Lecture 2: Introduction to Bitcoin

EE 374

April 2, 2025

Agenda

- Reading: Nakamoto's Bitcoin whitepaper
- Today's goals:
  - Two problems Bitcoin solved:
    - Data integrity – application layer
    - Data agreement – consensus layer
  - Two cryptographic primitives to solve them:
    - Digital signatures
    - Cryptographic hash functions

P2P foundation

The Foundation for Peer to Peer Alternatives

Main My Page Members Videos Forum Groups Blogs Chat

All Discussions My Discussions

+ Add

Welcome to
P2P Foundation

Sign Up
or Sign In

Bitcoin open source implementation of P2P currency

Posted by Satoshi Nakamoto on February 11, 2009 at 22:27

View Discussions

I've developed a new open source P2P e-cash system called Bitcoin. It's completely decentralized, with no central server or trusted parties, because everything is based on crypto proof instead of trust. Give it a try, or take a look at the screenshots and design paper:

Download Bitcoin v0.1 at http://www.bitcoin.org

The root problem with conventional currency is all the trust that's required to make it work. The central bank must be trusted not to debase the currency, but the history of fiat currencies is full of breaches of that trust. Banks must be trusted to hold our money and transfer it electronically, but they lend it out in waves of credit bubbles with barely a fraction in reserve. We have to trust them with our privacy, trust them not to let identity thieves drain our accounts. Their massive overhead costs make micropayments impossible.

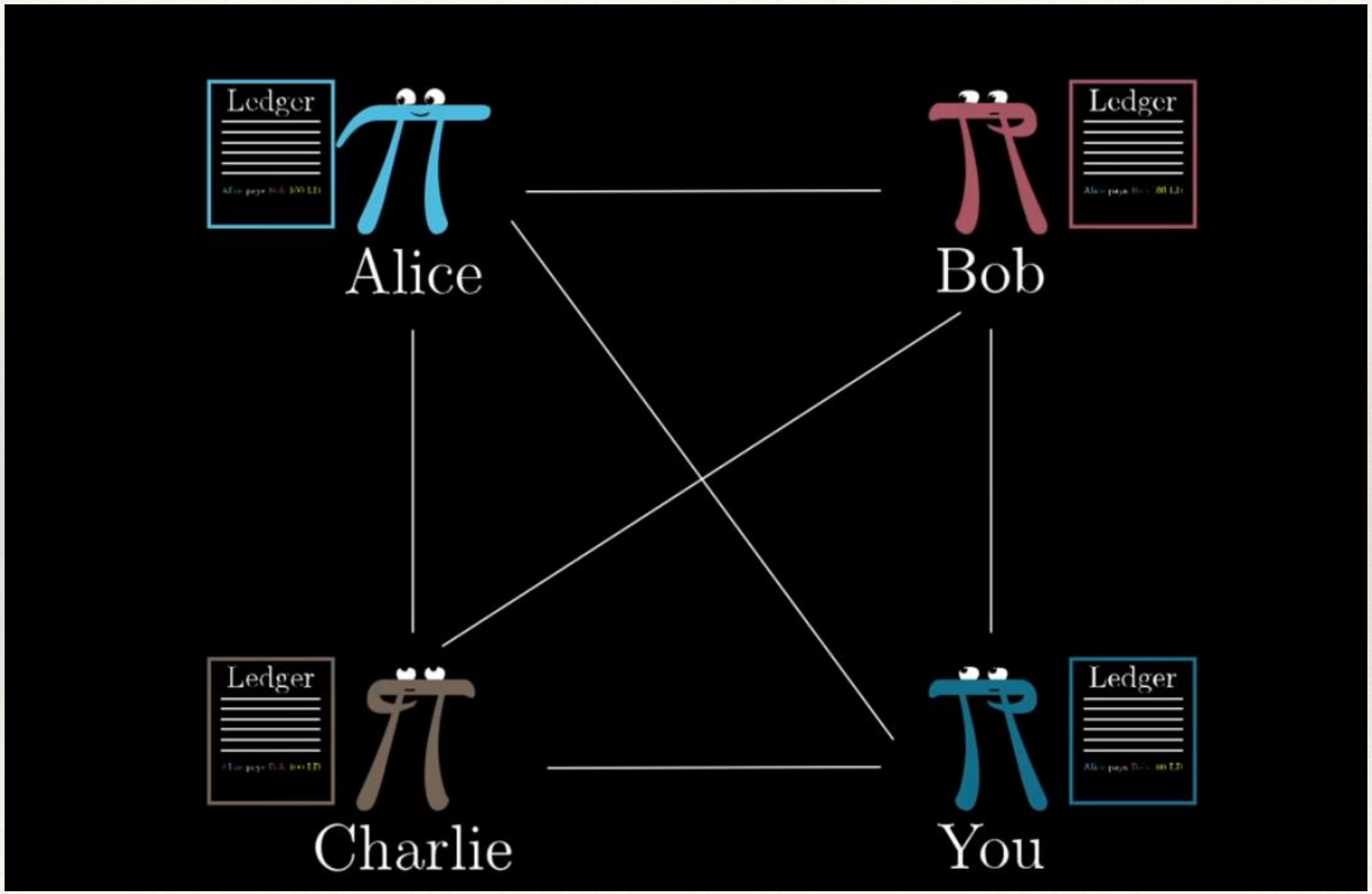
A generation ago, multi-user time-sharing computer systems had a similar problem. Before strong encryption, users had to rely on password protection to secure their files, placing trust in the system administrator to keep their information private. Privacy could always be overridden by the admin based on his judgment call weighing the principle of privacy against other concerns, or at the behest of his superiors. Then strong encryption became available to the masses, and trust was no longer required. Data could be secured in a way that was physically impossible for others to access, no matter for what reason, no matter how good the excuse, no matter what.

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

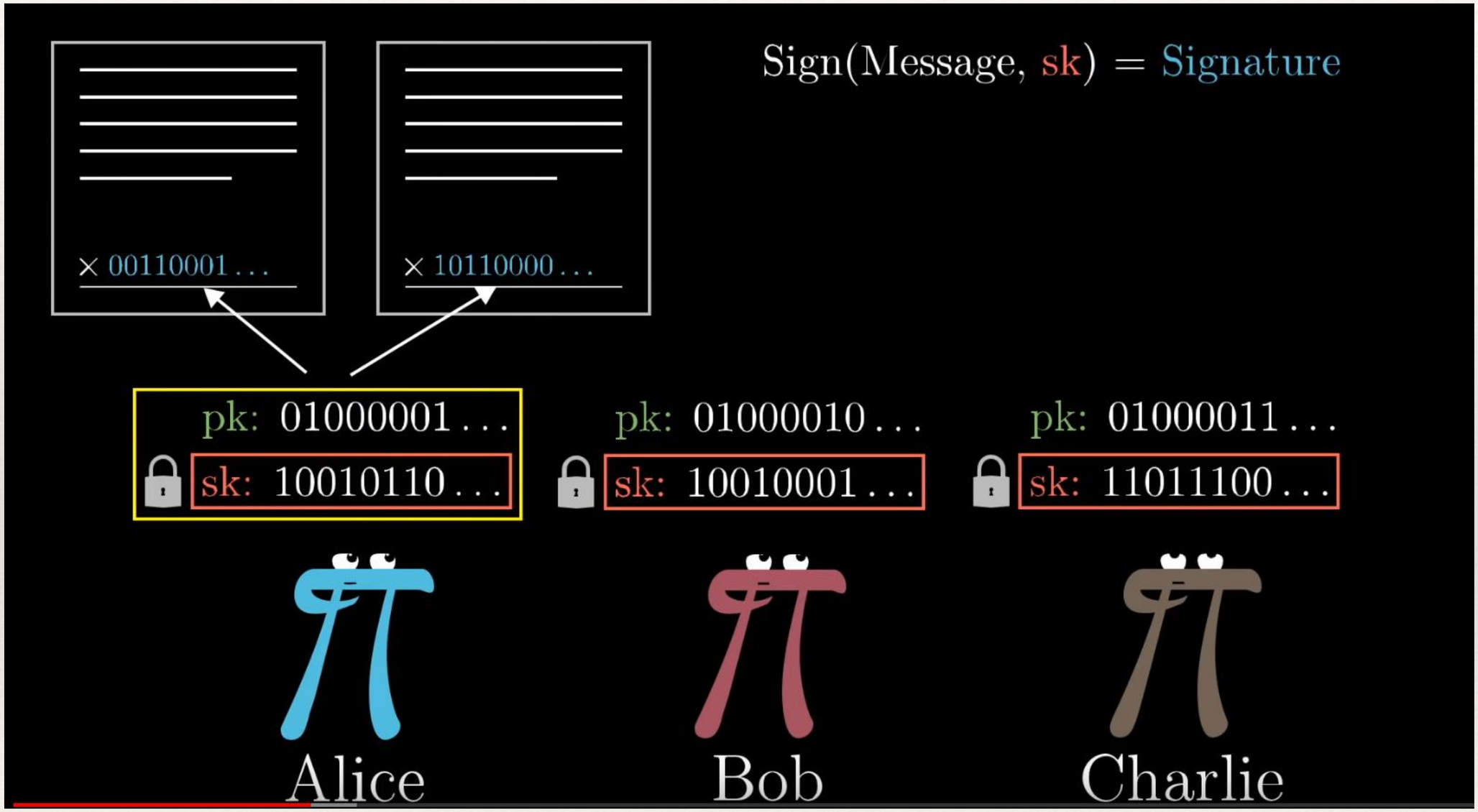
Peer-to-Peer Ledger



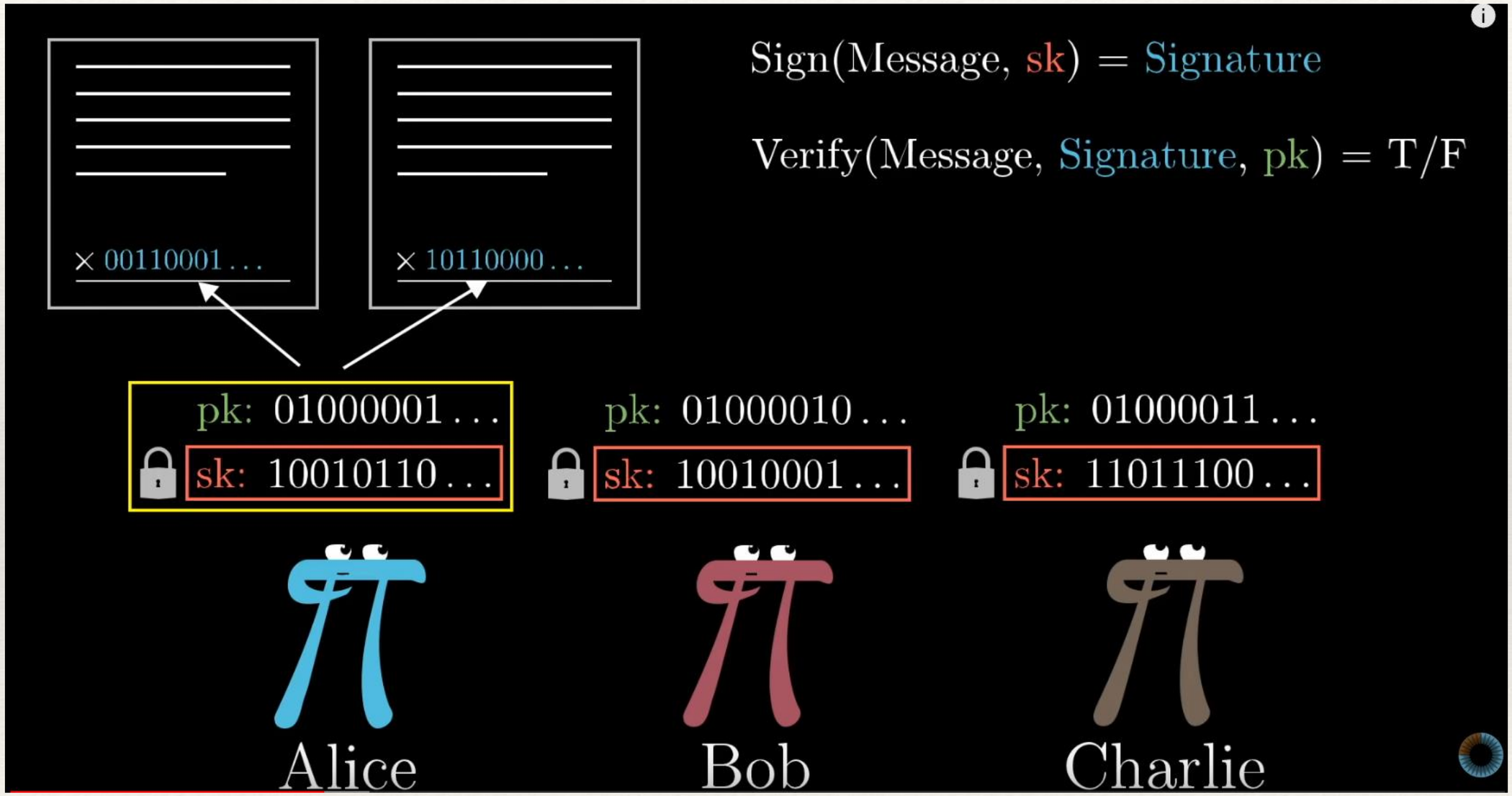
Data integrity and data agreement

- Data integrity: Data is legit.
- Data agreement: among all nodes and across time.
  - This is the “double spending” problem and is solved by a consensus protocol.

Data integrity: digital signatures



Verifying


Sign(Message, sk) = Signature
Verify(Message, Signature, pk) = T/F
× 00110001 . . .
× 10110000 . . .
pk: 01000001 . . .
sk: 10010110 . . .
pk: 01000010 . . .
sk: 10010001 . . .
pk: 01000011 . . .
sk: 11011100 . . .
Alice
Bob
Charlie

Digital Signatures

Definition 13.1. *A* ***signature scheme*** $\mathcal{S} = (G, S, V)$ *is a triple of efficient algorithms,* G, S *and* V*, where* G *is called a* ***key generation algorithm****,* S *is called a* ***signing algorithm****, and* V *is called a* ***verification algorithm****. Algorithm* S *is used to generate signatures and algorithm* V *is used to verify signatures.*

- G *is a probabilistic algorithm that takes no input. It outputs a pair* (pk, sk)*, where* sk *is called a secret* ***signing key*** *and* pk *is called a public* ***verification key****.*
- S *is a probabilistic algorithm that is invoked as* $\sigma \xleftarrow{R} S(sk, m)$*, where* sk *is a secret key (as output by* G*) and* m *is a message. The algorithm outputs a* ***signature*** σ*.*
- V *is a deterministic algorithm invoked as* $V(pk, m, \sigma)$*. It outputs either* accept *or* reject*.*
- *We require that a signature generated by* S *is always accepted by* V*. That is, for all* (pk, sk) *output by* G *and all messages* m*, we have*

$$\Pr[V(pk,\ m,\ S(sk,\ m)\ ) = \text{accept}] = 1.$$

Boneh & Shroup 2023

Forgery Game

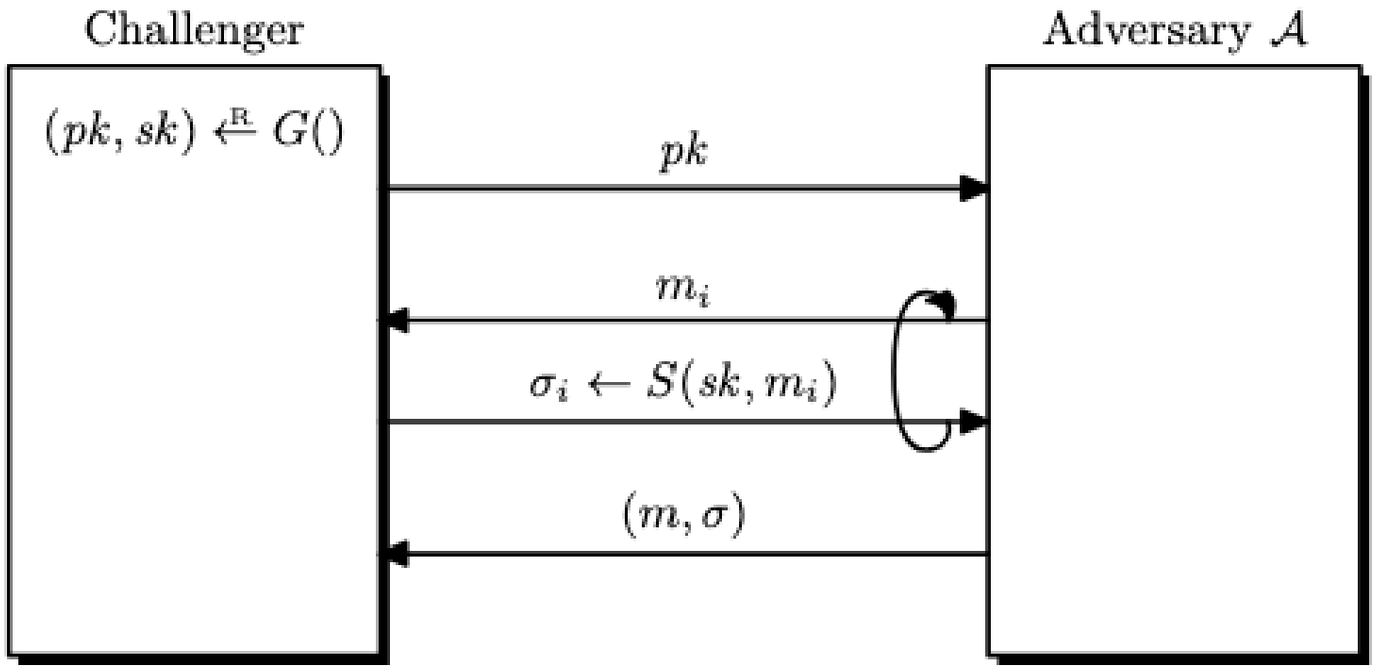


Figure 13.1: Signature attack game (Attack Game 13.1)

No (computationally bounded) attacker can forge a signature on any chosen message.

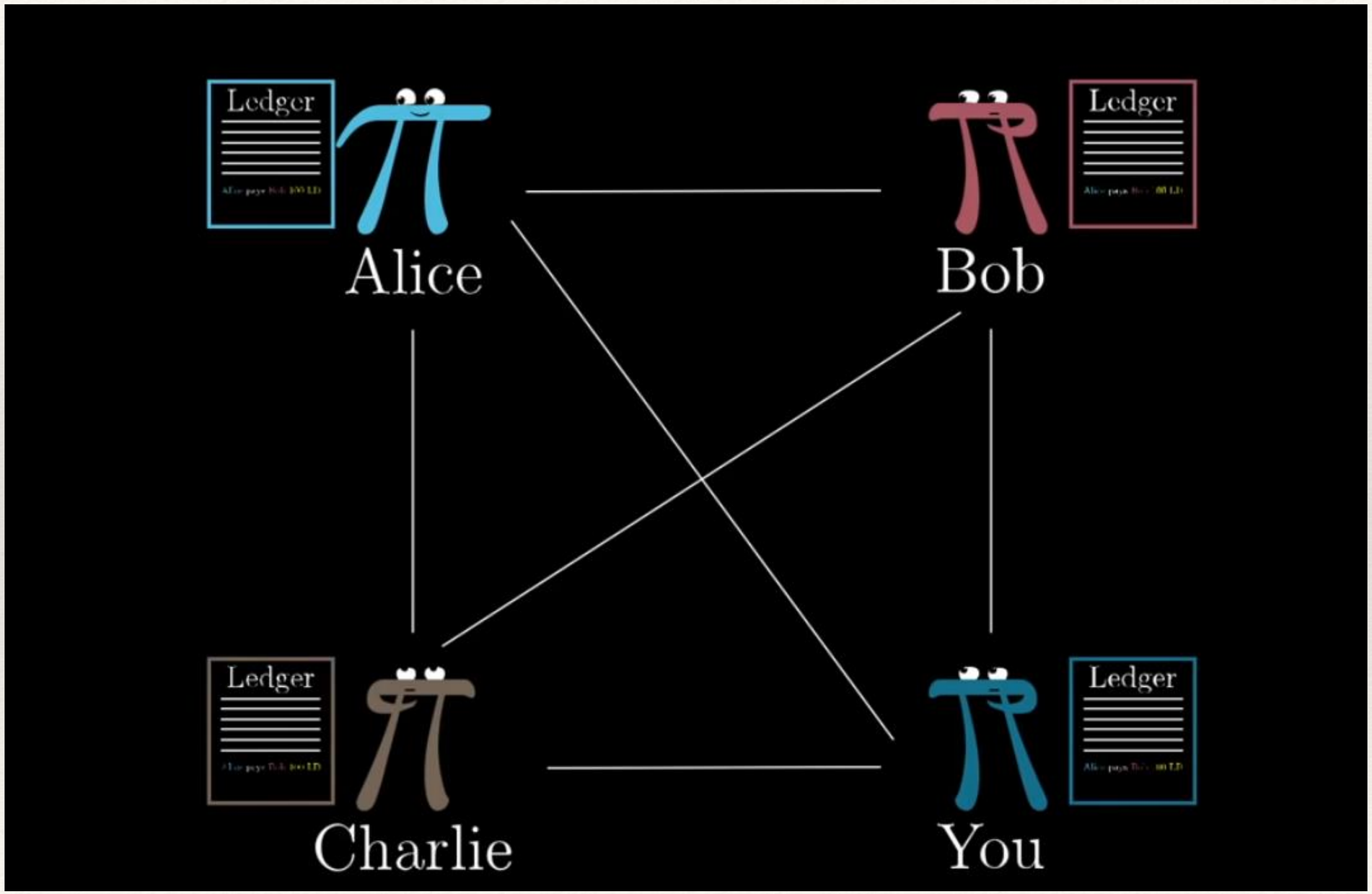
UTXO model

pk1 ----→ pk2---→ pk3 ---→ pk4 --→ ……….

Double spend

- Paul sends two correctly signed messages:
  - Paul pays John 1 bitcoin.
  - Paul pays Peter 1 bitcoin

Inconsistent Ledgers

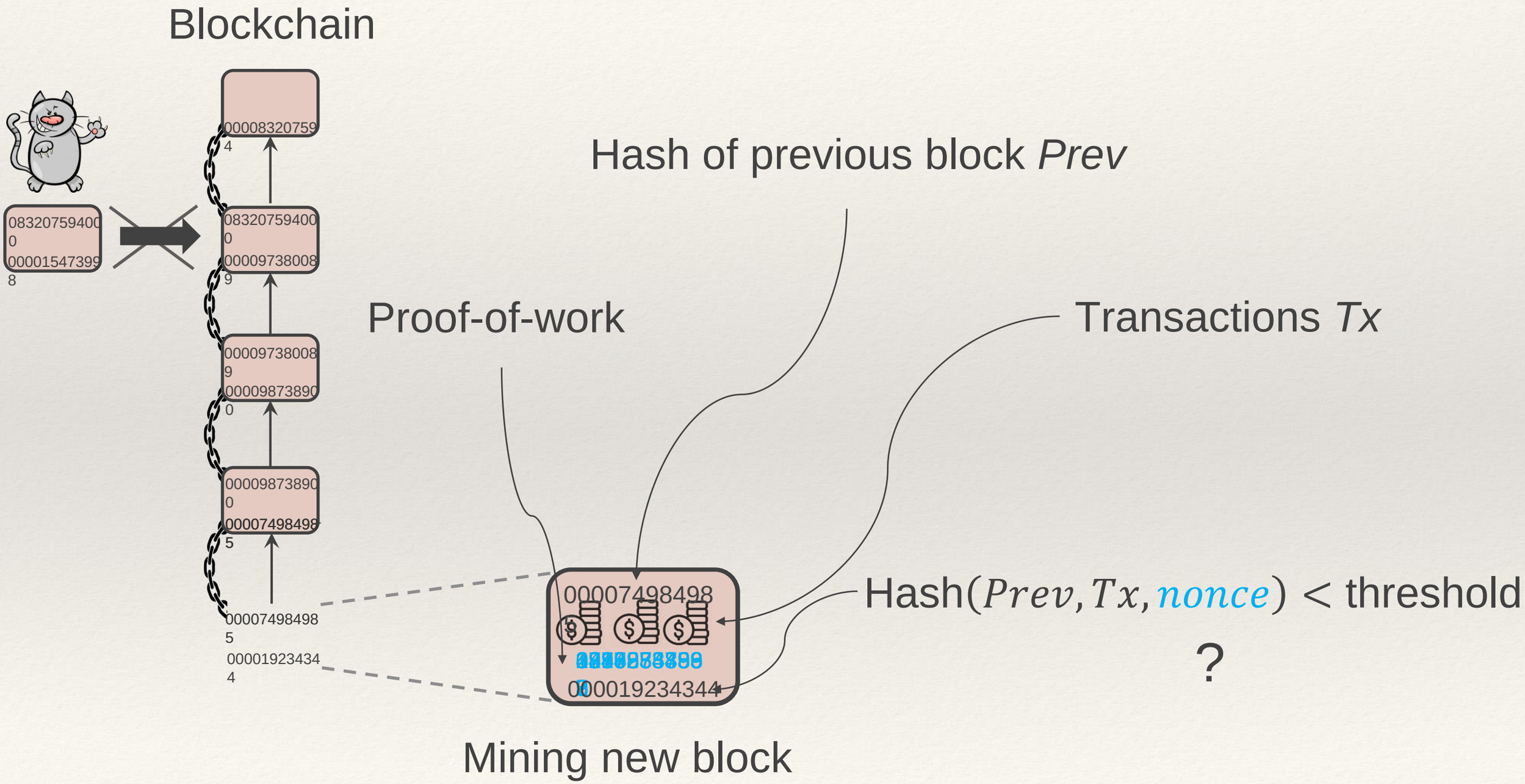


Challenges

- Synchronization (decentralization)
- Sybil attacks (permissionless)

Data agreement: Proof-of-work longest chain protocol

Proof-of-Work



Block verification

- Data integrity: each transaction spends an UTXO with a valid signature.
- Data agreement: proof-of-work is valid.