Topicos
Aula Smart Contracts

Ivan Sendin

FACOM - Universidade Federal de Uberlândia
ivansendin@yahoo.com,sendin@ufu.br

24 de setembro de 2024

- Consenso distribuído
  - Prova de Trabalho
    Prova de participação
  - Loteria
  - Mineração Egoísta
    p-value

- Criptografia
- Hashing e propriedades
  Paradoxo do Aniversário
- Compromisso
- Árvore de Merkle
- Filtro de Bloom
- PLD e DH
- (ECC)
- Paradoxo do Aniversário

- Transações: UTxO
- Endereços 1BitcoinEaterAddressDontSendf59kuE
- pseudo anonimato

- Ecossistema
- exchanges
- ATM
- Cassino
- Marketplaces/DNMs
- mixers
- ransomware

- Estatísticas: Gini e Benford
- H1
  Troco
- OSINT

- Caso do Farao
- H1
- Ampliamos o número de carteiras conhecidas
- Uma carteira deve ser exchange
- análise de Blockchain (grafo)
- (H1 das carteiras que pagam o faraó)

- ML aplicada a análise de Blockchain

Smart Contracts

- Computador Descentralizado
  Diferente da Nuvem
- Ethereum Principal "ecossistema" de Smart Contract
- Principal linguagem: Solidity

Ethereum - Somente o necessário

- Blockchain
- Mineradores além de validar as transações (simples) também executam os contratos
- As contas possuem saldo
- Os contratos são **imutáveis**
  Padrão de desenvolvimento pode ajudar, botão de pânico
- Ambiente de denvolvimento é mais complexo....

Ethereum - Somente o necessário

- Os contratos são desenvolvidos em *Solidity*
  Existem outras
- Ethereum bytecode - EVM
- São "entregues ao Ethereum"(deploy)
  Gera um identificador único(hash)
  Uma vez entregue o contrato esta autônomo

Ethereum - Somente o necessário

- Decentralized App (DApp)
  - Envia transações para Contratos
  - GUI,IPFS,...
- Iteração pela ABI - Application Binary Interface
  - $bytes4(keccak256("funcao(bytes,bool,uint256[])")$[1]
- web3.js
  Nó ethereum (local ou remoto) via HTTP, IPC ou WebSocket

[1] "ketchak"

Desenvolvimento

- Hard Hat
- Foundry
- Remix

```
contract Leilao {
   enum EstadosLeilao { LancesEstado, FinalizadoEstado}
   EstadosLeilao meuEstado;

   mapping (address => uint) lances;
   uint blocklimit;
   address vencedor;
   address payable dono;
   uint vencedorval;
   bool pagou;

   constructor(address payable d,uint tempo) public {
     blocklimit= block.number + tempo;
     meuEstado = EstadosLeilao.LancesEstado;
     vencedorval = 0;
     dono =d;
     pagou = false;
   }

...
```

Os primeiros...

- Nick Szabo
- Vending Machine
- Existem um "contrato" implementado pela mecânica da máquina
- Até certo ponto **confiável**
  - Código confiáel
  - Computador confiável
  - (todas as partes)
- O Código é a lei!
- Hybrid smart contracts will replace the legal system

- Os SC não possuem uma "complexidade"[2] algoritmica...
- (de forma alguma ele são fáceis!!)
- (e sempre são sensíveis!!)
- Não tem exercício de Fibonacci, fatorial ou LeetCode em SC
- Que problemas são resolvidos usando SC???

[2] Força de expressão

- Soluções que envolvam partes com interesses antagônicos
  Comprador X Vendedor
- Soluções que comportamentos honestos possam ser guiados/induzidos
  Não existe Procon
- Soluções que possam ser adaptados do mundo real para os SC
- **Soluções completamente novas!**

- Financeiros: compra e venda, leilões, DeX, emprestimos com ou **sem garantias**, seguros,...
- Governo: votação, sistema monetário, pagamento de tributos, saúde, distribuição de riquezas, seguridade, veículos,
- Identidade Digital
- Cartorios
- Cadeia de suprimentos
- Testes clínicos

- Em algum momento eu posso implementar um leilão
- (fácil em C,Python, Rust,...)
- Se vc rodar em um computador confiável as pessoas vão confiar no seu leilão
- (código fonte aberto)

Exemplo - Leilão

- Inicia o leilão
- Escuta os lances
- No fim, indica o vencedor.
- Legal!!
- Mas...

Um contrato tradicional

- Participantes
- Obrigações
- **Penalidades**

- O exemplo do leilão não é um SC....
- E se o ganhador não pagar??
- E se o dono do objeto leiloado não entregar o produto?
- Sistema juridico tradicional

- Algumas definições de SC incluem o termo *self-executing contract*
- **Importante!!**
- Um SC deve ser completo, auto-contido
- Não é a linguagem, não é a blockchain, não é criptomeda que fazem um programa ser um SC
- ~~Se não fizer isso, sistema judiciario!~~

- O ecossistema blockchain/criptomoedas permite:
- **garantir** a entrega do produto
  - Token...**NFT**

- O ecossistema das criptomoedas permite:
- **garantir** a entrega do produto
- **garantir** o pagamento
  - (nem sempre) literalmente
  - Multa, caução/collateral.
    Exemplo: leilão de um produto que deve valer R$1.000 exige um depósito de R$50
  - *Força um comportamento honesto*. Teoria dos Jogos.

Finalizando...

- *self-executing contract*
  Auto contido...não precisa pedir ajuda de fora
- *trustless*
  Não confie nos participantes, **apenas no contrato**

```
pragma solidity >=0.4.25 <0.6.0;


contract AlmostSmartAuction {
  enum AuctionStates { BidState, FinishedState}
  AuctionStates myState;

  mapping (address => uint) bids;
  uint blocklimit;
  address winner;
  uint winnerBid;

  constructor(uint auctionTime) public {
    blocklimit= block.number + auctionTime;
    myState = AuctionStates.BidState;
    winnerBid = 0;
  }
```

```
  function bid(uint bidValue) public {
    verifyFinished();
    require ( myState ==  AuctionStates.BidState, "Bids closed...");
    if (bidValue > winnerBid) {
      winnerBid = bidValue;
      winner = msg.sender;
      }
  }


  function verifyFinished() private {
    if (block.number > blocklimit) {
      myState = AuctionStates.FinishedState;
      }
    }

  function isWinner(address who) public returns (bool) {
    verifyFinished();
    require ( myState ==  AuctionStates.FinishedState, "Be patient...");
    return who==winner;
  }
}
```