Blockchain Programming

Blockchains with SmartContract

Blockchain 2.0

- Ethereum
- Eris
- Hyperledger

Contract

- A distributed contract is a method of using Bitcoin to form agreements with people via the block chain.
- Contracts don't make anything possible that was previously impossible, but rather, they allow you to solve common problems in a way that minimizes trust.

- Minimal trust often makes things more convenient by allowing human judgements to be taken out of the loop, thus allowing complete automation.

- Contracts are transactions which use the decentralized Bitcoin system to enforce financial agreements.

- Bitcoin contracts can often be crafted to minimize dependency on outside agents, such as the court system, which significantly decreases the risk of dealing with unknown entities in financial transactions.

Example: Providing a deposit

Imagine that you open an account on a website (eg, a forum or wiki) and wish to establish your trustworthiness with the operators, but you don't have any pre-existing reputation to leverage. One solution is to buy trust by paying the website some money. But if at some point you close your account, you'd probably like that money back. You may not trust the site enough to give them a deposit that they are tempted to spend. Another risk is that the site might just disappear one day.

The goal is to prove that you made a sacrifice of some kind so the site knows you're not a spambot, but you don't want them to be able to spend the money. And if the operators disappear, you'd eventually like the coins back without needing anything from them.

Programming Languages

- Solidity
- Codius
- C#, F#
- Node.js
- Go
- etc.

Programming on Ethereum

- Using Solidity Language (JavaScript - like)
- Writing a .sol file
- Compile into 2 files:
  - ABI file (.abi)
  - Byte codes (.bin)
- Deploy to the Blockchain
- Create a new contract on a Blockchain
- Call the contract

Solidity Sample 1

```
contract HelloSystem {

    address owner;

    // Constructor
    function HelloSystem(){
        owner = msg.sender;
    }

    function remove() {
        if (msg.sender == owner){
            selfdestruct(owner);
        }
    }

}
```

Solidity Sample 2

```
contract Users {
    // Here we store the names. Make it public to automatically generate an
    // accessor function named 'users' that takes a fixed-length string as argumen
    mapping (bytes32 => address) public users;

    // Register the provided name with the caller address.
    // Also, we don't want them to register "" as their name.
    function register(bytes32 name) {
        if(users[name] == 0 && name != ""){
            users[name] = msg.sender;
        }
    }

    // Unregister the provided name with the caller address.
    function unregister(bytes32 name) {
        if(users[name] != 0 && name != ""){
            users[name] = 0x0;
        }
    }
}
```

Solidity Sample 3

```
contract EtherVote {

    event LogVote(bytes32 indexed proposalHash, bool pro, address addr);

    function vote(bytes32 proposalHash, bool pro) {
        // don't accept ether
        if (msg.value > 0) throw;
        // Log the vote
        LogVote(proposalHash, pro, msg.sender);
    }

    // again, no ether
    function () { throw; }
}
```

Solidity Sample 4

```
contract Multiplication {

    int _multiplier;
    event Multiplied(int indexed a, address indexed sender, int result );

    function Multiplication(int multiplier) {
        _multiplier = multiplier;
    }

    function multiply(int a) returns (int r) {
       r = a * _multiplier;
       Multiplied(a, msg.sender, r);
       return r;
    }
}
```

Dev Env and Frameworks

- Mix standalone IDE by ETHDEV
- in-browser app that connects to `geth` via RPC. By Nick Dodson
- `embark` framework by Iuri Mathias
- `truffle` by Tim Coulter

Beware the impossible smart contract

The three most common smart contract misconceptions

- Contacting external services
- Enforcing on-chain payments
- Hiding confidential data