

# BitCoin, Blockchain, and Peer-to-Peer Ledgers

Some ideas in how to scale up log management



# Process State

- A stateful process has time varying properties.
- Process state is the snapshot of the system at a given point in time.
- We usually think of these as finite or discrete.
- Example: CREATE, UPDATE, and DELETE operations on a data base change the state of the database.

## Logging State Changes

- A state change is the non-zero difference between  $\text{State}(t_m)$  and  $\text{State}(t_n)$  where  $n > m$ .
- We frequently need to record state changes in logs or ledgers.



# Distributed System State



A system is a collection of cooperating processes

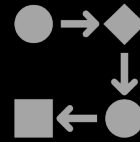


The state of the system is the state of its constituent processes

# Distributed State Coordination



Sometimes, we just need to know the state of the whole system



Sometimes, we need at least some of the processes that make up the system to have the same process state



Distributed systems need to consider tradeoffs in scaling, fault tolerance, consistency, ...

# RAFT



Has a strong leader



Provides fault tolerance and consistency with limited scaling



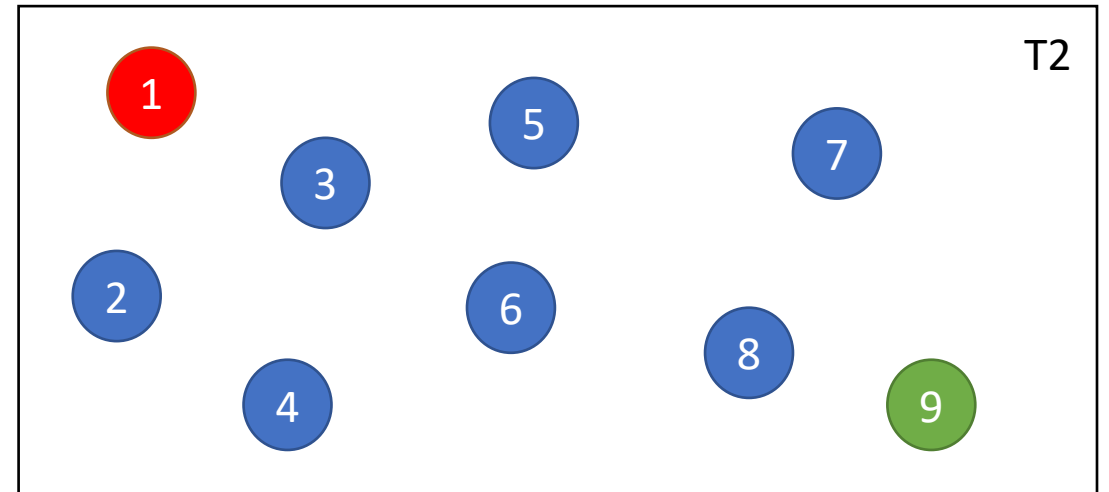
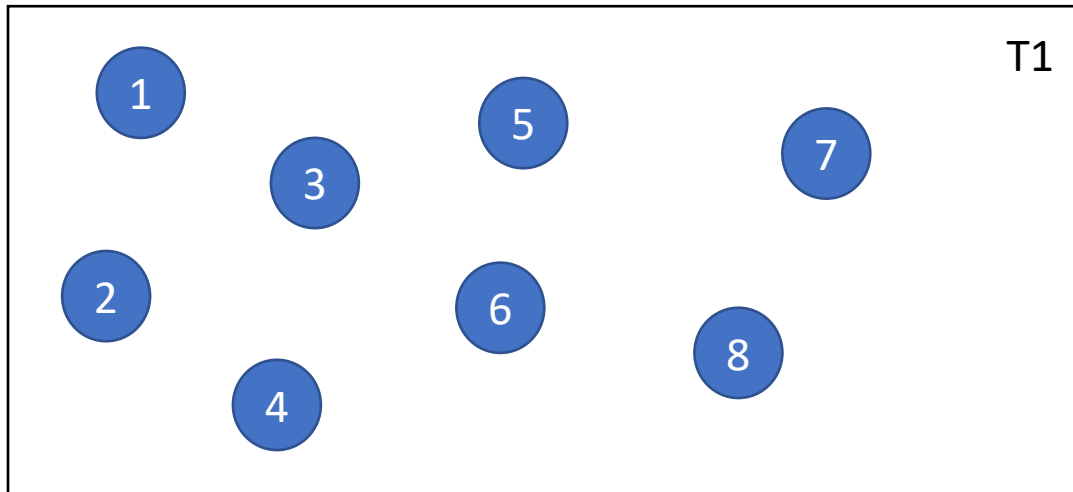
Can coordinate process states in a system that needs consensus consistency.



Example: transaction logs in a database

# SWIM

- Peer-to-peer
- Eventually consistent
- Good for tracking state changes that don't need an audit trail.
- Ex: group membership changes (maybe)



Can We...



Find a process that scales like SWIM, but...



Can create consistent audit trail logs?



Maybe...



<https://bitcoin.org/en/bitcoin-paper>

# Bitcoin Overview



Bitcoin uses keys and hashes to create a globally scalable monetary system.



In other words, it is a peer-to-peer transaction system.



So it needs to both scale and be consistent.



How?



Start with security basics



# Public Key Infrastructure (PKI)

- **Private Keys:** Cryptographically sign messages
- Send the signature along with the message
- Recipients use the **public key** to verify that the message came from the signer

# PKI Limitations



PKI works as long as the private keys are kept private



Public keys of compromised private keys need to be revoked



This is another well known distributed systems problem

# SSH Private Key Example

```
Marlons-MacBook-Pro:~ marpierc$ more junk_keyfile
```

```
-----BEGIN OPENSSH PRIVATE KEY-----
```

```
b3BlbnNzaC1rZXktdjEAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABFwAAAAAdzc2gtcn
NhAAAAAwEAAQAAAEAr+9uzr5hyuHx+IBN+XwhVGqcRPTBMgz9hioMu+f53WFqP3EuTGZv
uEW1XAV5Nm2Acvr0DVXLD7rqz6emcmCwo0TEesX7IAJAAWlKWhmneqdLSQ5ZY7LSlk3oes
sM+LW5VVIUCs1oZSd/attsd3HqFoflpZHIcmg9q0IIQy/GTAhGX1J90aVfep2KQxYEk0TV
QXxFMdQ06yEcyRQC8R2HuamioEevrXs0XKLwQdSgdYRHhEmMsu0TpmD6Kw5K8ID/NE07im
L4j8u4hw9XkqA9o3YAEkiTWN2IDclJP0qpsS2mU2gwJyJ1PCIKsLHTUiW5Eb8ElGsD70c
81awBqCJiQAAA+DZHP9l2Rz/ZQAAAAAdzc2gtcnNhAAABAQCv7270vmHK4fH4gE35fCFUap
xE9MEyDP2GKgy75/ndYWo/cS5MZm+4RbVcBXk2bYBy+vQNveUPuurPp6ZyYLCg5MR6xfsg
AkABaUpaGad6p0tJDlljstKWTEh6yWz4tblVUHQKzWhlJ39q22x3ceowh+wlkciZyD2o4g
hDL8ZMCEZfUn05pV96nYpDFgSQ5NVBfEUx1A7rIRzJFALxHYe5qaKgR6+tew5covBB1KB1
hEeESYyy7R0mZ3orDkrwgP80TTuKYviPy7iHD1eSoD2jdgASSJNYE3YgNyUk86qmxLaZTa
DAnInU8IggYUdNSJbkRvwSUawPs5zzVrAGoImJAAAAAwEAAQAAAEArRImuS7D20dIN6NQ
EXsw9nAh5hu36dqpK8/N0x0y0zq/YEWgu/uRL38zL6Cyyv4RfAqvBmdW/JBt6XUM4juHxd
8GAZi9H5HXEQxY3iWagagNAYMiIFeLndxqNFGHIyrxdKNXoADND6U5TQ8ptp7THvL00FmH
MvCu53HjmuRmd+eSddZ0Lviuy0nr+wV2obePAbs6dN6p17RGzJATAaevC3rNzLZk6AdTp
+VpCkgiR5CxBEQeFPAvzG7fMoexAUIB2MMPVIil8QC9kvIpofiHKUwg0UeIDUuR305L3k0
X7W5vgV/9zLUeHgYm+aBeLsvYgnJcm4f2yUgJhx2eIAAAQAAAEApL2RLGksFYPwWyP9Bb
L4017/ezxCUvL9YSa+0fCs6VbTpeP3cTBbqWcusv1rIzmSNaJ92mzJ4s5vIQ8WzB0lIxIj
35BN3XfT5FNX+FjEYnMAu9joUZxyzfcLYpLRsJqR0+P6Fpqsk1IsoelmQ0vwvbsxOpX7j+
DPY8S6U8YFcTUAACBANU0h4NxEQr3lGTMr4KVSviggXr+33hd/UT1VPQJajUG1LRzoaE6
FV8KJAwJ3yKe1i8fwyp2H3Iy04bmV6mxB0AB4SUpmY5MhFvI4m2wzCDrAY1xEBNamZfjPx
5fqGfRT61+lxZ3yhnpXi3EwVPBB+gUlxeUfd1PfpYq7pK7mzABAAAQDTP82Mef45s2vp
dZm7on44CAu6i6znVy8L+g+7Qnu3lBpRuAPSFx0QbyNEbNLa6gHG9yEFn8N8IdCH6L28WC
rERSDnr/7e8T0jcUPosIxfWV83ERzNo3vT6Uwc0fC+D9H2m/Nz8RZ03qX5BIja8Ium32gu
ZVpSl/BXFJnHh0PZiQAAACJtYXJwaWVY0BNYXJsb25zLU1hY0Jvb2stUHJvLmxvY2FsAQ
```

```
IDBAUGBw==
```

```
-----END OPENSSH PRIVATE KEY-----
```

# SSH Public Key Example

```
Marlons-MacBook-Pro:~ marpierc$ more junk_keyfile.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQBAQCv7270vmHK4fH4gE35fCFUapxE9MEyDP2GKgy75/ndYWo/cS5MZm+4Rb
VcBXk2bYBy+vQNVeUPuurPp6ZyYLCg5MR6xfsgAkABaUpaGad6p0tJD1ljstKWTeh6y wz4tblVUhQKzWhlJ39q
22x3ceoWh+WlkciZyD2o4ghDL8ZMCEZfUn05pV96nYpDFgSQ5NVBfEUx1A7rIRzJFALxHYe5qaKgR6+tew5cov
BB1KB1hEeESYyy7R0mZ3orDkrwgP80TTuKYviPy7iHD1eSoD2jdgASSJNYE3YgNyUk86qmxLaZTaDAnInU8Igq
yUdNSJbkRvwSUawPs5zzVrAGoImJ marpierc@Marlons-MacBook-Pro.local
```

# OpenSSL Command Line Tools

- Create key pairs
  - openssl req -nodes -x509 -sha256 -newkey rsa:4096 -keyout "**meps.key**" -out "**meps.crt**" -days 365 -subj "/C=US/ST=Indiana/L=Bloomington/O=IU/OU=PTI/CN=MEPs Sign Key"
- Sign files
  - openssl dgst -sha256 -sign "**meps.key**" -out sign.txt.sha256 sign.txt
- Verify signatures
  - openssl dgst -sha256 -verify <(openssl x509 -in "**meps.crt**" -pubkey -noout) -signature sign.txt.sha256 sign.txt

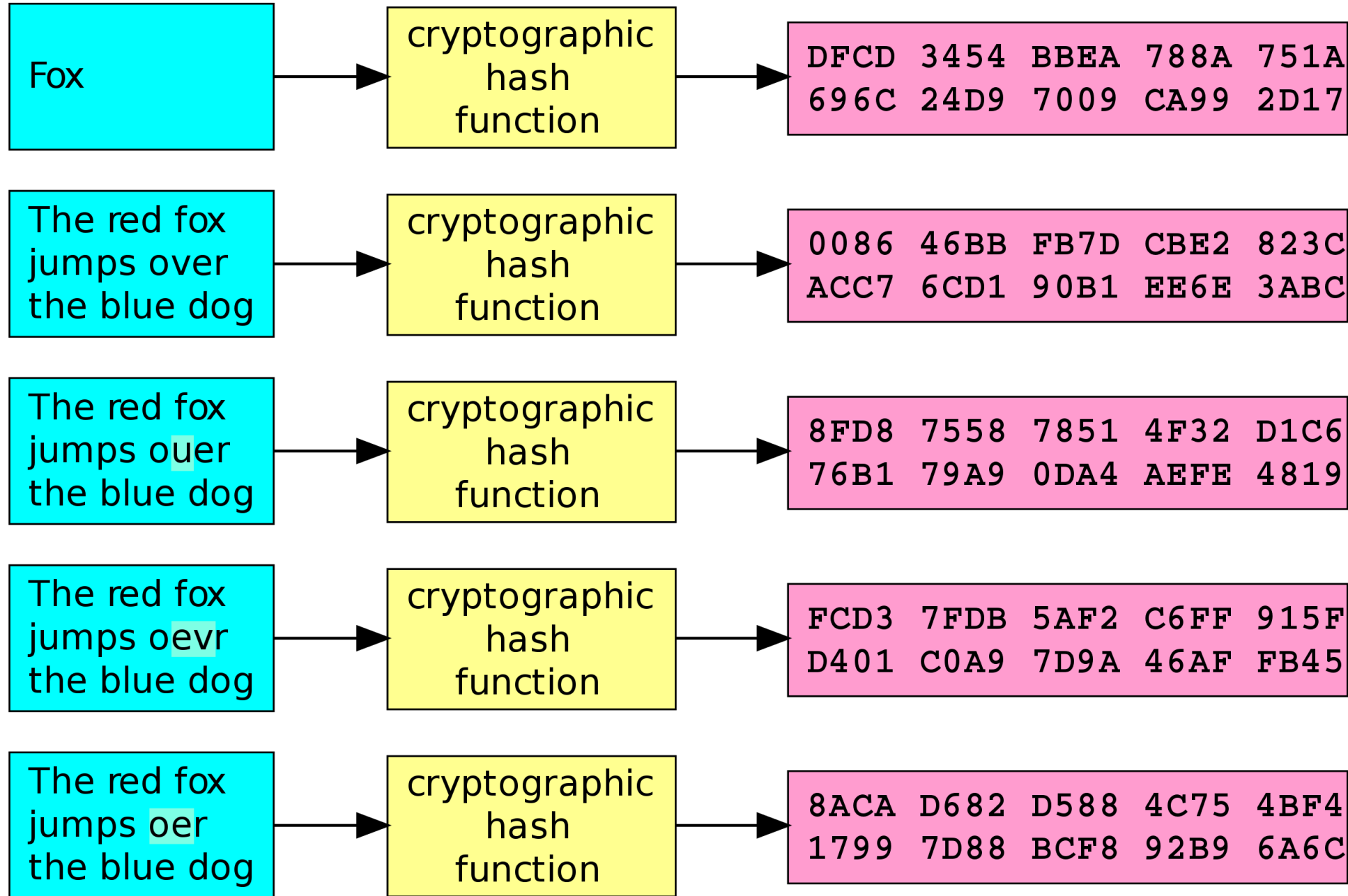
# Cryptographic Hashing

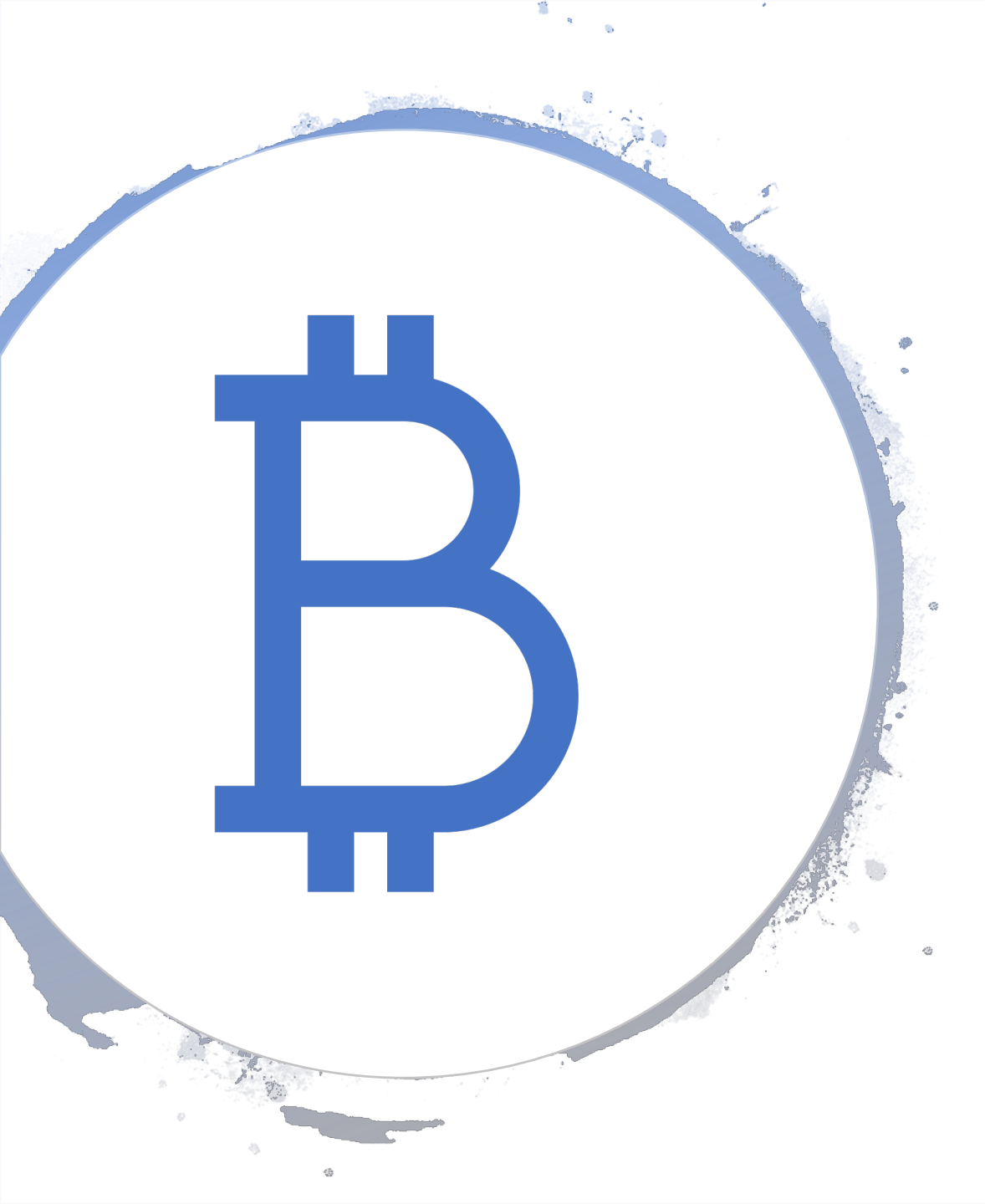
- A *hash* algorithm is a fast mathematical function that generates a unique, hard-to-guess numerical value from a given input
- Two messages differing by a single character generate completely different hashes.
- Hashes are not reversible: given a value, you can't easily guess the original input
- Hashes are a simple way to verify that data hasn't been corrupted or modified during transmission
- Hashing is often combined with signing



# Input

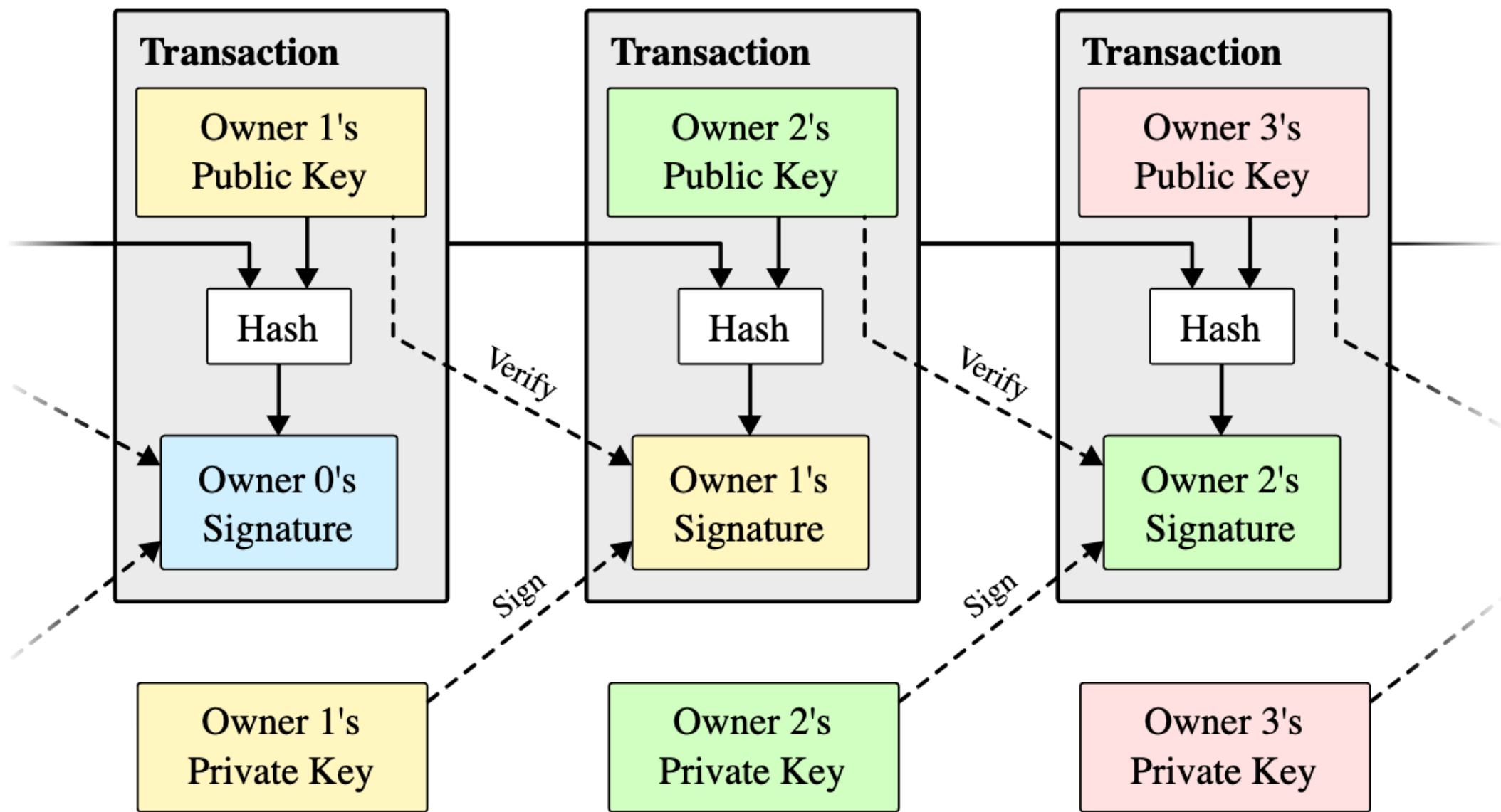
# Digest





# Bitcoin Basic Transactions

- Assume for the moment that bitcoins exist
- Marlon wants to buy a pizza from Suresh and pay in 1 bitcoin.
- Marlon transfers the bitcoin to Suresh in exchange for the pizza.



# Steps in a Transaction



Only Marlon (Owner 1) can spend the coin because his public key is embedded in the last block.



Marlon transfers the coin to Suresh (Owner 2) by digitally signing a hash of the previous transaction and Suresh's public key



Marlon adds these to the end of the coin.



Suresh can verify the signatures to verify the chain of transactions.



Compare with BFT Raft's logs

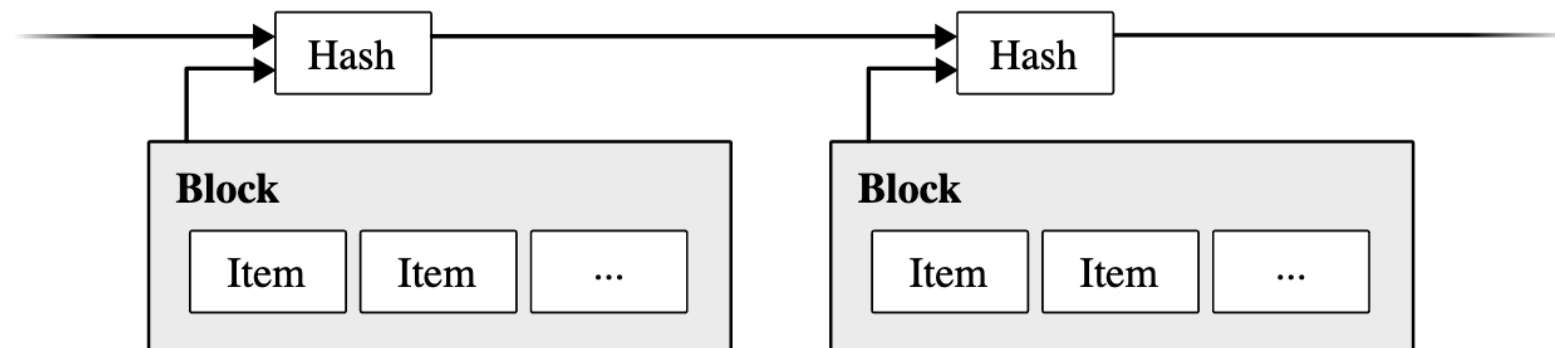


But Can Suresh  
Trust Marlon?

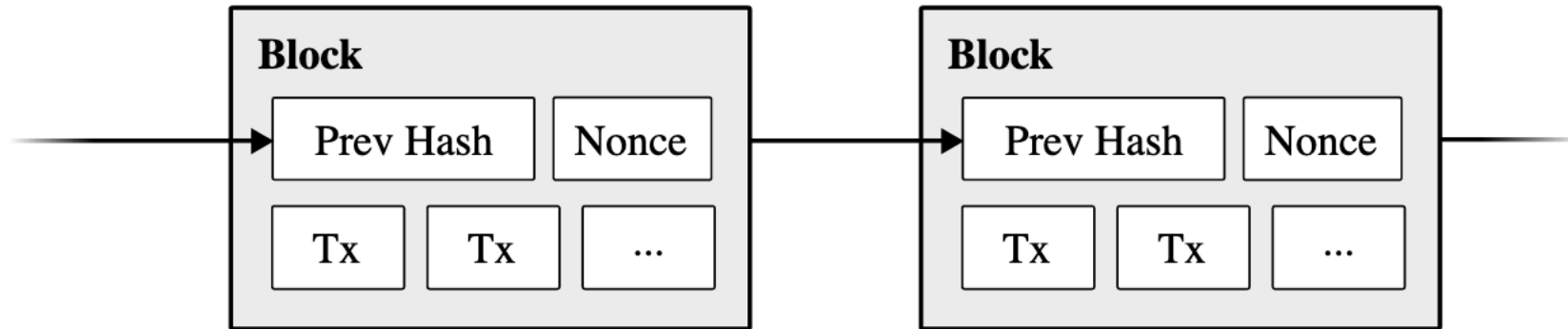
- What if Marlon also used the same bitcoin to pay Dimuthu for tacos at the same time?
- This is called double-spending.

# Bitcoin's Solution, Part 1: Timestamping and Ledgers

- We publish this information broadly to public ledgers
  - We include a time stamp
- The timestamp proves that the data must have existed at the time, obviously, in order to get into the hash.
- Each timestamp includes the previous timestamp in its hash, forming a chain, with each additional timestamp reinforcing the ones before it.



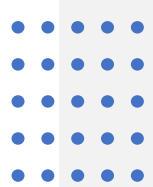
# Bitcoin's Solution, Part 2: Proof of Work



X-Hashcash: 1:52:380119:calvin@comics.net:::9B760005E92F0DAE



00000000000000756af69e2ffbdb930261873cd71



# A 1 CPU, 1 Vote Meritocracy

You can vary the computing requirements by changing the required number of 0's in the hash.

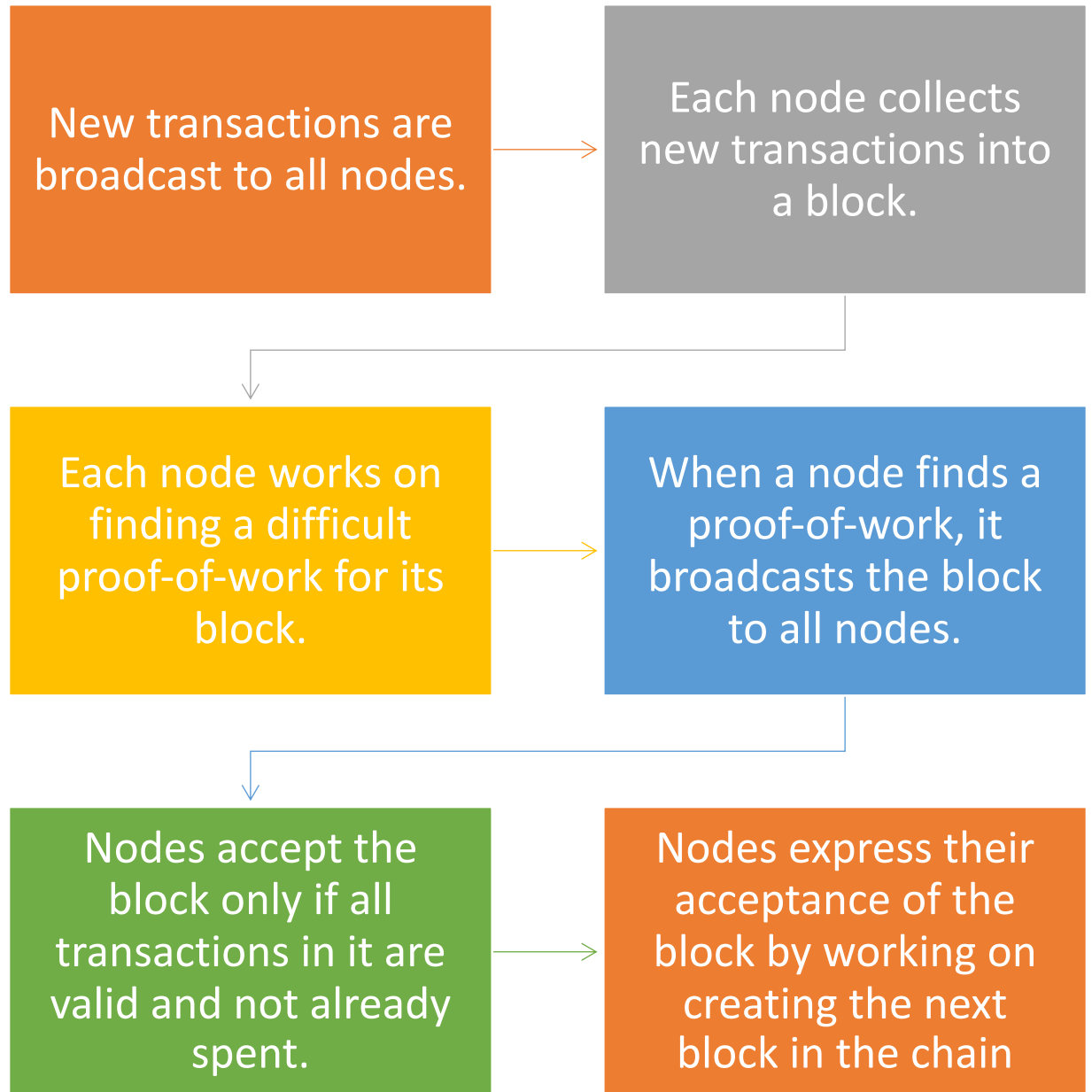
Email clients can do this in a second on average

Bitcoin's required 00000...000 prefix is set to take about 10 minutes on average.

Bitcoin proof-of-work is done by the ledgers in the network



# Steps in a Bitcoin Network



# Trust the Longest Chain

Nodes always consider the longest chain to be the correct one and will keep working on extending it.

If two nodes broadcast different versions of the next block simultaneously, some nodes may receive one or the other first.

In that case, they work on the first one they received, but save the other branch in case it becomes longer.

The tie will be broken when the next proof-of-work is found and one branch becomes longer;

The nodes that were working on the other branch will then switch to the longer one.

# Global Crime Fighting



Variation in computing time foils Marlon's attempt at double-spending.

One transaction will eventually get chosen



But it also works globally: as chains grow longer, the proof-of-work burden to subvert a chain becomes very large



As long as most participants are honest, dishonest network nodes will fall exponentially behind.

# Bitcoin, Blockchain, and Scalable Registries



Bitcoin shows a way to have scalable, peer-to-peer transaction ledgers (blockchains) that can also be kept consistent.



Bitcoin assumes competing interests and uses incentives and to encourage honesty: emergent system properties



Is there a case for blockchain registries in Service Meshes and Microservices?