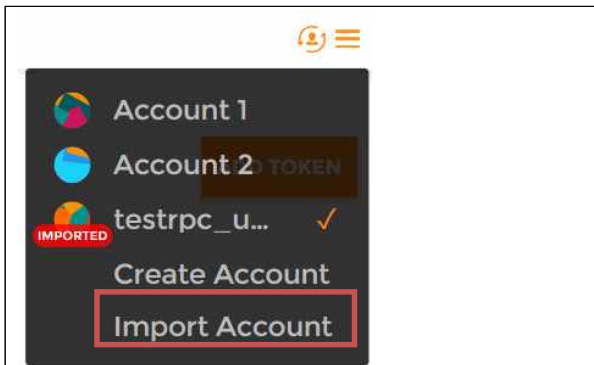
- 이더리움상에서 동작하는 프로그램 = 스마트컨트랙트 = DApp = 블록체인 어플리케이션
- solidity라는 언어로 스마트컨트랙트 개발
JS와 유사하며 튜링 완전 언어(모든 어플 관련 작업을 스마트컨트랙트 상에서 가능하게 함 : 변수,상수,반복문 등)
- 블록에 쌓이는 데이터(계정데이터, Tx데이터, 스마트컨트랙트 관련 데이터)
ex) 비트코인 실습 간 했던 가장 긴 노드만 살아남는 그런 것들 등
- 계정과 지갑은 약간 다른 개념(헷갈ㄴ)
- account는 은행의 계좌번호와 유사한 개념으로 2가지가 있음
 - * : 비트코인에 쓰이는 것과 같은 계좌
 - * : 스마트컨트랙트에 쓰이는 계좌
- GAS : 모든 트랜잭션을 처리하는 데 필요한 수수료(채굴자들의 보상)

7. Solidity

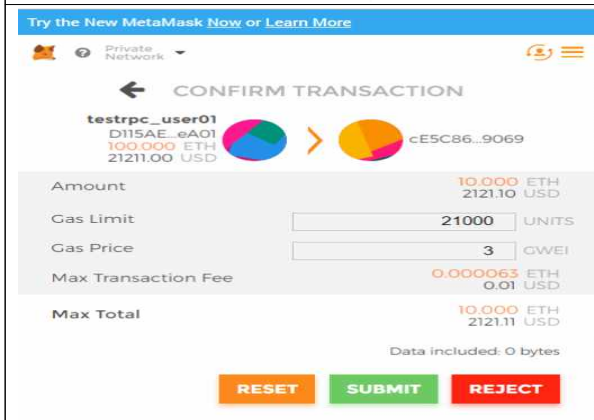
- 이더리움을 위한 언어로, JS와 유사하며 컴파일러가 필요!!(Bytecode로 변환)
현재 v0.4.23(초기 설정이 중요)
- <http://remix.ethereum.org/#optimize=false&version=soljson-v0.4.25+commit.59dbf8f1.js>
 - * 변수형에 address타입이 존재
 - * uint는 unsigned int의 약자로 부호 없는 정수형(기본 32byte)
- p153의 그림12-3을 참고!
위의 링크에서 solc를 통해 작성한 코드를 compile해서 EVM을 통해 Deploy(배포)하는 과정!
우분투에서 아래 코드를 입력하면 key가 생성됨(잘 기억해둬야 로그인 가능!)
`testrpc -m "apple banana mango hihi" // 한 단어 당 10자를 안 넘고 총 12단어를 넘지 않으면 됨!(고정 key)`



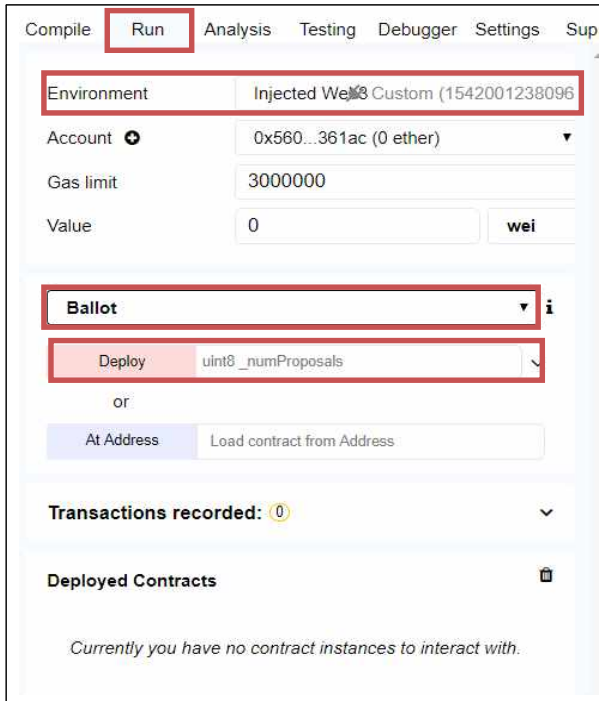
- ① 버전은 0.4.0을 선택 → auto compile 선택 → compile을 누르면 작성한 코드가 '바이트코드'로 변경 → 맨 아래 초록색 창에 'Ballot'이 뜨면 성공한 것!!



- ② 하기에 앞서 **MetaMask(여우 있는 앱 추가해서)** 새로운 계정 하나 import 후 실시하자! 클릭 후 비밀번호 입력은 ubuntu에서 'testrpc' 해서 나온 private key값 이용(1)



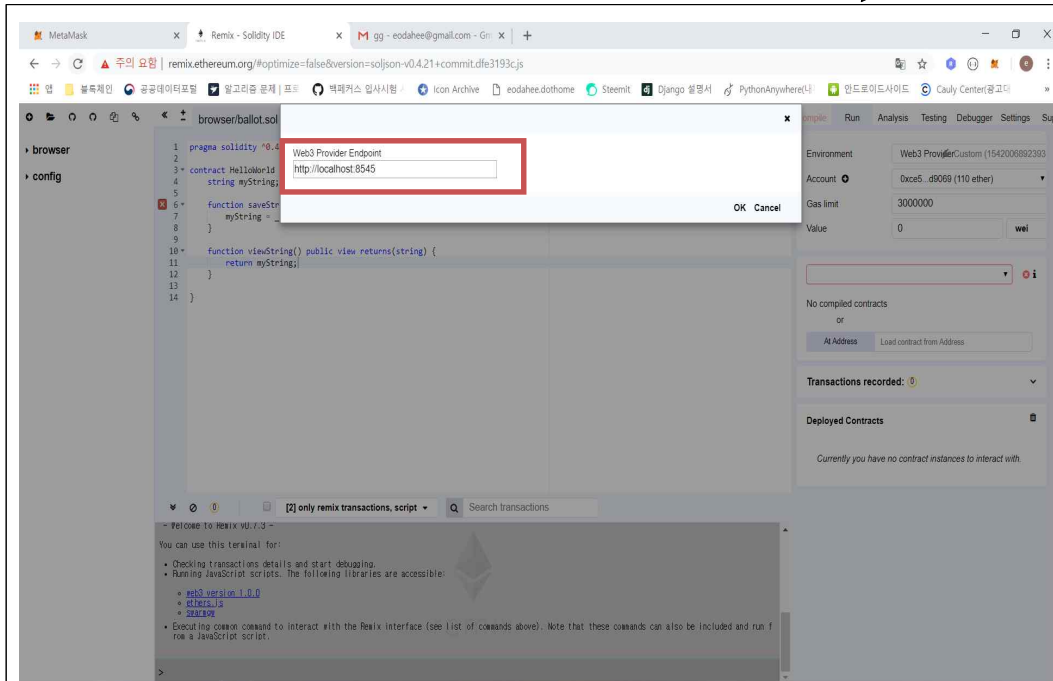
- ③ ether 보내는 것이 가능한지 확인 (Gas는 원하대로 설정가능)



④ Compile을 했으면 EVM을 통해 블록에 쓰여 질 수 있도록 'Run'을 해줘야함 → Environment를 Web3로 설정 → 'Ballot'써있는 곳에 원하는 contract 선정 → 'Deploy' 누르면 배포!!(블록에 올라감)

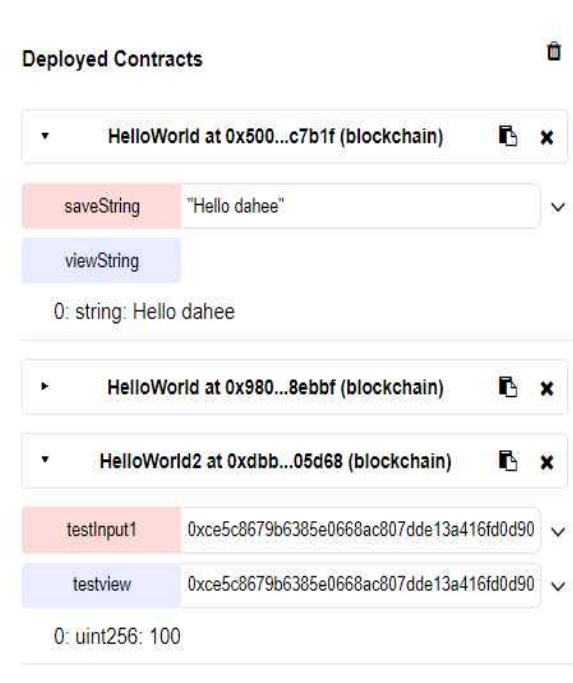
* 현재는 testnetwork이기 때문에 mining의 과정이 따로 없이 바로 블록에 쓰여지지만, 실제라면 채굴 되기 전까지 padding 상태로 대기한다고 생각!

<https://www.cryptokitties.co/catalogue> 이더리움을 이용한 새로운 커뮤니티(굉장히 성공적임)




ubuntu에서 설정한대로 (포트포워딩) 주소 써주면 가상환경의 port와 매핑 되서 접속 가능!!

즉, 실제로 이더리움을 돌리면 돈이 나가니 test 버전으로 하는데, 가상환경을 이용한 것!



⑤ 분홍, 보라색이 내가 작성한 함수! → 입력 또는 view 하고자 하는 데이터를 입력 후 클릭하면 좌하단에



이런 식으로 뜸.(화살표클릭하면 세부내용 나오는데 정보 확인) 소모된 GAS, 주소, 해쉬값 등 표현!

* 현재는 test network이기 때문에 client단이 없음! 채굴 과정 없이 그냥 진행(실제로는 mining될 때 까지 대기 필요)

- solidity 문법 : 맨 위에 'pragma'로 버전 명시
 - : contract가 존재(다른 언어와 다른 것)
 - : 새로운 변수인 address가 존재(다른 언어와 다른 것) : 유일한 값!!
 - : 문장의 마지막에 ;으로 종료!!
 - : 대소문자 구분, //로 주석 가능(/** */은 여러 문장)
 - : 데이터 타입 int 앞에 u를 붙여 unsigned로 가능(부호 없는 정수만)
 - : 배열은 'uint[10] tel' 이런 식으로 tel이란 이름으로 uint형 10칸짜리 방 할당
 - : 배열에 데이터 추가는 .push() 명령어를 사용
 - : 배열의 공간을 지정하지 않고 선언할 경우, gas가 더 나가서 좋은 방법 아님!
 - : view를 제외한 거의 모든 함수에 gas가 나간다고 생각(코딩에 유의)
 - : Mapping은 파이썬의 dictionary와 유사개념으로 키-값 쌍으로 이루어진 데이터타입
 - ex) mapping(uint => address) seller; // seller[uint] = address 라는 뜻
 - : 함수 = function = method(객체지향 언어에서의 함수, class 단위)
 - function 함수명(변수) [visibility] [modify] [return] [실제 실행내용]
 - public / private view, add 등 { }안에 작성!
 - :

```

1  pragma solidity ^0.4.21;
2
3  contract toyAuction {
4      address public theHighestBidder;
5      uint public theHighestBidPrice = 0;
6      string public itemName = "My Picture";
7
8      // do auction
9      function doBid() public payable {
10         require(msg.value > theHighestBidPrice); // if
11         uint refund = theHighestBidPrice;
12         address currentBidder = theHighestBidder;
13
14         theHighestBidder = msg.sender;
15         theHighestBidPrice = msg.value;
16
17         currentBidder.transfer(refund); // payback
18     }
19
20     // get current status
21     function checkBid() public view returns(address, uint, string) {
22         return(theHighestBidder, theHighestBidPrice, itemName);
23     }
24 }

```

- 경매하는 코드(solidity 언어로 작성)
- 버전은 0.4.21사용(pragma)
- contract의 이름은 toyAuction이며 변수는 3가지 선정(address, uint, string)
- 경매 참여하는 사람의 주소, 배팅 금액, 아이템이름 3가지!
- 함수는 2가지(doBid와 checkBid)
 - * 배팅해서 기존 금액보다 클 경우, refund에 이전 금액 대입, 주소에 현 사람 대입해서 payback(transfer)
 - * 경매 종료 후 현재 상태 출력은 [주소, 가격, 상품명] 순으로 반환!

< 이더리움 네트워크 실습 >

2018. 11. 13(화)

1. 복습

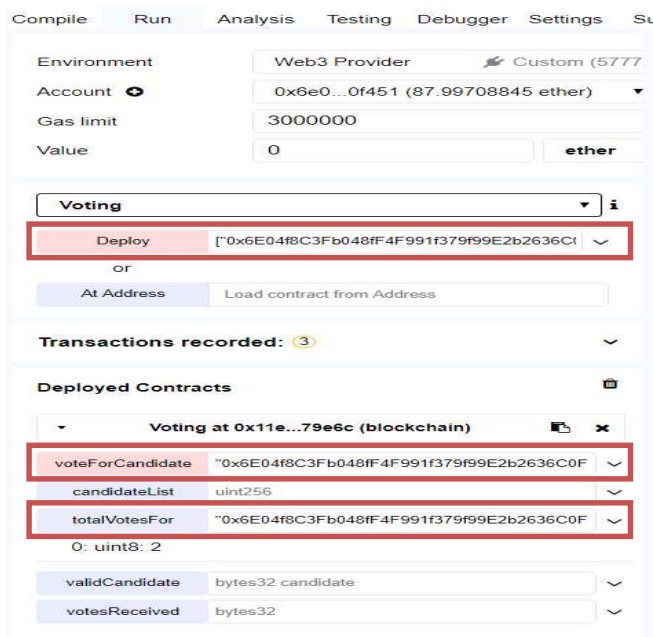
- 이더리움이 블록체인 네트워크 플랫폼을 최초로 제공해줌
- 팀프로젝트 할 때 웹페이지를 먼저 만들고 → DB제작 → 일부를 블록체인화 구현하는 것 추천!
- 프레임워크? 어떤 프로그램을 만들 수 있도록 규칙을 잔뜩 모아놓은 것
규칙만 지키면 원하는 프로그램을 금방 만들 수 있음(ex - Django 등)
- solidity? 역시 이더리움을(토큰) 써서 원하는 목표를 만들기 위해서 ERC를 지켜야 함.(표준 토큰문서)
[<https://tokenmarket.net/> 토큰 살펴보기] : 'ICO calender' 누르면 토큰 종류들 짝 나옴.
클릭해서 'Technology' 누르면 뭐로 구현했는지, Git에 관련 코드도 공개됨!!
- GAS의 limit은 transaction의 GAS와 block의 GAS 두 가지 종류가 있음
거래에 쓰이는 수수료 블록이 가질 수 있는 최대 가스량

2. 블록체인은 거래내역이 담긴 블록들을 전체 노드가 공유하는 것이고 그 블록 안에 내용도 모두가 공유하고 승인하게 됩니다. 이렇게 A가 B에게 보내는 트랜잭션이 담긴 블록이 전파되고 그 계약을 모든 노드가 가지고 있는 상태를 이더리움에서는 State라고 한다.

이더리움의 트랜잭션도 송금을 하거나 계약을 실행시키기 위해 수수료를 내야합니다. 그리고 그 수수료는 트랜잭션 안에 담겨있습니다.

nonce	이 Account에서 이제까지 생성한 트랜잭션 수(카운터)
gaslimit	트랜잭션을 실행하기 위해 지불할 용의가 있는 최대 gas 수량
gasprice	gas 1단위 당 가격
to	받는 사람 주소
value	수신자에게 전송한 이더의 양
서명	송신자가 이 계약이 자신이 만든 것임을 입증하는 서명
data	컨트랙트 메시지를 담을 수 있는 데이터 필드

3. Remix 사용법(교재 p153참고) : 코딩 vote예제



- contract를 코딩할 때 **constructor(생성자)**를 지정해줬기 때문에 deploy 할 때 주소를 써줘야 함!!
써주는 방법은 ["~~~, ~~~, ~~~"]

- voteForCandidate에 주소를 써주면 해당 주소값에 해당하는 uint가 +1 됨
- totalVotesFor에 주소를 써주면 해당 주소값에 해당하는 득표수(uint)가 return됨

3. Django 프로젝트 만들기

- ABI(Application Binary Interface)가 중요

응용프로그램과 OS, 응용프로그램과 해당 라이브러리, 응용프로그램의 구성요소 간 사용되는 낮은 수준의 Interface API는 소스코드에서 사용되고, ABI는 바이너리에서 호환됨! / 함수 호출 규약을 정의함 / 외부 라이브러리 사용시 중요!

Remix의 compile 하단에 있는 버튼 눌러보면 JSON 형식으로 내가 입력한 contract의 함수에 대한 정보들이 짝 나옴(이를 static/index.js 파일에 써서 연결해주기)

II

```
{
  "constant": false,
  "inputs": [
    {
      "name": "candidate",
      "type": "bytes32"
    }
  ],
  "name": "voteForCandidate",
  "outputs": [],
  "payable": false,
  "stateMutability": "nonpayable",
  "type": "function"
},
{
  "inputs": [
    {
      "name": "candidateNames",
      "type": "bytes32[]"
    }
  ]
}
```

이를 JSON.parse해서 JS형태로 변환
후 데이터 가져다 쓰기!!