Sistemas Distribuídos
Aula 22

Roteiro

- Registrando transações
- Blockchains
- Eleição de líder (diferente)
- Construindo blocos (Merkle)
- Bitcoin

Transações Distribuídas

- Conjunto de usuários realizando transações
  - de forma distribuída, claro
- Queremos garantir duas propriedades

  1) Transações não podem ser refutadas ou removidas

  2) Transações precisam de ordenação global
- Exemplo

  U_1: $T_{1,1}$, $T_{1,2}$, ...

  U_2: $T_{2,1}$, ...

  U_3: $T_{3,1}$, $T_{3,2}$, $T_{3,3}$, ...

Sistema transacional *to the rescue*!

- Precisa pagar e confiar no dono do sistema
  - para não remover, não refutar, e garantir ordem

Transações Distribuídas

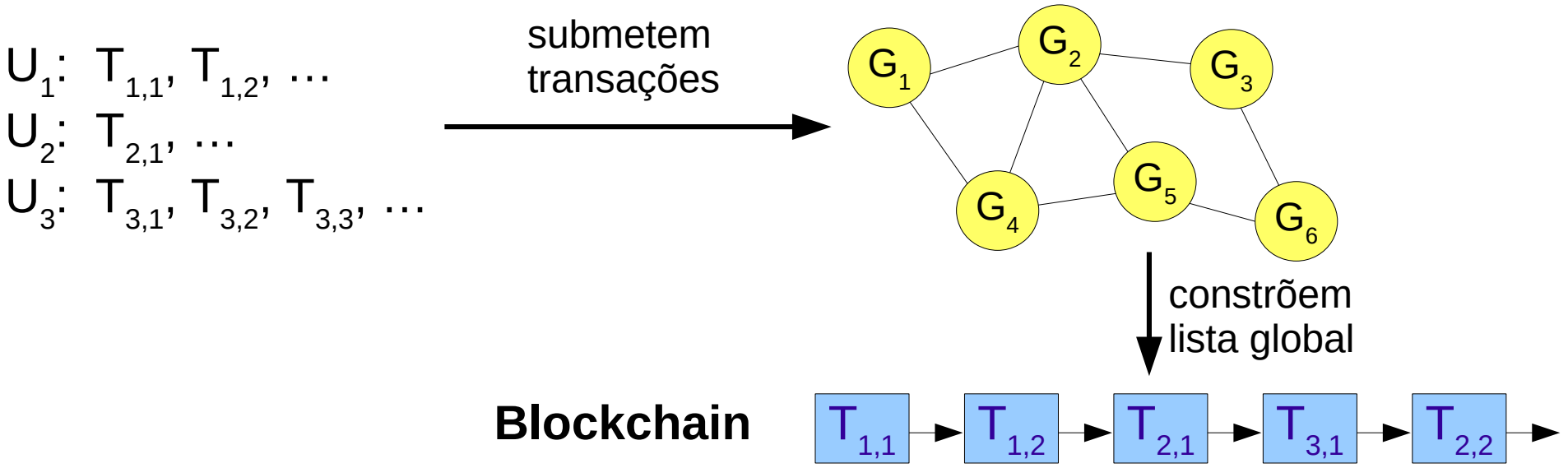
- Como garantir essas propriedades sem confiar uma única entidade?

funções de hash + assinaturas digitais + algoritmos de consenso

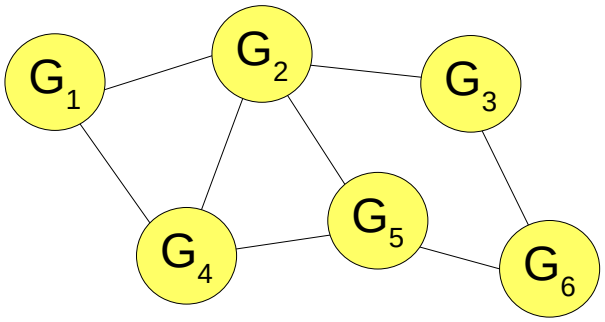
- Não repudiar: usuários assinam suas transações (chave pública/privada)
- Não remover: transações são inseridas em uma lista e assinada por terceiros
- Ordenação global: algoritmo de consenso entre terceiros determina ordem da lista

Blockchain

- Cada usuário possui chave pública/privada
  - toda transação é assinada com chave privada
- Terceiros: grupo (dinâmico) de processos que trabalha na construção e ordenação da lista
  - organizados em rede, adicionam transação ao final da lista, rodam algoritmo de consenso
  - lista está replicada em cada processo



Grupo de Terceiros



- Quem são os terceiros que formam este grupo?
- Precisamos confiar neles?

- Dois modelos
  - grupo restrito: precisa de permissão para participar, em geral membros são empresas
  - grupo irrestrito: qualquer um pode participar, incluindo pessoas físicas (e hackers)
- Incentivo: membros do grupo recebem comissão (dinheiro) ao inserir transação na lista
- Confiança: precisamos confiar que parte dos membros é honesto (o famoso 2/3)

Inserindo Transações na Lista

- Transações (assinadas) enviadas para todos do grupo (via broadcast na rede)
  - usuário pode enviar para mais de um do grupo
- Determinar processo que vai atualizar a lista

Eleição de líder (variação)

- Processo vencedor valida a transação, faz modificação, e envia para todos os outros
  - todos os outros validam a transação e a inclusão no final da lista (verificam assinaturas)
- Algoritmo de consenso garante que inclusão na lista é legítima (se ao menos 2/3 são honestos)
  - diferentes algoritmos são utilizados, em geral visando eficência na troca de mensagens

Eleição de Líder

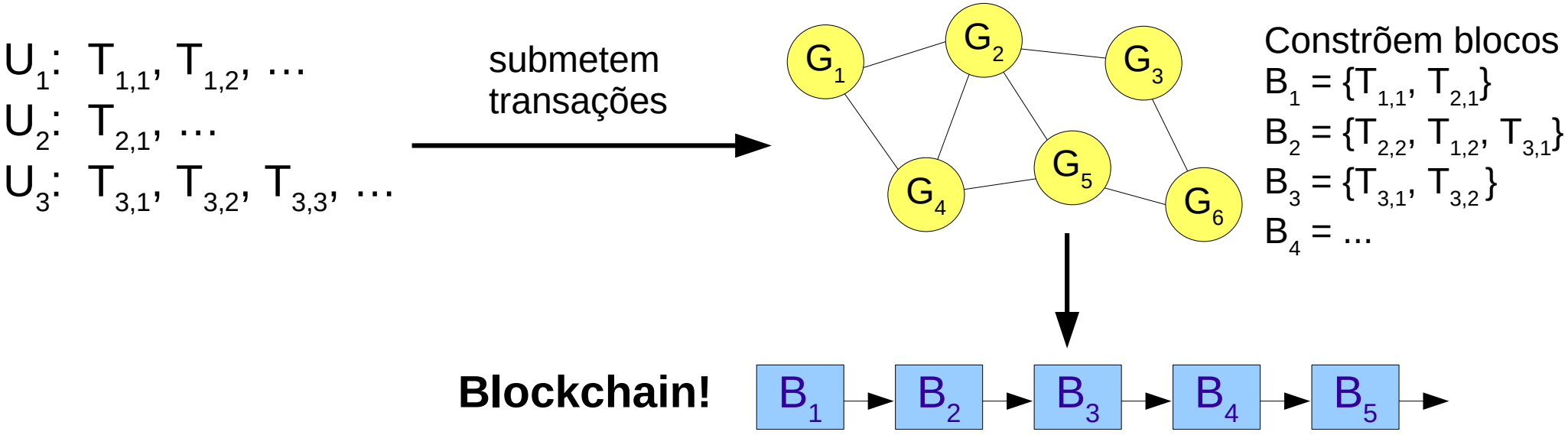
- Como determinar o processo que vai atualizar a lista?
  - maior ID não funciona (monopólio da ordenação das transações)
- Ideia: competição computacional entre os membros do grupo (*proof of work*)
  - competição: resolver um problema computacional que seja fácil verificar a solução
  - vencedor atualiza a lista (quem resolver primeiro)
- Problema: Dado x encontrar y tal que H(x,y) < D
  - H é uma função de hash, D é um parâmetro que controla a dificuldade (menor D é mais difícil)
  - trivial verificar se y é solução (basta calcular hash)
  - no Bitcoin: H(x,y) = SHA256(SHA256(x.y))

Proof of Work

- Se H é “boa” então melhor algoritmo testa valores para y escolhidos de forma aleatória
  - altamente paralelizável
- Chances de vencer proporcional (linearmente) ao poder computacional
  - vencedor apresenta valor de y
  - eleição não é justa neste sentido
- Vencedor deve ser aleatório
  - evita conhecer a priori quem vai atualizar a lista para cada transação
- Despediça energia (e poder computacional)
  - membros que perdem a eleição
  - bloco de transações vai ajudar

Bloco de Transações

- Ineficiente: atualizar a lista com cada transação (em tempo e memória)
- Ideia: bloco de transações
  - lista de blocos (ao invés de transações)
  - cada bloco contém k transações (distintas)
- Transações são registradas nos blocos

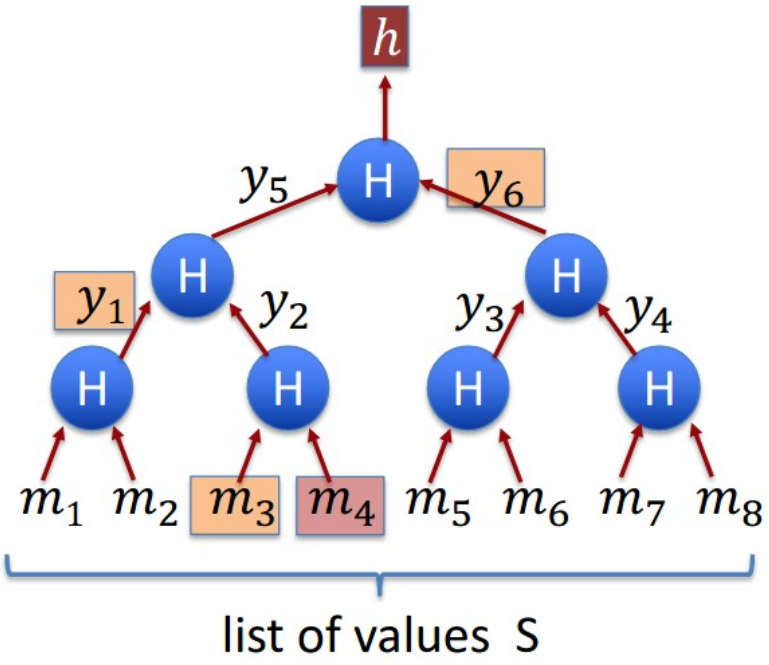


Construindo Blocos

- Membros do grupo conhecem todas as transações
- Escolhem subconjunto de transações para formar um bloco (respeitando ordenação local dos usuários)
  - competem para inserir bloco na lista
  - maior o bloco, maior a comissão
- Após inserido um bloco, todos os membros validam todas as transações do bloco
  - prova de validação é gerada pelo criador do bloco, e enviada a todos os membros
  - algoritmo de consenso: maioria precisa concordar que bloco é válido

Árvore de Merkle

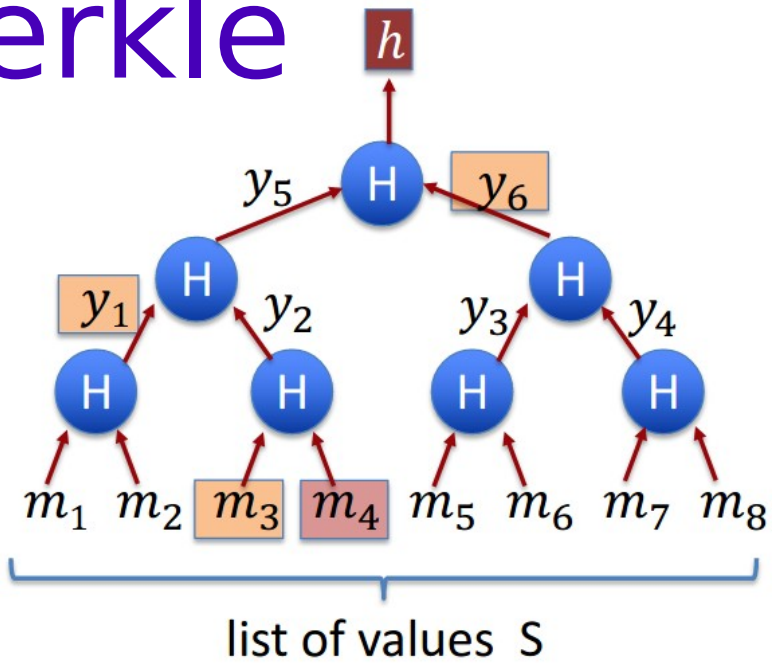
- Seja conjunto S = {m_1, m_2, ..., m_n}
- Estrutura de dados para mostrar que elemento está no conjunto
  - sem conhecer elementos do conjunto
  - sem poder modificar o conjunto



  - y = resumo do conjunto
  - H(x,y) = função de hash
  - y_1 = H(m_1,m_2), y_2 = H(m_3, m_4), ...
  - Árvore é contruída e valores y_j gerados
  - Cada valor m_i tem uma “prova”
    - irmão na árvore e valores de y_j até a raiz

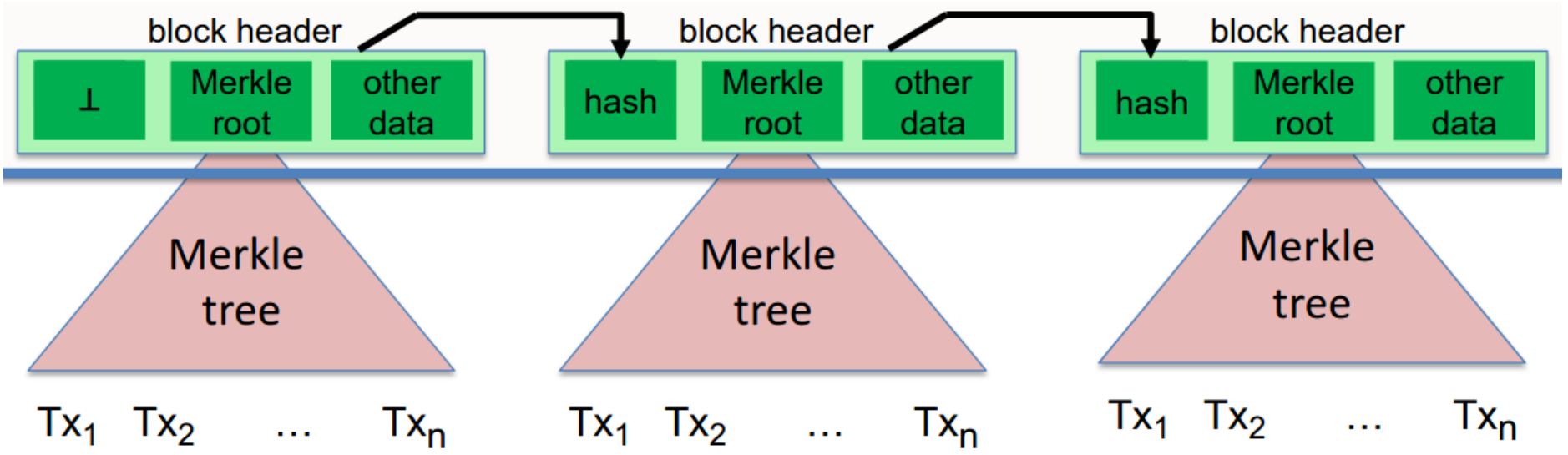
Árvore de Merkle

- Exemplos
  - Prova de m_4: m_3, y_1, y_6
  - Prova de m_7: m_8, y_3, y_5
- Prova tem tamanho $\log_2 |S|$



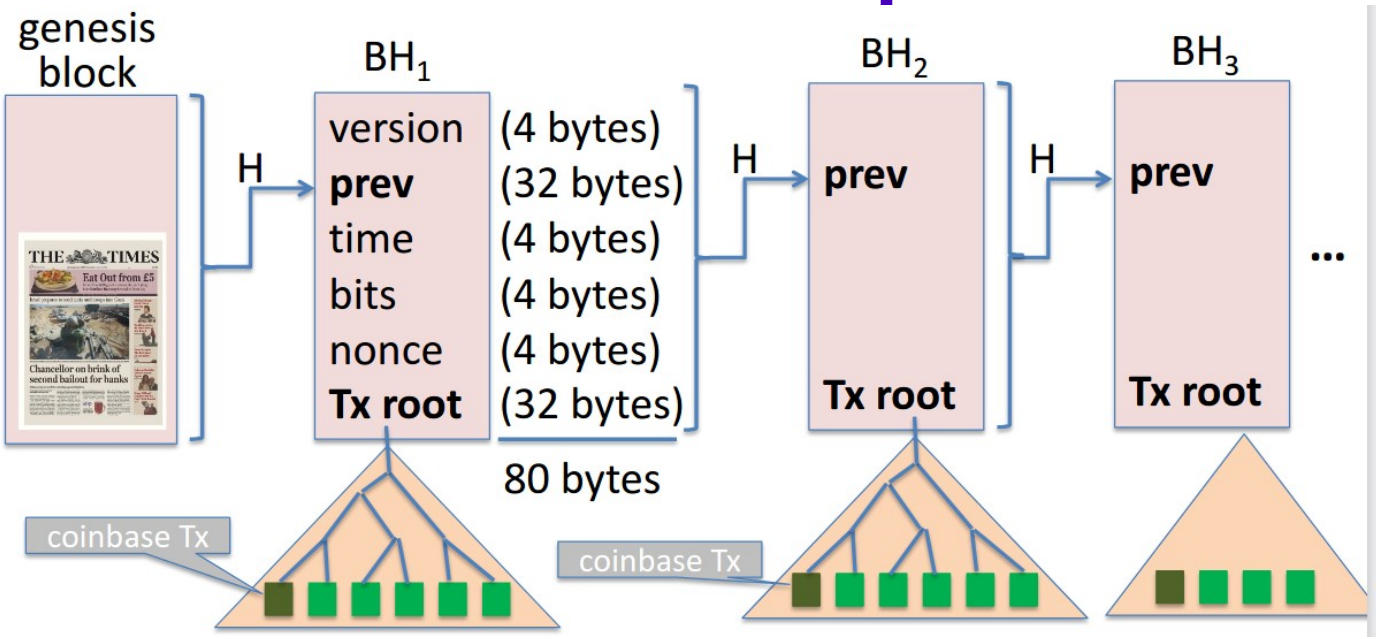
- Bob conhece h (registrado no blockchain)
- Como saber se m_4 pertence ao conjunto (bloco)?
  - Bob usa a prova de m_4 para verificar
    - se $H(y_6,H(y_1,H(m_3,m_4)) == h$ então OK!
  - Alice (dona da transação m_4) pode enviar a prova para Bob
- Se H for boa, impossível alterar as transações do bloco

Blockchain



- Cabeçalho possui hash do cabeçalho anterior (garante ordenação dos blocos)
- Cada bloco representa uma árvore de Merkle
  - cabeçalho do bloco tem tamanho fixo, apenas valor h é usado
- Lista de transações do bloco poder ser armazenada junto ao bloco

Exemplo Bitcoin



- Sequência da cabeçalhos de blocos de 80 bytes cada

- *version*: versão do sistema (define parâmetros, etc)
- *prev*: valor de hash do cabeçalho anterior
- *time*: hora que o bloco foi criado (UTC)
- *bits*: dificuldade do *Proof of Work* (PoW)
- *nonce*: prova do resultado do PoW
- *Tx root*: raiz da árvore de Merkel
- Novo bloco adicionado a cada ~ 10 minutos

Exemplo

Block 648493

Timestamp	2020-09-15 17:25	
Height	648493	
Miner	SlushPool	(from coinbase Tx)
Number of Transactions	2,826	
Difficulty (D)	17,345,997,805,929.09	(adjusts every two weeks)
Merkle root	350cbb917c918774c93e945b960a2b3ac1c8d448c2e67839223bbcf595baff89	
Transaction Volume	11256.14250596 BTC	
Block Reward	6.25000000 BTC	Bitcoin pago ao miner (moeda criada)
Fee Reward	0.89047154 BTC	(Tx fees given to miner in coinbase Tx)

Double Spending

- Bob faz duas transações, mas uma não é compatível com a outra
  - ex: Bob tem 100 reais, paga 80 para Aline (T_1) e 80 para Ana (T_2)
- Apenas uma transação deve entrar no blockchain
  - a outra não é válida, dada a primeira
- Bloco com a segunda transação tem que ser considerado inválido

Algoritmo de Consenso

- Para determinar se um bloco é válido
  - Bitcoin: PoW + votaçaõ com maioria

Algoritmos de Consenso

- Atacam o problema de falhas bizantinas
- Baseados em votação (como vimos aula passada)
  - Dovlev Strong, Streamlet, Nakamoto Consensus
- Implementados na prática em diferentes sistemas de blockchain
  - Bitcoin: PoW, votação simples (i.e., Lamport)
- Mais detalhes
  - Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction (livro)
  - Foundations of Consensus (livro aberto, mais teórico)
  - CS 251: Cryptocurrencies and Blockchain Technologies (curso em Stanford, https://cs251.stanford.edu/). Algumas figuras destes slides retiradas deste material.