BitCoin, Blockchain, and Peer-to-Peer Ledgers

Some ideas in how to scale up log management

A Thought Exercise

- You are the architect for a new distributed system for a national scale retail corporation.
- It is a requirement that you have a “fog” architecture,
  - Some servers are deployed on- or near-premises at the retail locations
  - Other services are deployed on a commercial cloud vendor

How Would You Build This System?

- What are the challenges for building this type of system?
- What are the advantages and disadvantages of the following?
  - RabbitMQ-based messaging for edge-to-cloud communication
  - Kafka-based log systems for edge-to-cloud communication
  - Wide area Kubernetes deployments
  - Service meshes for edge-to-cloud communication

BitCoin, Blockchain, and Peer-to-Peer Ledgers

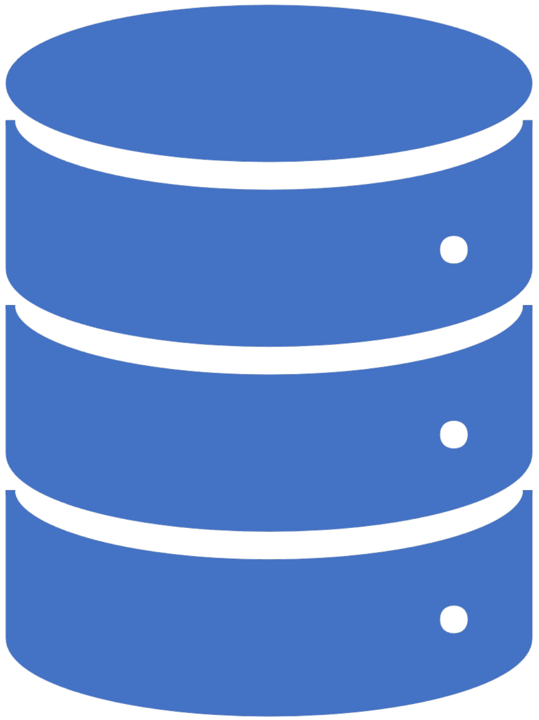
Some ideas in how to scale up log management

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 state changes as 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 State(t_m) and 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

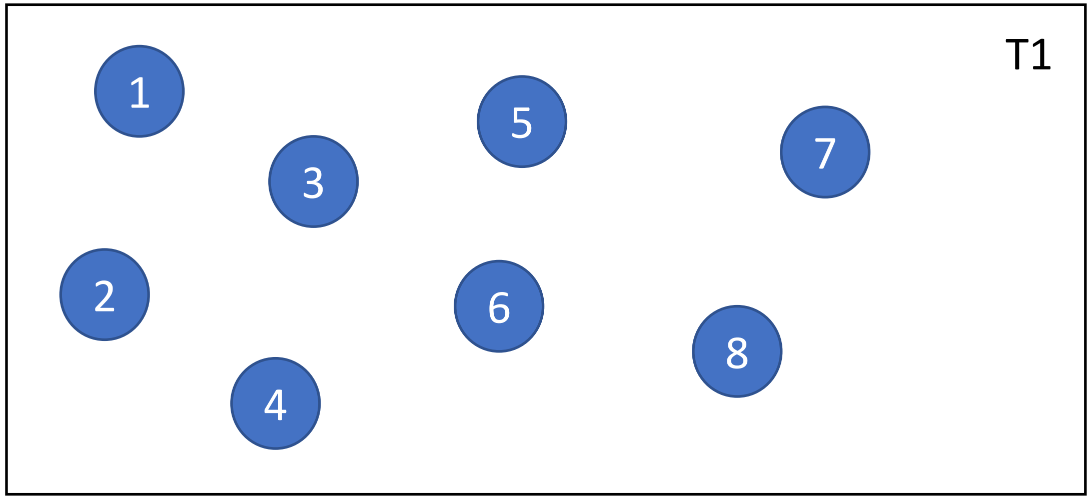
The state of the system is the state of its constituent processes

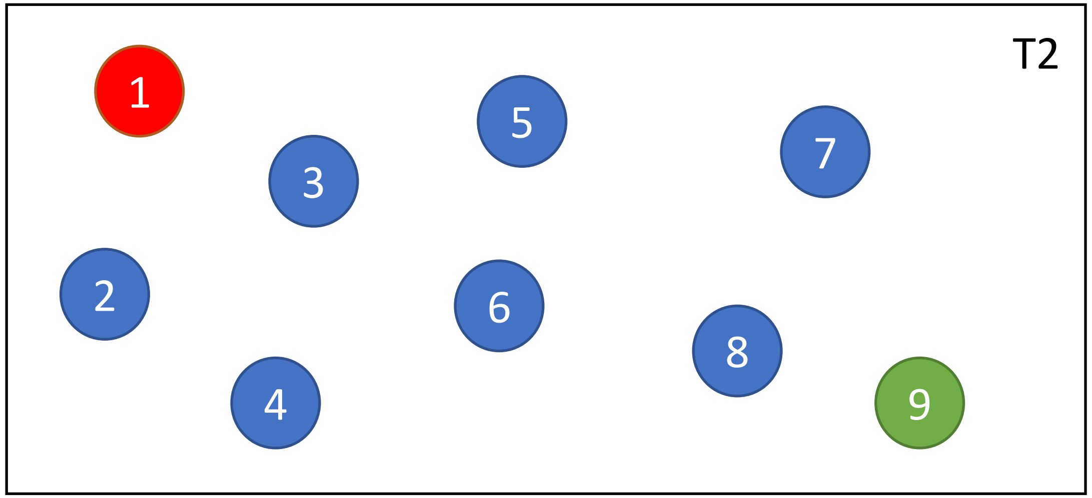
RAFT

- Has a strong leader
- Provides fault tolerant and consistent log management 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...

A state management approach that scales like SWIM, but...

Can create consistent audit trail logs like RAFT?

Maybe...

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

Image by Bitboy - Bitcoin forums, Public Domain, https://commons.wikimedia.org/w/index.php?curid=15411063

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.
- It therefore must 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-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABFwAAAAdzc2gtcn
NhAAAAAwEAAQAAAQEAr+9uzr5hyuHx+IBN+XwhVGqcRPTBMgz9hioMu+f53WFqP3EuTGZv
uEW1XAV5Nm2Acvr0DVXlD7rqz6emcmCwoOTEesX7IAJAAWlKWhmneqdLSQ5ZY7LSlk3oes
sM+LW5VVIUCs1oZSd/attsd3HqFoflpZHImcg9qOIIQy/GTAhGX1J9OaVfep2KQxYEkOTV
QXxFMdQO6yEcyRQC8R2HuamioEevrXsOXKLwQdSgdYRHhEmMsu0Tpmd6Kw5K8ID/NE07im
L4j8u4hw9XkqA9o3YAEkiTWBN2IDclJPOqpsS2mU2gwJyJ1PCIKslHTUiW5Eb8ElGsD7Oc
81awBqCJiQAAA+DZHP9l2Rz/ZQAAAAdzc2gtcnNhAAABAQCv727OvmHK4fH4gE35fCFUap
xE9MEyDP2GKgy75/ndYWo/cS5MZm+4RbVcBXk2bYBy+vQNVeUPuurPp6ZyYLCg5MR6xfsg
AkABaUpaGad6p0tJDlljstKWTeh6ywz4tblVUhQKzWhlJ39q22x3ceoWh+WlkciZyD2o4g
hDL8ZMCEZfUn05pV96nYpDFgSQ5NVBfEUx1A7rIRzJFALxHYe5qaKgR6+tew5covBB1KB1
hEeESYyy7ROmZ3orDkrwgP80TTuKYviPy7iHD1eSoD2jdgASSJNYE3YgNyUk86qmxLaZTa
DAnInU8IgqyUdNSJbkRvwSUawPs5zzVrAGoImJAAAAAwEAAQAAAQEArRImuS7D2OdIN6NQ
EXsw9nAh5hu36dqpk8/N0xOy0zq/YEWgu/uRL38zl6Cyyv4RfAqvBmdW/JBt6XUM4juHxd
8GAZi9H5HXEQxY3iWagagNAYMiIFeLndxqNFGHIyrxdKNXoADND6U5TQ8ptp7THvL00FmH
MvCu53HjmuRmd+eSddZ0Lviuy0nr+wV2obePAbs6dN6p17RGzJATAtaevC3rNzlZk6AdTp
+VpCkgiR5CxBEQeFPAvzG7fMoexAUIB2MMPVIil8QC9kvIpofiHKUwgOUeIDUuR305L3kO
X7W5vgV/9zLUeHgYm+aBeLsvYgnJcm4f2yUgJhx2eIAAAQAAAIEApL2RLGksFYPwWyP9Bb
L4O17/ezxCUvl9YSa+0fCs6VbTpeP3cTBbqWcusv1rIzmSNaJ92mzJ4s5vIQ8WzBOlIxIj
35BN3XfT5FNX+FjEYnMAu9joUZXyzfcLYpLRsJqRO+P6Fpqsk1IsoelmQOvwvbsxOpX7j+
DPY8S6U8YFcTUAAACBANU0h4NxEQr3lGTMr4KVSViggXr+33hd/UT1VPQJajUG1LRzoaE6
FV8KJAwJ3yKe1i8fwyp2H3IyO4bmv6mxB0AB4SUpmY5MhFvI4m2wzCDrAY1xEBNamZfjPx
5fqGfRT61+lxZ3yhnpXi3EwVPBB+gUlxeUfd1Pfpyq7pK7mzABAAAAgQDTP82Mef45s2vp
dZm7on44CAu6i6znVy8L+g+7Qnu3lBpRuAPSFxOQbyNEbNLa6gHG9yEFn8N8IdCH6L28WC
rERSDnr/7e8T0jcUPosIxfWV83ERzNo3vT6UwcOfC+D9H2m/Nz8RZO3qX5BIja8Ium32gu
ZVpSl/BXFJnHhOPZiQAAACJtYXJwaWVyY0BNYXJsb25zLU1hY0Jvb2stUHJvLmxvY2FsAQ
IDBAUGBw==
-----END OPENSSH PRIVATE KEY-----
```

SSH Public Key Example

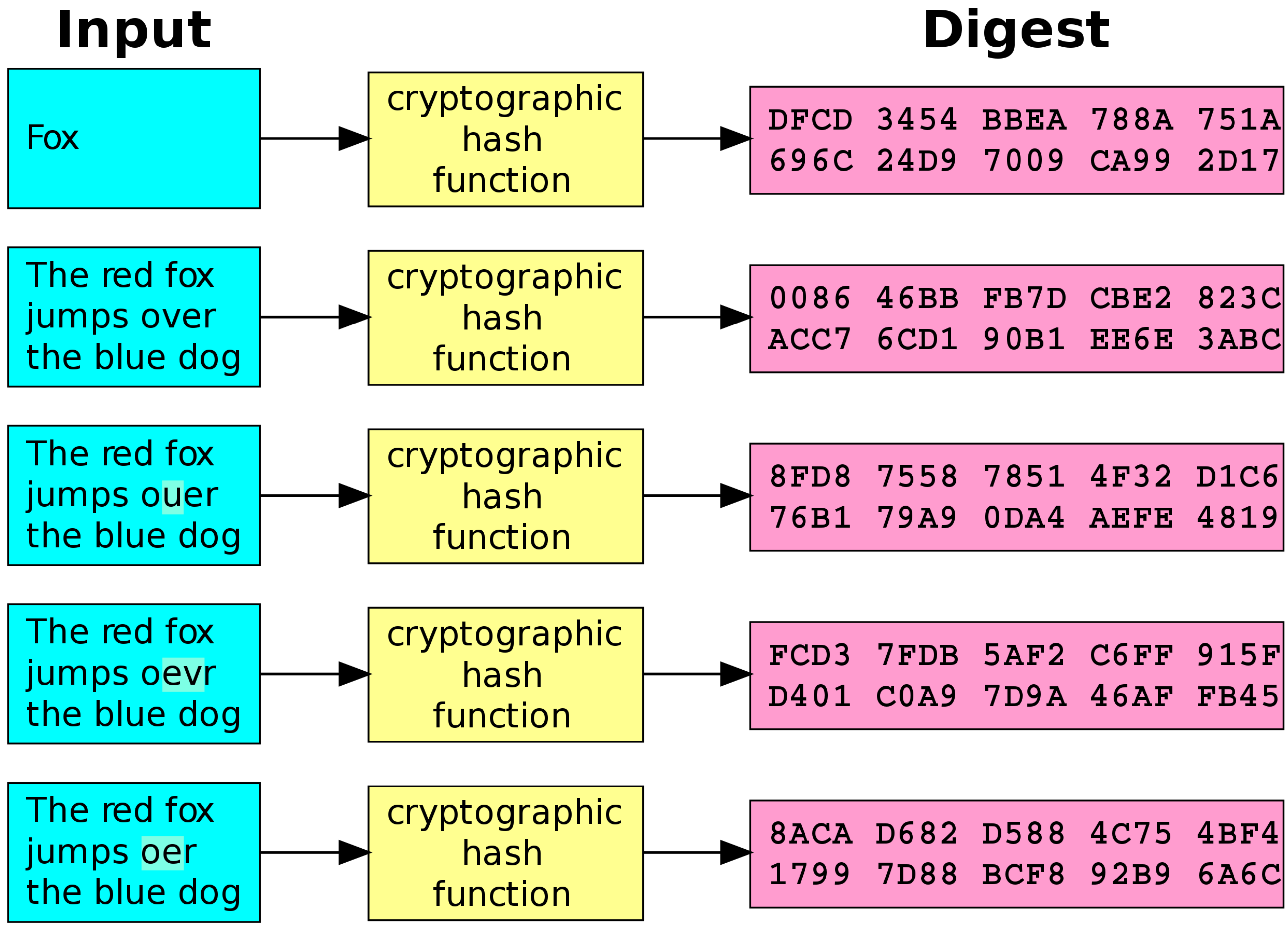
```
Marlons-MacBook-Pro:~ marpierc$ more junk_keyfile.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCv727OvmHK4fH4gE35fCFUapxE9MEyDP2GKgy75/ndYWo/cS5MZm+4Rb
VcBXk2bYBy+vQNVeUPuurPp6ZyYLCg5MR6xfsgAkABaUpaGad6p0tJDlljstKWTeh6ywz4tblVUhQKzWhlJ39q
22x3ceoWh+WlkciZyD2o4ghDL8ZMCEZfUn05pV96nYpDFgSQ5NVBfEUx1A7rIRzJFALxHYe5qaKgR6+tew5cov
BB1KB1hEeESYyy7ROmZ3orDkrwgP80TTuKYviPy7iHD1eSoD2jdgASSJNYE3YgNyUk86qmxLaZTaDAnInU8Igq
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

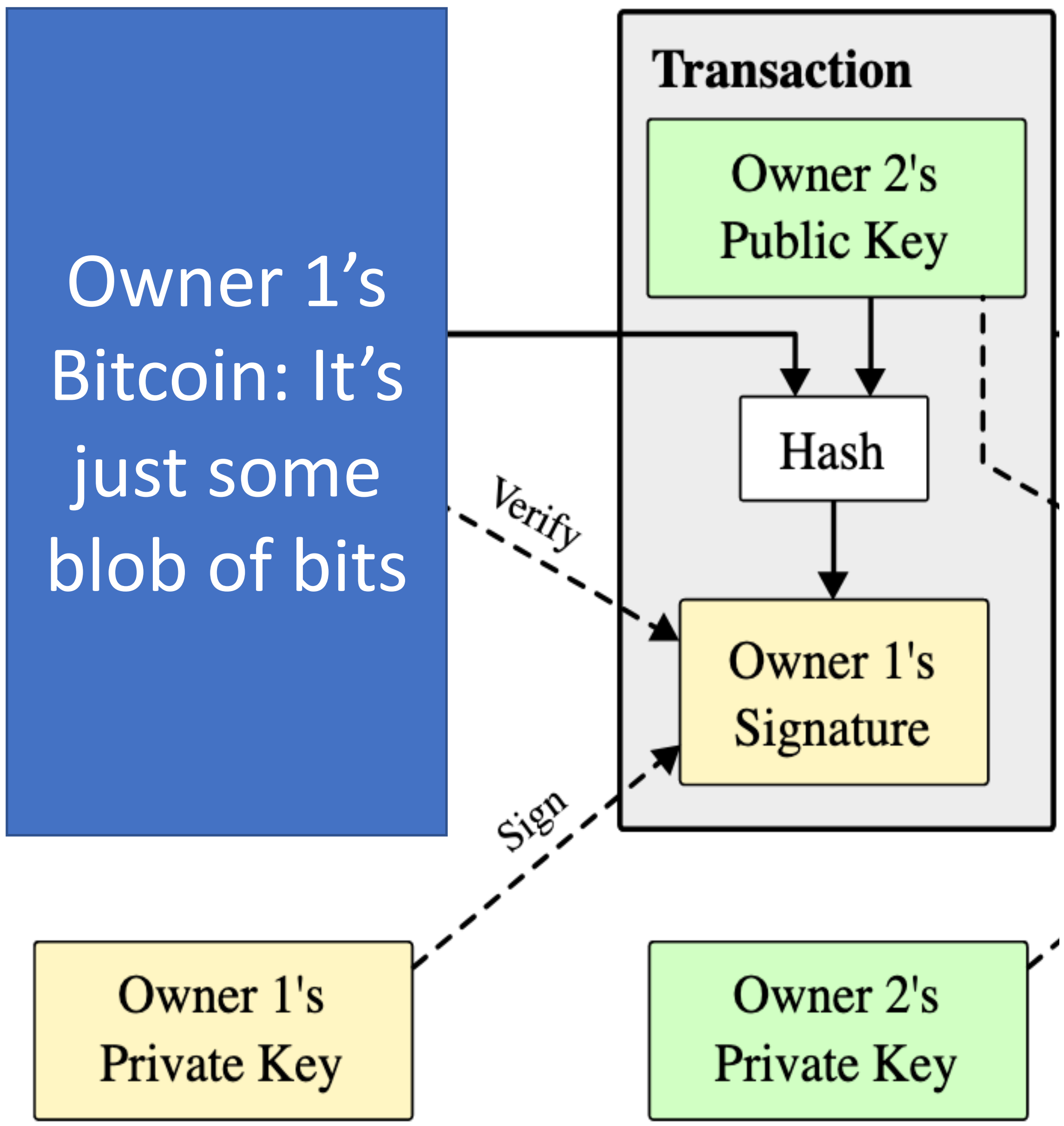
Cryptographic Hashing

- 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
Fox
cryptographic hash function
DFCD 3454 BBEA 788A 751A 696C 24D9 7009 CA99 2D17
The red fox jumps over the blue dog
cryptographic hash function
0086 46BB FB7D CBE2 823C ACC7 6CD1 90B1 EE6E 3ABC
The red fox jumps ouer the blue dog
cryptographic hash function
8FD8 7558 7851 4F32 D1C6 76B1 79A9 0DA4 AEFE 4819
The red fox jumps oevr the blue dog
cryptographic hash function
FCD3 7FDB 5AF2 C6FF 915F D401 C0A9 7D9A 46AF FB45
The red fox jumps oer the blue dog
cryptographic hash function
8ACA D682 D588 4C75 4BF4 1799 7D88 BCF8 92B9 6A6C

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.


Owner 1’s Bitcoin: It’s just some blob of bits
Transaction
Owner 2's Public Key
Hash
Verify
Owner 1's Signature
Sign
Owner 1's Private Key
Owner 2's Private Key

Steps in a Transaction

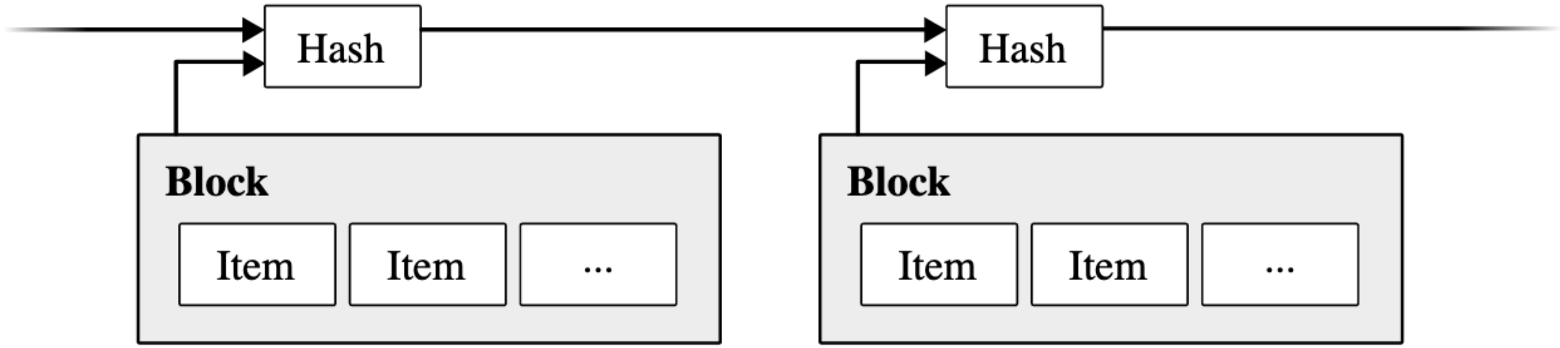
- Only Marlon (Owner 1) can spend the coin because his public key is embedded in the last block.
- Marlon transfers the coin to Suresh (Owner 2) by digitally signing a hash of the previous transaction and Suresh’s public key
- Marlon adds these to the end of the coin.
- Suresh can verify the signatures 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 Isuru for tacos at the same time?
- This is called double-spending.

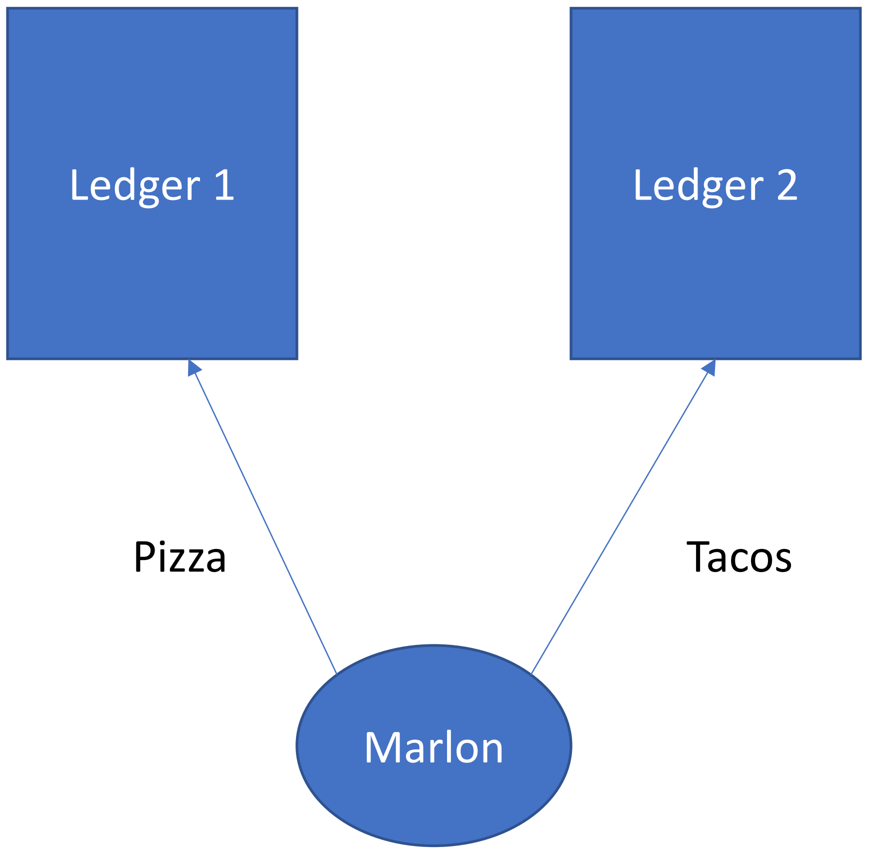
Bitcoin's Solution, Part 1: Timestamping and Ledgers

- We publish transaction 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.

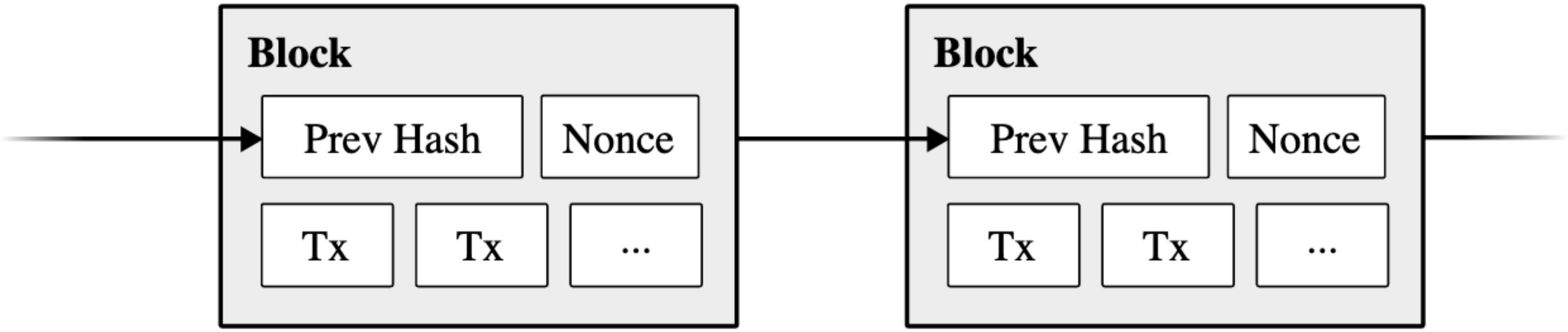


But What If Marlon Sends Two Transactions with the Same Time Stamp?

- We need a way, independent of Marlon's assertions, to order the transactions and prune out extraneous ones.
- If we have multiple ledgers, we can't rely on real time stamps for ordering.
- We need a way to logically order ledger entries.
- Different ledgers must have the same ordering.



Bitcoin’s Solution, Part 2: Proof of Work



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

0000000000000756af69e2ffbdb930261873cd71

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

New transactions are broadcast to all ledger nodes.

Each node collects new transactions into a block.

Each node works on finding a difficult proof-of-work for its block.

When a node finds a proof-of-work, it broadcasts the block to all nodes.

Nodes accept the block only if all transactions in it are valid and not already spent.

Nodes express their acceptance of the block by working on creating the next block in the chain

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 for the proof of work foils Marlon’s attempt at double-spending.

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 quickly (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 eventually consistent.

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

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