Lecture 1:
Introduction and Overview

(Lecture Series on Foundations of Blockchains)

Focus of Lecture Series

Intended audience: has some amount of computer science training (possibly self-taught).

Focus: the science + tech of blockchains & applications built on top of them.

- not about: hype, investing, entrepreneurship, nitty-gritty engineering details
- instead: fundamental principles of blockchain design + analysis

Warning: not standardized, in flux

Overview of Lecture Series

"Blockchain stack":

application layer (stuff built on top)

layer 2 (scaling layer)

layer 1 (consensus layer) (+ a compute layer)

layer 0 (= the Internet)

Outline:

[will skip layer 0]

- 1st 60% of lectures about layer 1
  - classical consensus protocols, Nakamoto/longest-chain consensus, Sybil-resistance (PoW/PoS), difficulty adjustment, selfish mining, deep dives on Bitcoin, Ethereum
- next 20% on layer 2
  - scaling Bitcoin via payment channels, Lightning network
  - scaling Ethereum via "rollups"
  - next-generation layer 1 protocols
- last 20% on application layer
  - DeFi primitives (oracles, stablecoins), DeFi apps (borrowing, lending, trading via AMMs)

Maybe - MEV, DAOs

Comments

1. a new computing paradigm
2. not about digital money
3. principles over protocols

Digital Signature Schemes

Consensus: (informally) keep multiple machines ("nodes") in sync, even in the face of an unreliable network, malicious attacks.

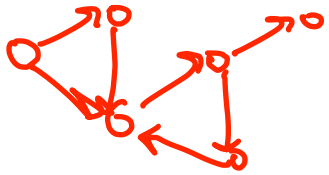
Permanent assumptions: (i) the Internet exists (ii) cryptography exists.

Definition of a DSS: (defined via 3 efficient algorithms)

1. Key generation algorithm (maps random seed $r \mapsto$ (pk, sk) pair)
2. signing algorithm (maps msg + sk $\longmapsto$ msg + sig)
3. verification algorithm (maps msg + sig + pk $\longmapsto$ "yes" / "no")

Assumption: ("ideal" signatures) don't know sk $\Rightarrow$ impossible to generate valid msg + sig.

By default: nodes sign all messages that they send.



The SMR Problem

(SMR = "state machine replication")

(1980s)

- clients submit transactions (txs) to nodes
- each node maintains a local append-only data structure (ordered sequence of txs) (a "history")
- want to keep all nodes in sync, identical local histories

Definition: a protocol = code to be run by each node

(local computations, receive msgs from other nodes + clients, send msgs to other nodes)

Goals: (1) **Consistency:** all nodes agree on history (i.e., same tx sequence)

(2) **liveness:** every valid tx submitted eventually added to history

Next: the synchronous model and the Dolev-Strong protocol!