Criptografia - Mais Cripto

Ivan Sendin

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

5 de agosto de 2024

Dois jogadores de xadrez mental ficaram entediados.

"Vamos jogar poker mental", disse o primeiro.

"Sim. Eu dou as cartas!", respondeu o outro.

Shamir, Rivest, Adleman; Mental Poker

Aulas Passadas

- História, dinheiro,...
- Hashing e aplicações
- Prova de trabalho
- Mineração e Consenso Distribuído Simulação
- Selfish
- ...

Aulas Passadas

- New Directions in Cryptography / Diffie-Hellman
- Merkle
  Lei de Stigler
- Estabelecimento de chave
- $g^{ab} \pmod{p}$
- p e g precisam ser devidamente escolhidos
- Problema do Logaritmo Discreto
  Vale a pena estudar
- Mixing, Monero, Zcash,...

Aulas Passadas

- Curvas Elipticas
- Cultura Geral
  Obrigatoria para profissionais criptografia, criptomoedas,etc...
- Livro “Zero to Monero”

Aulas Passadas

- Assinatura Digital
- PK e SK
  $(PK, SK) = geraPar(Random)$
- $a = Assina_{SK}(M)$
- Mensagem assinada: (M, a)
- $Verifica(a, PK, M)$
- O **endereço** Bitcoin é a PK

Bitcoin's academic pedigree

- Bitcoin foi "iventado"pelo Satoshi Nakamoto
- Será? Como?

"Se encherguei mais longe é pq estava nos ombros de gigantes" - Newton(?)

nani gigantum humeris insidentes

Bitcoin's academic pedigree

- **Bitcoin's academic pedigree** Narayanan, Arvind Clark, Jeremy.
- Timestamping/log seguros: merkle Tree, encadeamento (Haber/Stornetta)
- Digital Cash: sempre esbarravam centralização
- PoW: comprometimento com o sistema. HashCash
- Tolerância a Falhas: Generais Bizantinos
- PK como identidades
  Security Whitout Identification/Chaum
- Blind Signatures
  Chaum de novo
- Smart Contracts: Szabo

Bitcoin's academic pedigree

- Minhas inserções
- Filtros De Bloom
  ED fantástica!! Quase mágica!! $O(1)$, probabilística e privacidade!
- Merkle Patricia Trie
- DHT
- Zero-knowledge proof
  $R = A + B$ é verdade....

Funções de Hashing

- Unidirecionalidade
  $H(x)$: ok H^{-1}: não existe!
- Colisões
  $H(x_1) == H(x_2)$: não!
- 2a pré-imagem
  Dados $y = H(x)$...não encontro $x'|y = H(x')$

O hash code é uma impressão digital de um string de bits!!

Compromissos - Commitments

- *Commitment Schemas*
- Eu preciso me **comprometer** com uma informação
- Eu não posso/quero revelar a informação **no momento**
- Manuel Blum em Coin Flipping by Telephone
- Shamir, Rivest, Adleman em Mental Poker

Compromissos - Commitments

- Um *Commitment Schemas* tem duas fases:
  1. Compromisso
  2. Abertura ou revelação

O seu funcionamento é ridulamente simples!!!
(OK! Tem detalhes...)

Compromissos - Commitments

- (Compromisso) Alice conhece M e gera $h = \mathcal{H}(M)$
- Alice envia h para Bob
- (Abertura) Em algum momento do futuro Alice envia M' para Bob
- Bob verifica se $M' == M$
- Fim!!

Compromissos - Commitments

- Bob tem condições de saber se Alice teve um comportamento honesto??
- $M \stackrel{?}{=} M'$
- $h \stackrel{?}{=} \mathcal{H}(M')$

Compromissos - Commitments

- **Bob consegue determinar M?**
- Não, propriedade de unidirecionalidade de $\mathcal{H}$
- **Bob tem condições de saber se $M == M'$??**
- Sim, propriedade de resistencia a colisão de $\mathcal{H}$
- **Bob tem condições de gerar um M^{Bob} e trapaçear??**
- Não, resistência a 2a pré-imagem

Compromissos - Commitments

- Então se Alice e Bob combinam de usar "cara" e "coroa" não fica difícil para Bob "inverter" o valor h...
- basta tentar calcular o hash de "cara" e comparar...
- ... e depois (se for necessario) repetir com "coroa"
- Quando o espaço de entrada das funções de hashing for pequeno, um uso *ingênuo* das funções de hashing permite a sua inversão
- Para resolver esse problem, vamos **aumentar o espaço de entrada**

Compromissos - Commitments

- Um *nonce* é uma sequencia de bits aleatórios usado uma única vez em um protocolo
- O commitment de um valor v usando um *nonce* n pode ser feito por

$$\mathcal{H}(n|v)$$

- No passo de revelação n e v são enviados
- Se n tiver um tamanho grande (160 bits?) impede a força bruta

Compromissos - Commitments / Aplicações

- Leilão de Envelope fechado
- Os candidatos enviam envelopes fechados e lacrados ate uma data limite
- No dia do leilão, o leiloeiro abre os envelopes e determina o melhor lance
- (comum em licitações/privatizações)

Compromissos - Commitments / Aplicações

- Leilão de Envelope fechado
- O candidato i envia $h_i = \mathcal{H}(N_i|M_i)$
- Cada h_i é publicado
- Após a fase de compromisso, os candidatos revelam M_i e N_i
- O leiloeiro e os demais candidatos podem verificar a legitimidade de cada M_i
- (Estritamente falando, o candidato pode desistir de revelar M_i....mas uma **caução** pode obrigar a revelação!)

- Uma Árvore de Merkle é uma forma **eficiente** de fazer um compromisso com diversos "documentos"
- Espaço constante!!!
  Rede P2P é cara!!
- Custo log para revelar os valores

Como publicar informações...

- Um banco precisa publicar informações das transações do dia.
- $Tx_1, Tx_2, Tx_3 \ldots Tx_n$
- Verificação posterior....

$$Tx_k??$$

Como publicar informações...

- Um banco precisa publicar informações das transações.
- $Tx_1, Tx_2, Tx_3 \ldots Tx_n$
- Verificação posterior....
- Como fazer isso de forma eficiente?

Como publicar informações...

- Solução 1: publicar as transações
- (WhatsApp para os clientes, broadcast em uma rede...)
- Custo?
- Vantagem?

Como publicar informações...

- Solução 1: publicar as transações
- Custo?

$$\sum_{1}^{n} |Tx_n|$$

- Vantagem?
  Fácil...direto...

Como publicar informações...

- Solução 2: publica o hash das transações....
- Custo?
- Vantagem?

Como publicar informações...

- Solução 2: publicar as transações
- Custo?

$$|h| \times n$$

- Vantagem?
  Fácil...Fixo...direto...

Como publicar informações...

- Solução 3...
- Custo **espaço** pequeno...muito pequeno...
- $O(1)$

- Uma solucao simples seria publicar $H(Tx_1|Tx_2|Tx_3|Tx_4)$

- Uma solucao simples seria publicar $H(Tx_1|Tx_2|Tx_3|Tx_4)$
- Como verificar se Tx_1', Tx_2', Tx_3', Tx_4'?

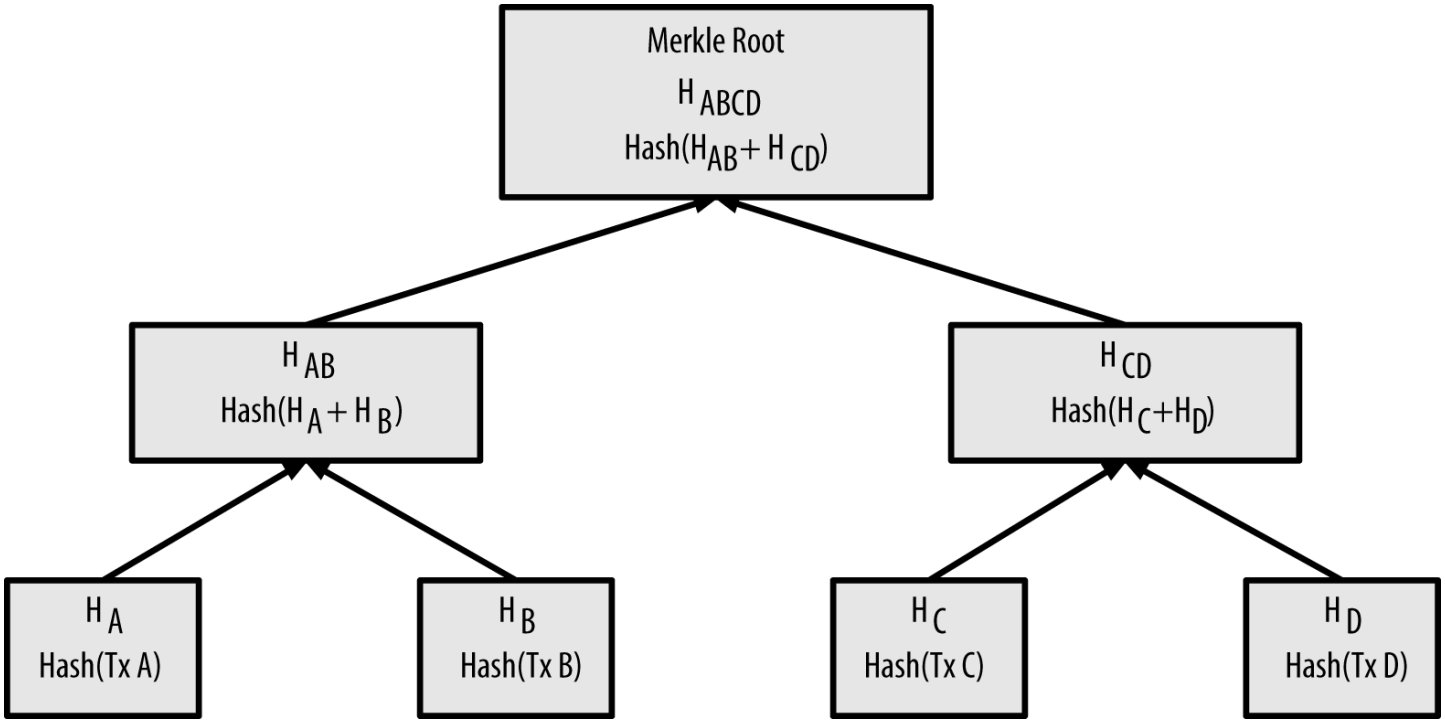
- Uma solucao simples seria armazenar $H(Tx_1|Tx_2|Tx_3|Tx_4)$
- Como verificar se Tx_1', Tx_2', Tx_3', Tx_4'?
- $H(Tx_1', Tx_2', Tx_3', Tx_4')$ e compara....

- Como verificar se Tx_1'?
- **Recuperar** todas as outras transações....
- $H(Tx_1', Tx_2', Tx_3', Tx_4')$
- Proibitivo para **muitas** transações....

- Como publicar apenas um hash e...
- Não usar todas as transações para verificar uma Tx_k??

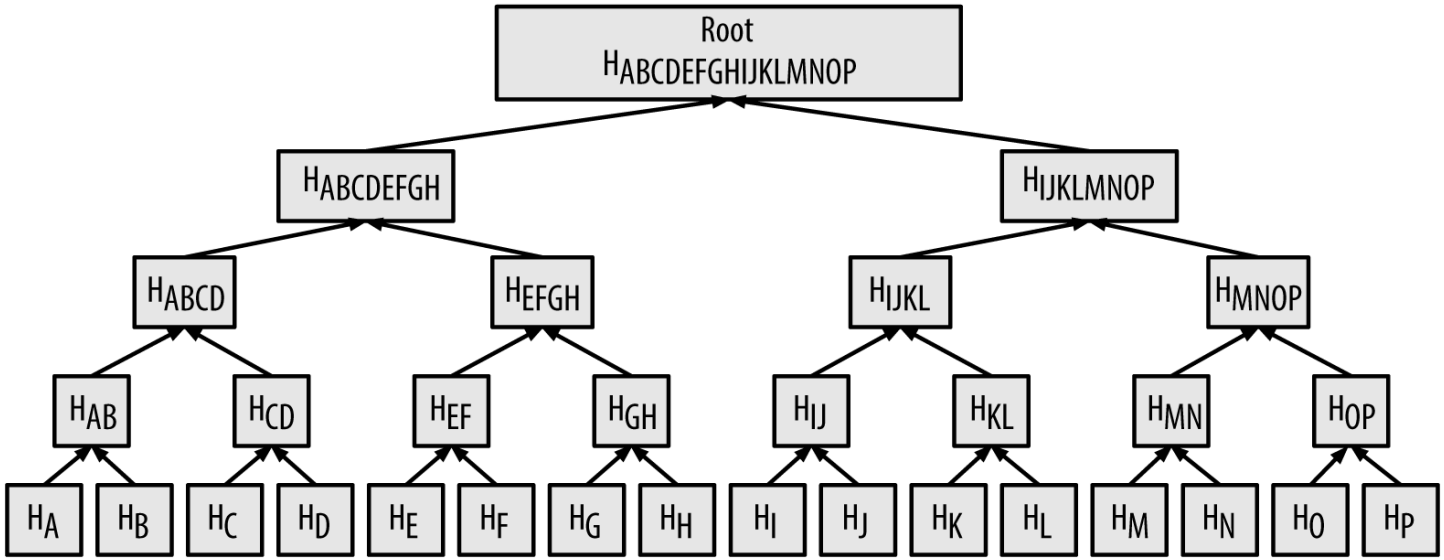
Merkle Tree

- Montar uma arvore de hashes
- e armazenar apenas a raiz


Merkle Root
H ABCD
Hash(H AB + H CD)
H AB
Hash(H A + H B)
H CD
Hash(H C+H D)
H A
Hash(Tx A)
H B
Hash(Tx B)
H C
Hash(Tx C)
H D
Hash(Tx D)

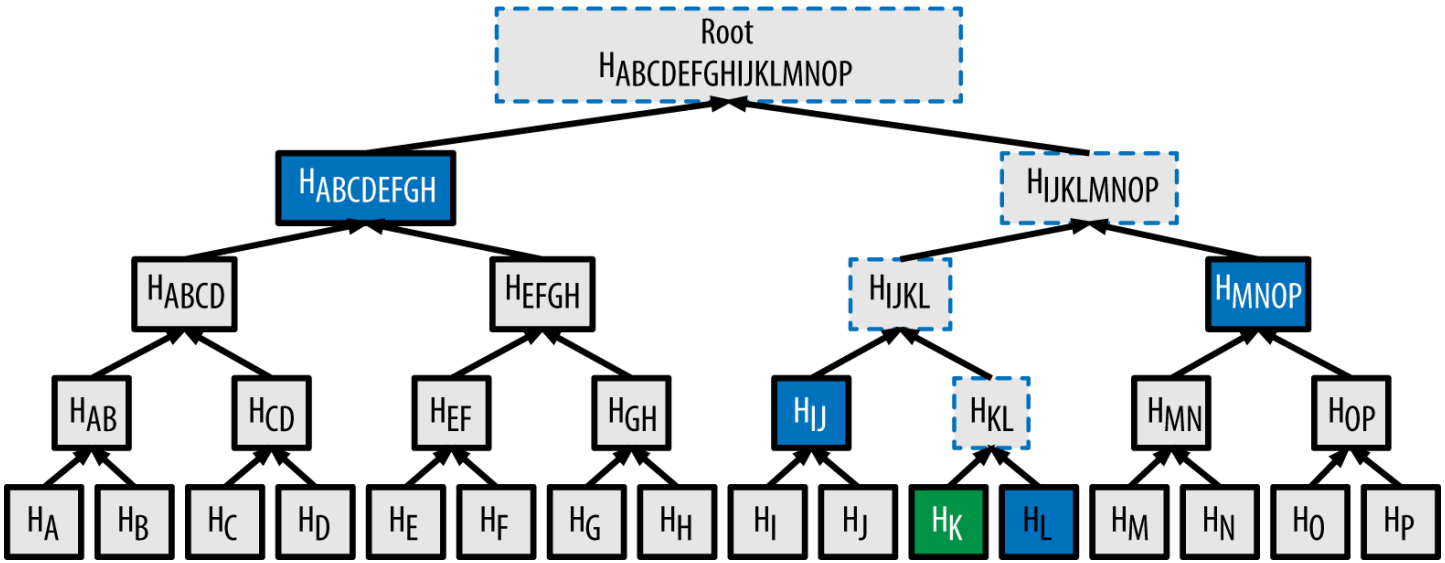
Merkle Tree

- Folhas impares sao solucionadas com duplicacao
- Como toda arvore, a altura e dada por *log*


Root
HABCDEFGHIJKLMNOP
HABCDEFGH
HIJKLMNOP
HABCD
HEFGH
HIJKL
HMNOP
HAB
HCD
HEF
HGH
HIJ
HKL
HMN
HOP
HA
HB
HC
HD
HE
HF
HG
HH
HI
HJ
HK
HL
HM
HN
HO
HP

Merkle Tree

- A raiz é publica...
- Dada uma transacao K como eu verifico se ela e valida ($=$ esta na arvore?)?


Root
HABCDEFGHIJKLMNOP
HABCDEFGH
HIJKLMNOP
HABCD
HEFGH
HIJKL
HMNOP
HAB
HCD
HEF
HGH
HIJ
HKL
HMN
HOP
HA
HB
HC
HD
HE
HF
HG
HH
HI
HJ
HK
HL
HM
HN
HO
HP

Merkle Tree

- Usada no bitcoin/blockchain
- Cada bloco do blockchain armazena a raíz da Merkle Tree das transações "validadas" neste bloco
- Quando alguem que provar que uma determinada Tx pertence a um determinado bloco....
- .. envia ao verificador apenas as informações necessárias $O(\lg n)$