CS5412 Recitation

Blockchain

Block Chain

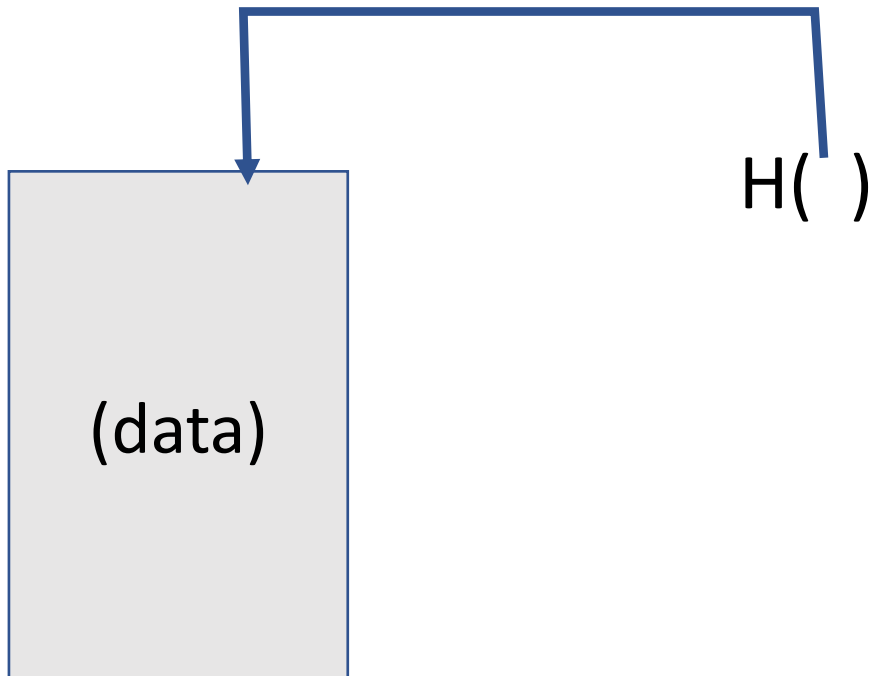
Block Chain

- Block chain is an open, distributed ledger that can record transactions between two parties efficiently in a verifible and permanent way

- A blockchain provides
  - Coordination between many parties
  - When there is no single trusted party

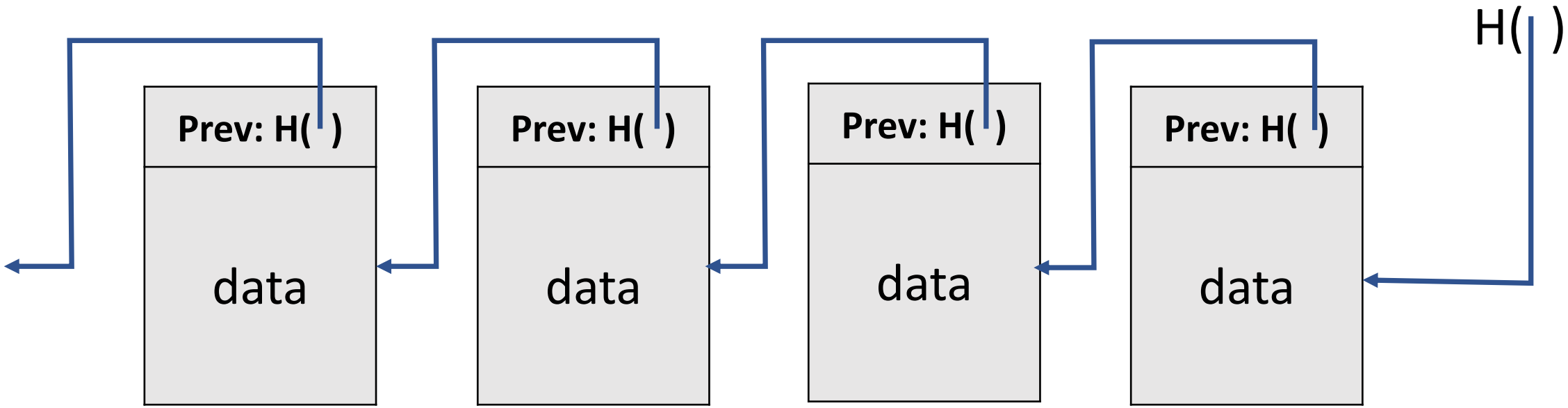
Block Chain

- Hash pointer: a hash pointer is a pointer where data is stored together with a cryptographic hash of the value of this data at some fixed time



Block Chain

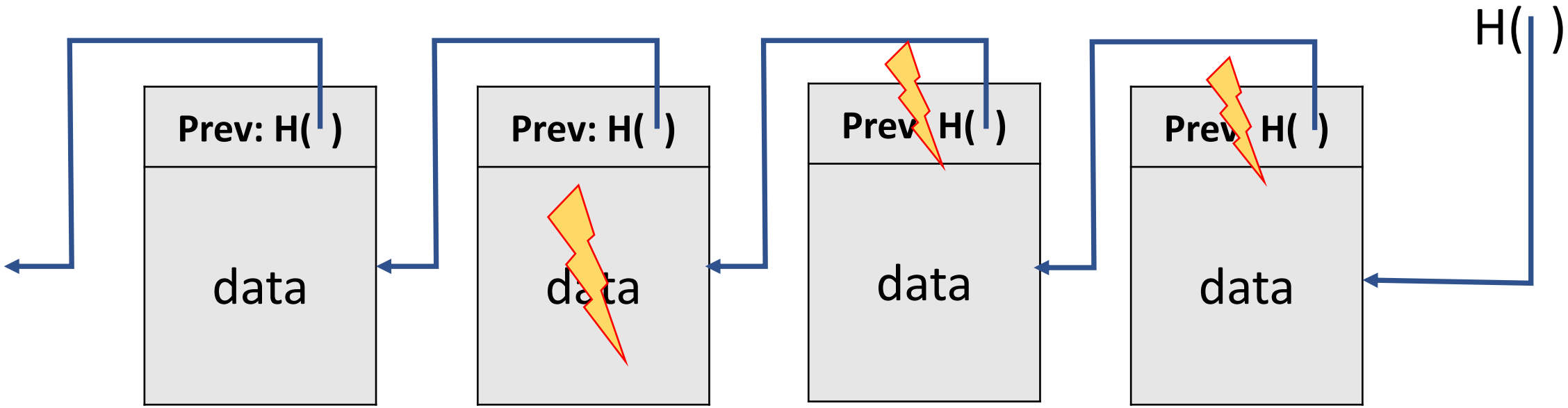
- In blockchain, the previous-block pointer is a hash pointer (similar to linked list)
- Each block tells us:
  - where the value of the previous block is
  - a digest of that value, which allows us to verify that the value hasn't been changed
- We store the head of the list, which is a hash-pointer that points to the most recent data block.



Block Chain

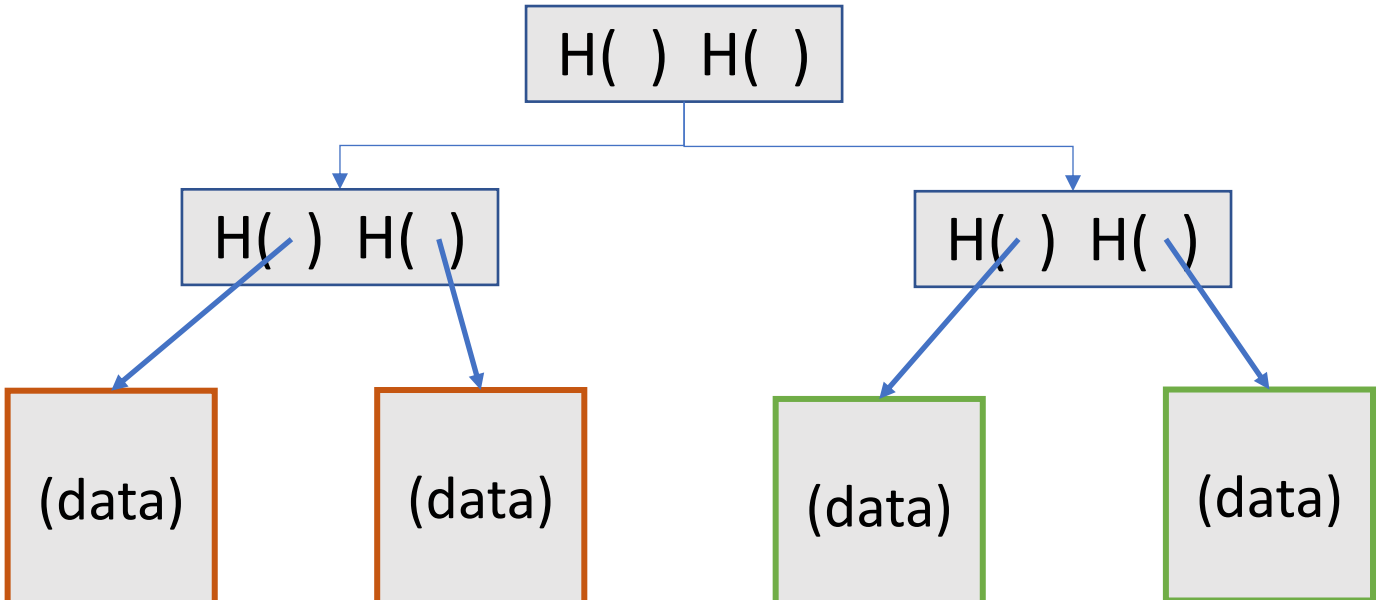
--- tamper-proof

- Suppose an adversary change the data of block **k.** Then the data in block **(k+1)**, which is the hash of the entire block k, is not going to match up.
- So we detect the inconsistency between the new data in block k and the hash pointer in block **(k+1)**.
- If the adversary continue to try and cover up this change by changing the next block's hash as well, it will fail when it reaches the head of the list



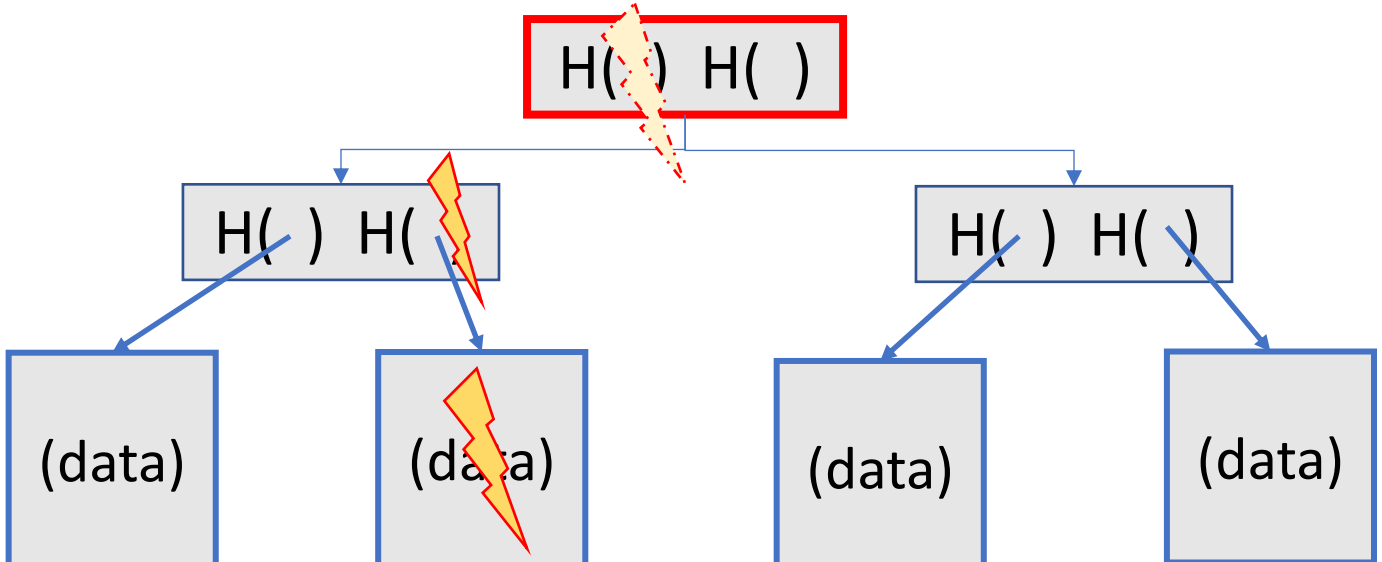
Merkle Tree

- Merkle tree is a tree structure built using hash pointers
- Example: suppose we have 4 blocks containing data.
  - We group these data blocks into pair of two, and then for each pair we build a data structure that has two hash pointers
  - These hash pointers data structures make up the next level of the tree, we then in turn group these into groups of two, and create new hash pointers for this level
  - …



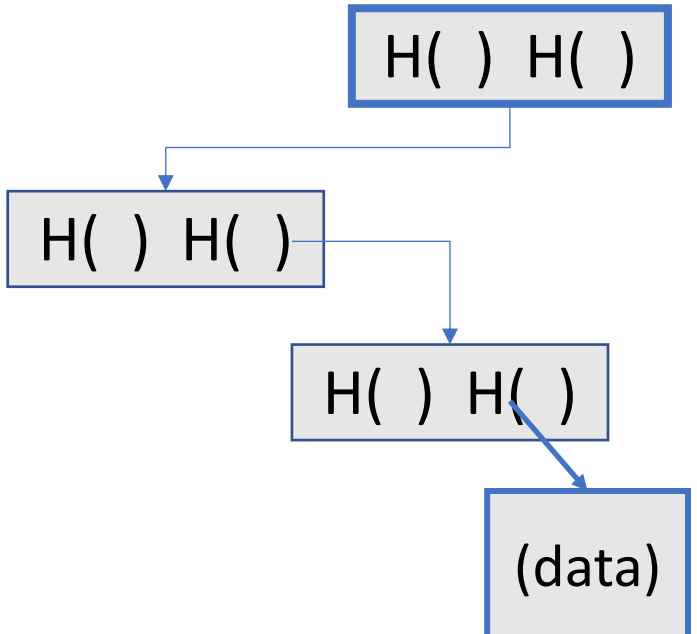
Merkle Tree --- tamper-proof

- We only need to remember one hash pointer, the one at the root of the tree. We can then traverse through the hash pointers to any point in the list.
- Temper-proof:
  - If an adversary tampers with some data block at the bottom of the tree
  - the change will cause the hash pointer one level up to not match
  - Even if the adversary continue to tamper with other blocks farther up the tree, the change will eventually propagate to the top, where it wont' be able to tamper with the hash pointer that all users stored.



Merkle Tree --- proof of members

- Merkle tree allows a concise proof of membership.
- To prove a data block is included in the tree only requires showing the blocks in the path from that data block to the root.
- If there are n nodes in the tree, only about log(n) items need to be shown.



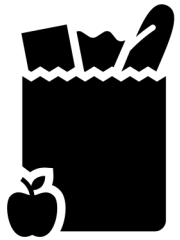
Permissioned & Permissionless Block Chain

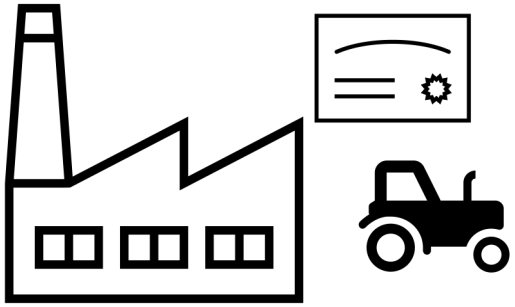
- Permissioned Block Chain:
  - Permissioned blockchains use an access control layer to govern who has access to the network.
  - In contrast to public blockchain networks, validators on private blockchain networks are vetted by the network owner.
- Permissionless Block Chain
  - allow anyone to transact and join as a validator.
  - The data on these blockchains is publicly available, and complete copies of the ledgers are stored across the globe.

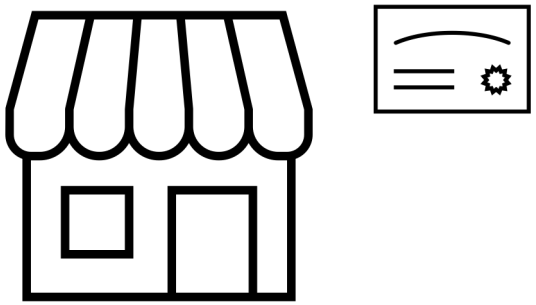
Case Study1: Block chain in IoT Food Industry

Permissioned or permissionless?

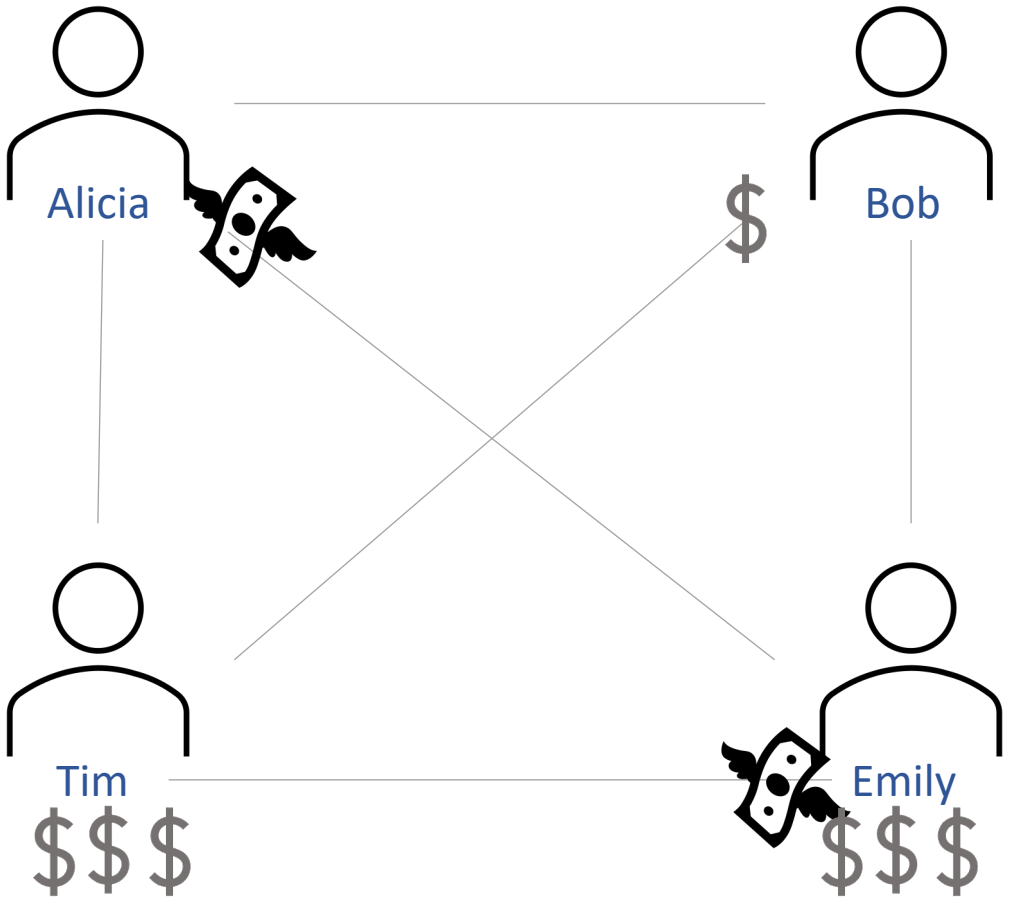
- To track the process of food production
- Blockchain helps to create a digital certificate forr each piece of food to prove
  - where it came and where it has been?
  - Was it kept at the right temperature areas?
  - Who has been in contact with it?







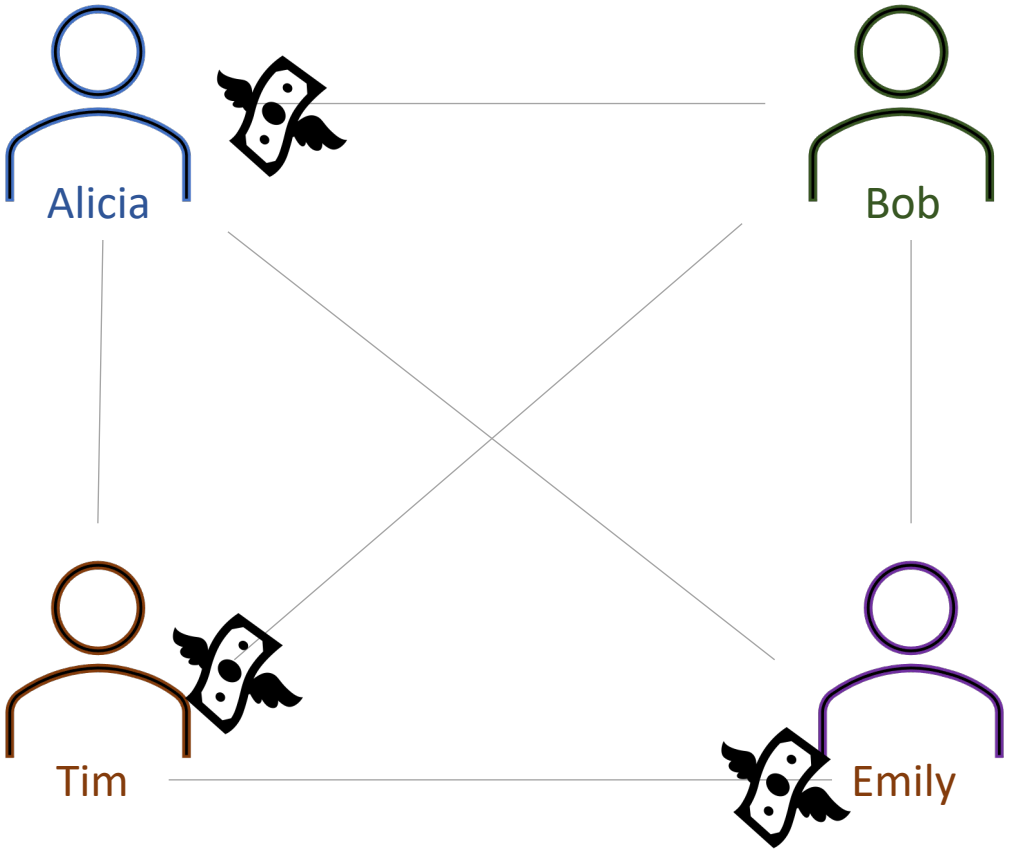
Case Study2: CryptoCurrency -- BitCoin



Case Study2: CryptoCurrency -- BitCoin

Ledger

Alicia pays Bob	$20
Tim pays Bob	$30
Emily pays Tim	$10

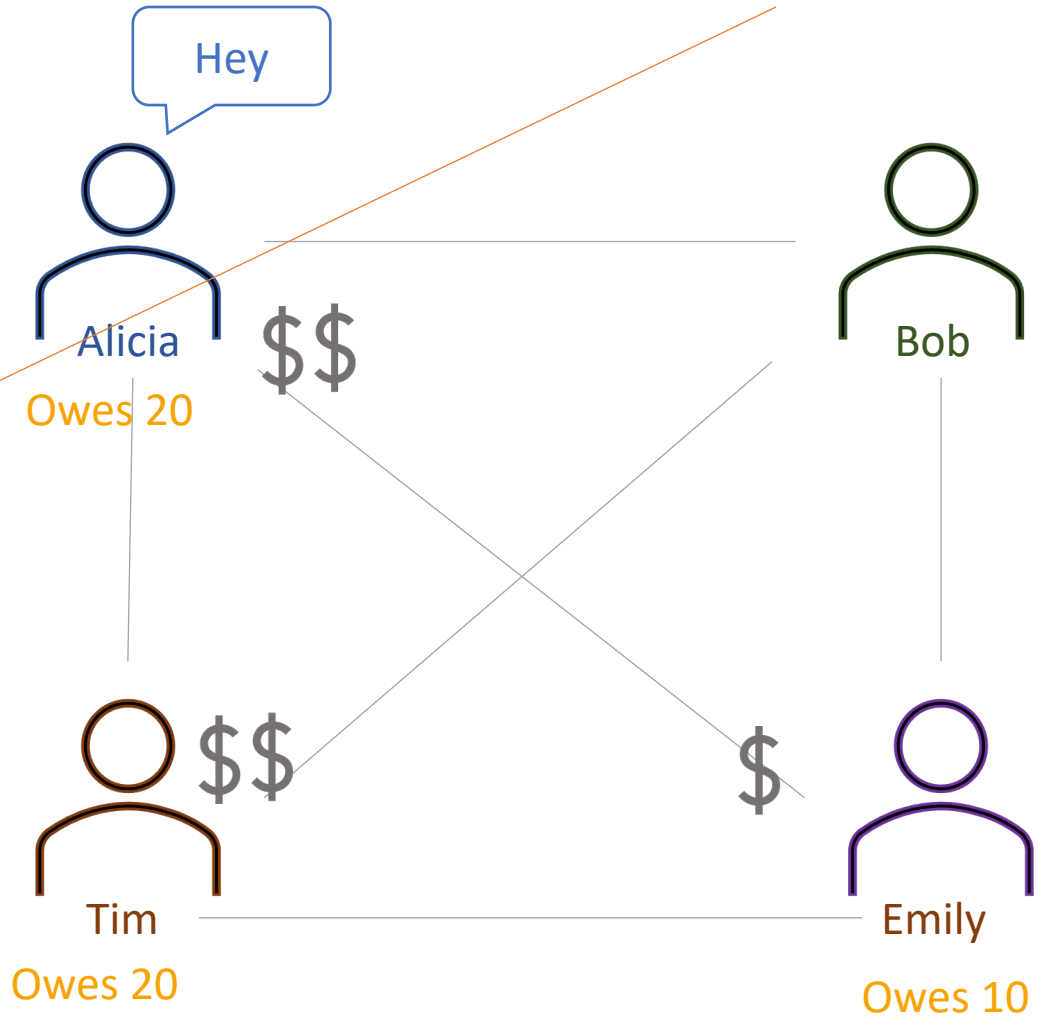


Case Study2: CryptoCurrency -- digital signature

Anyone can add a line

Ledger

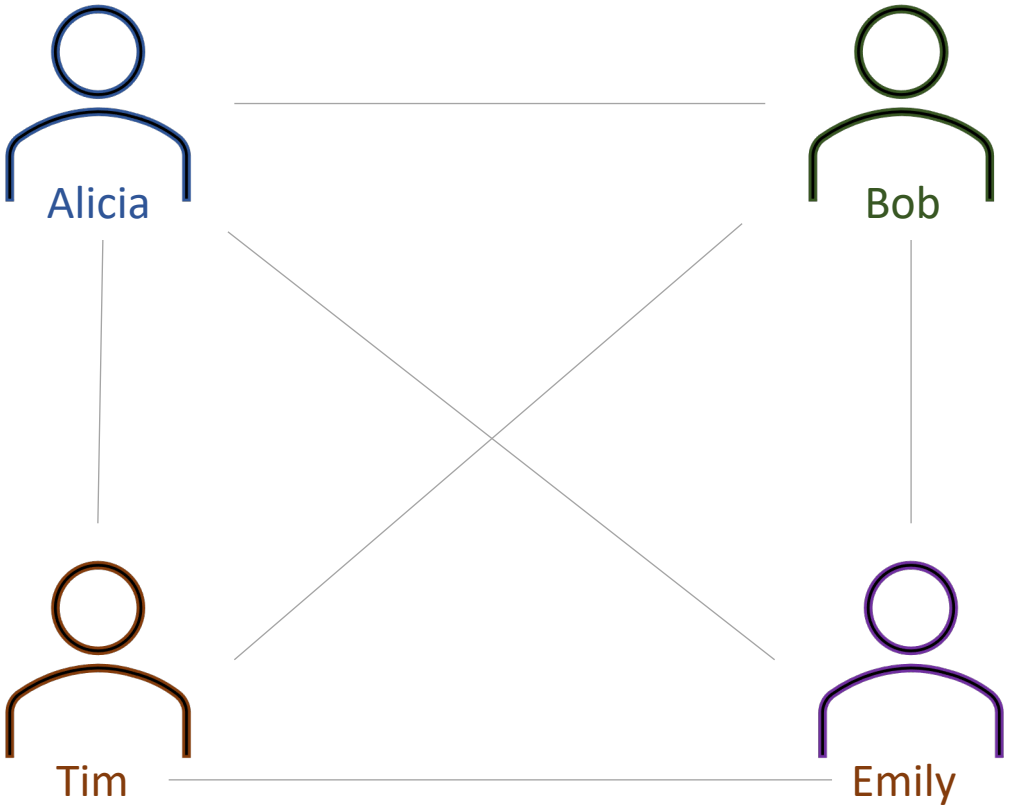
Alicia pays Bob	$20
Tim pays Bob	$30
Emily pays Tim	$10
Alicia pays Bob	$100



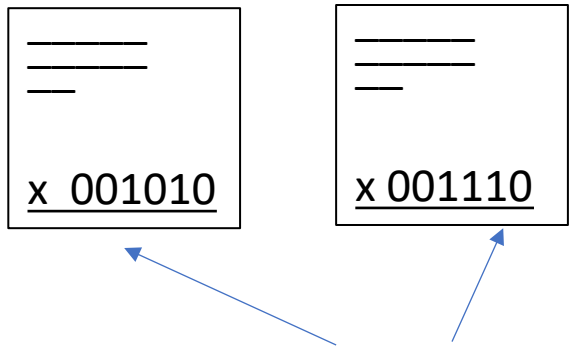
Case Study2: CryptoCurrency -- digital signature

Ledger

Alicia pays Bob	$20	alicia
Tim pays Bob	$30	Tim
Emily pays Tim	$10	Emily



Case Study2: CryptoCurrency -- digital signature



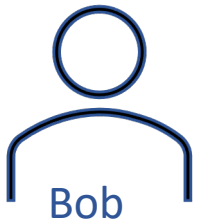
Difference messages

Completely different signatures

public_key: 010100011 …
secrete key: 000010010…

public_key: 010100101 …
secrete key: 000010111…

public_key: 010110101 …
secrete key: 000011010…



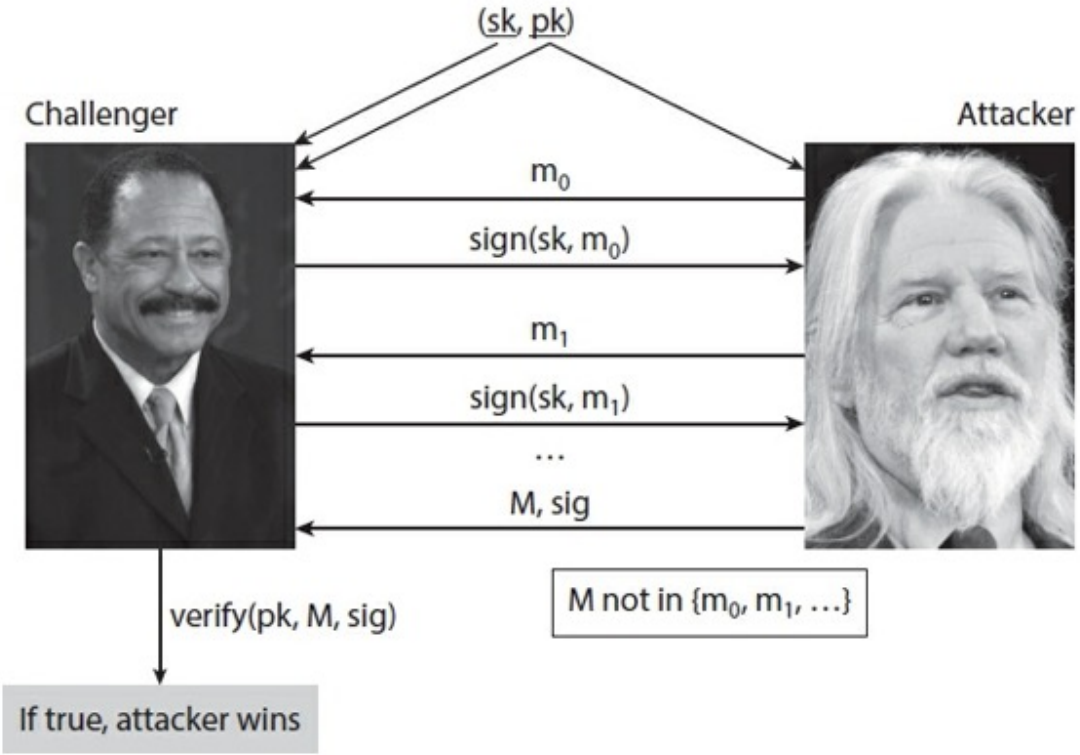
Case Study2: CryptoCurrency -- digital signature

Scheme:

- $(\boldsymbol{sk}, \boldsymbol{pk}) := \boldsymbol{generateKeys}(\boldsymbol{keysize})$ The generateKey method takes a key size and generate a key pair of secrete key(sk), and public verification key (pk)
- $\boldsymbol{sig} := \boldsymbol{sign}(\boldsymbol{sk}, \boldsymbol{message})$ the sign method takes a message and a secrete key(sk) as input and outputs a signature for message under sk
- $\boldsymbol{isValid} := \boldsymbol{verify}(\boldsymbol{pk}, \boldsymbol{message}, \boldsymbol{sig})$ the verify method takes a message, a signature, and a public key as input, and returns if a signature is valid for message under public key pk

Case Study2: CryptoCurrency -- digital signature

- $(sk, pk) := generateKeys(keysize)$
- $sig := sign(sk, message)$
- $isValid := verify(pk, message, sig)$



Two properties of signature:

1. Valid signature must verify, and its verification needs to be deterministic.

   $verify(pk, message, sign(sk, message)) == true$

2. Signatures are existentially unforgeable

Case Study2: CryptoCurrency -- digital signature

- $(sk, pk) := generateKeys(keysize)$
- $sig := sign(sk, message)$
- $isValid := verify(pk, message, sig)$

2^{256} possible signatures

```
0000000000000000000000000000000000
0000000000000000000000000000000000
0000000000000000000000000000000000
0000000000000000000000000000000000
0000000000000000000000000000000000
0000000000000000000000000000000000
0000000000000000000000000010001011
```

Verify(pk, Message, 256 bit signature) = T/F

Case Study2: CryptoCurrency -- digital signature

alicia

000010010011100...

x 001010

x 001110

Signature(Message, sk) = Signature

Verify(pk, Message, signature) = T/F

public_key: 010100011 ...
secrete key: 000010010...

public_key: 010100101 ...
secrete key: 000010111...

public_key: 010110101 ...
secrete key: 000011010...

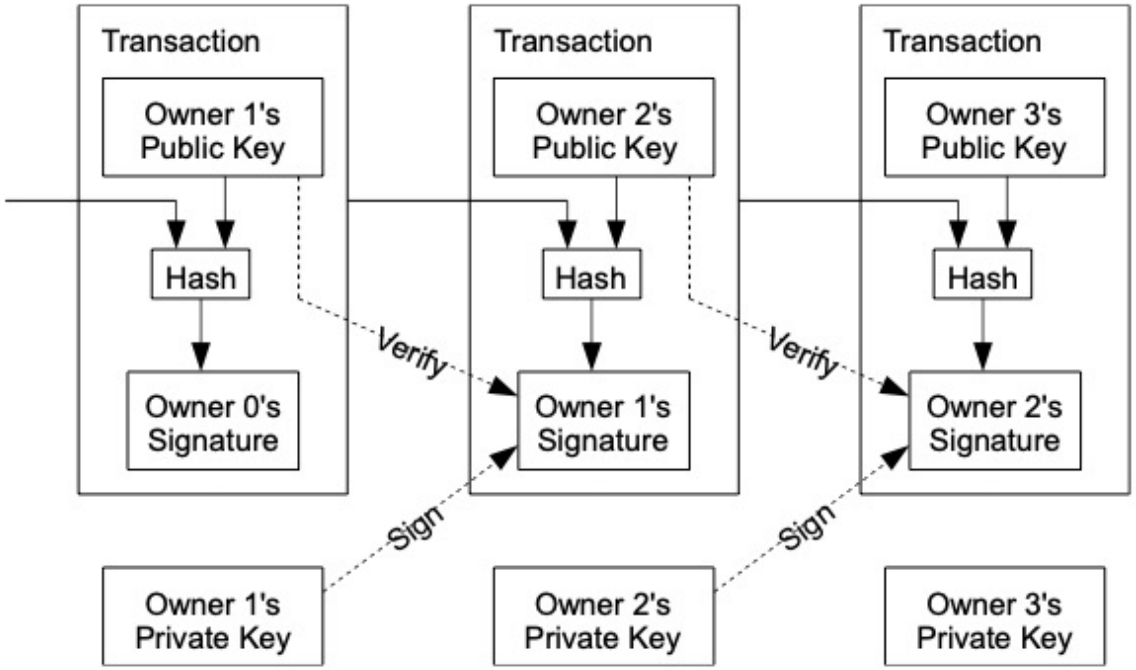
Alicia

Bob

Emily

Case Study2: CryptoCurrency -- BitCoin

- Digital Signatures, Timestamp server
- The ledger is the currency

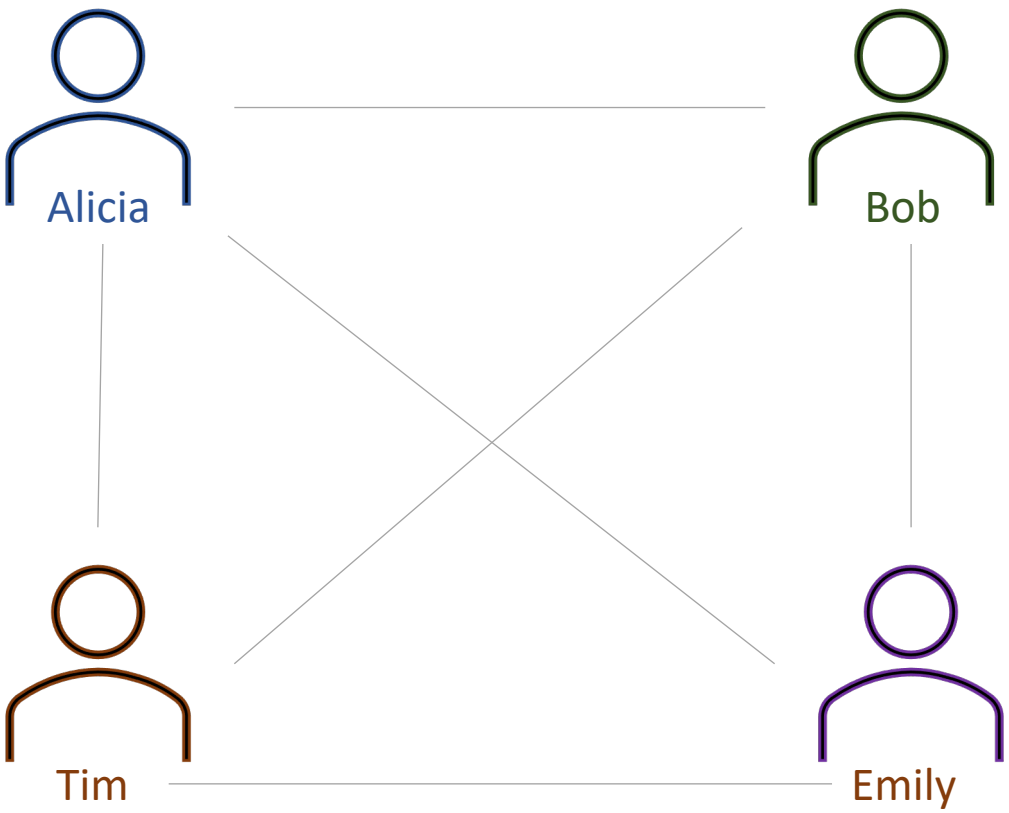


Case Study2: CryptoCurrency -- BitCoin

Where is this? Who control this?

Ledger

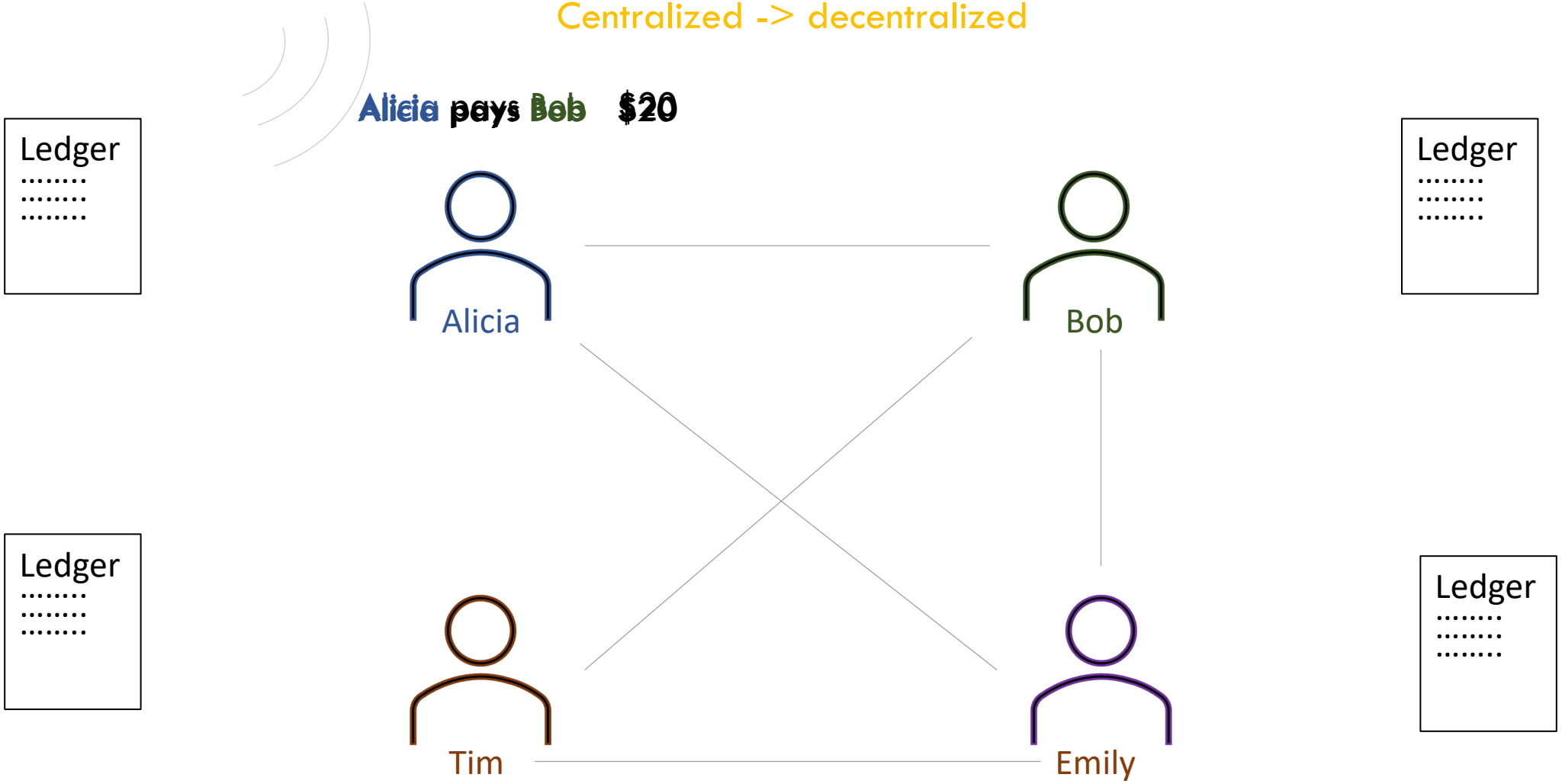
0	Alicia pays Bob	$20	001001 alicia
1	Alicia pays Bob	$20	001011
2	Alicia pays Bob	$20	001111



Transaction history ⇔ Ledger

Case Study2: CryptoCurrency -- BitCoin

Centralized -> decentralized



Case Study2: BitCoin

- Digital Signatures, Timestamp server
- The ledger is the currency
- Decentralize (anyone can add to the block)

Proof of Work

- A mechanism could be used to reach consensus among many nodes in the network, and to secure the Bitcoin blockchain

Proof of Work

Cryptographic
Hash Function

SHA256(“DairyFarm1”) = 00001100010001000010000010000010110 ….

Message/File

SHA256(“Diaryfarm1”) = 11011001100001111011001100100001 11 ….

Proof of Work & Verification

Probability of the below reuirement?

P = 1/2^30 ~ 1/1,000,000,000

Ledger

Alicia pays Bob $20

1,2,3,4,5,…….

SHA256 →

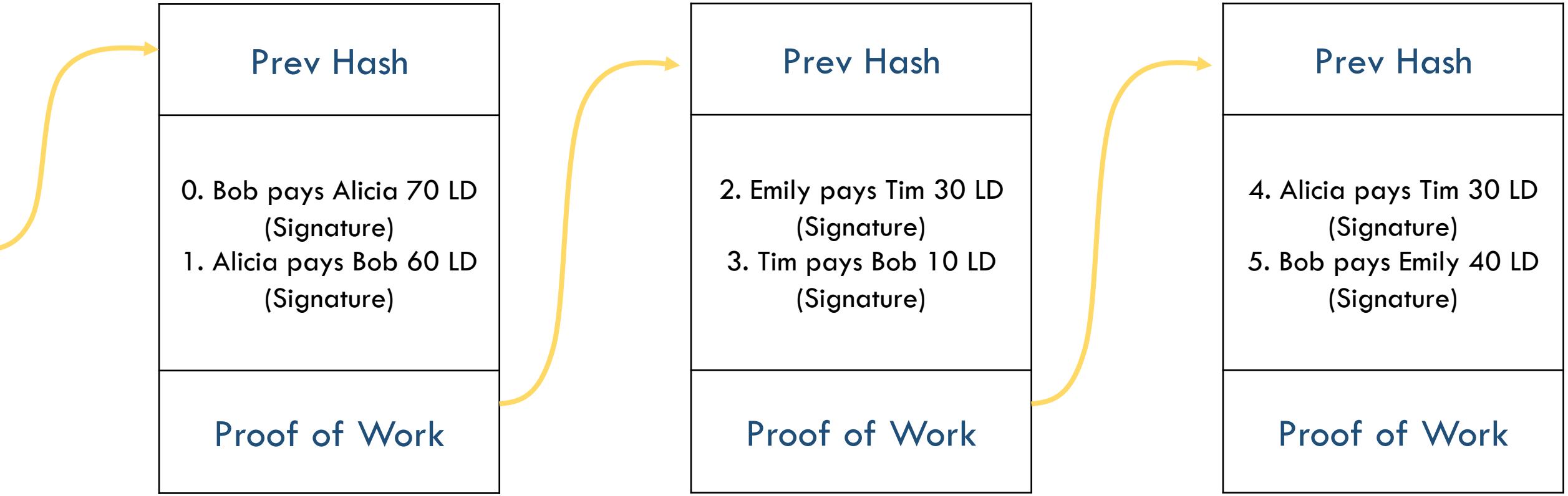
30 zeros

000000000000000000000011
000100010000100001000001
011000000100111010100100
101010010010110011001011
11….

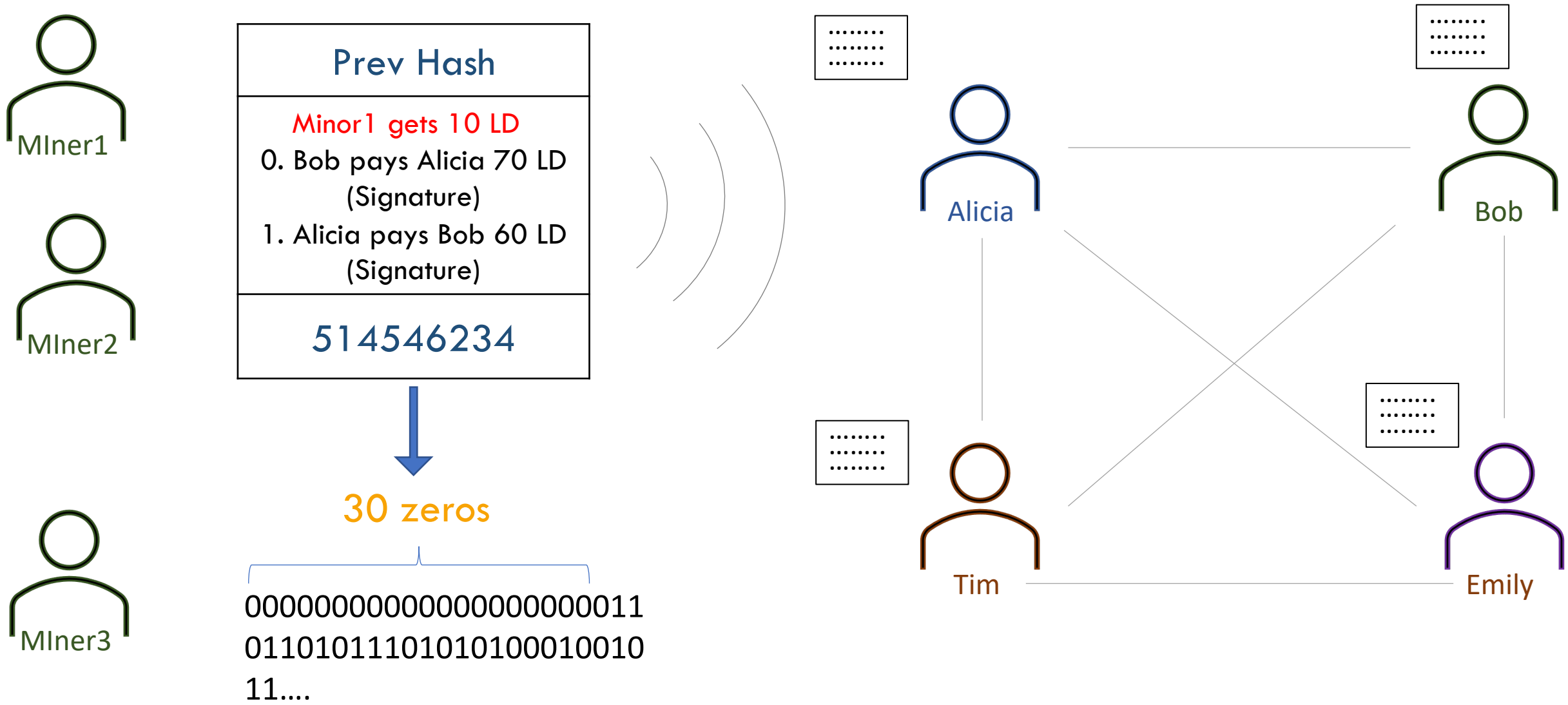
Case Study2: BitCoin

- Digital Signatures, Timestamp Server
- The ledger is the currency
- Decentralize
- Broadcast transactions

Blocks

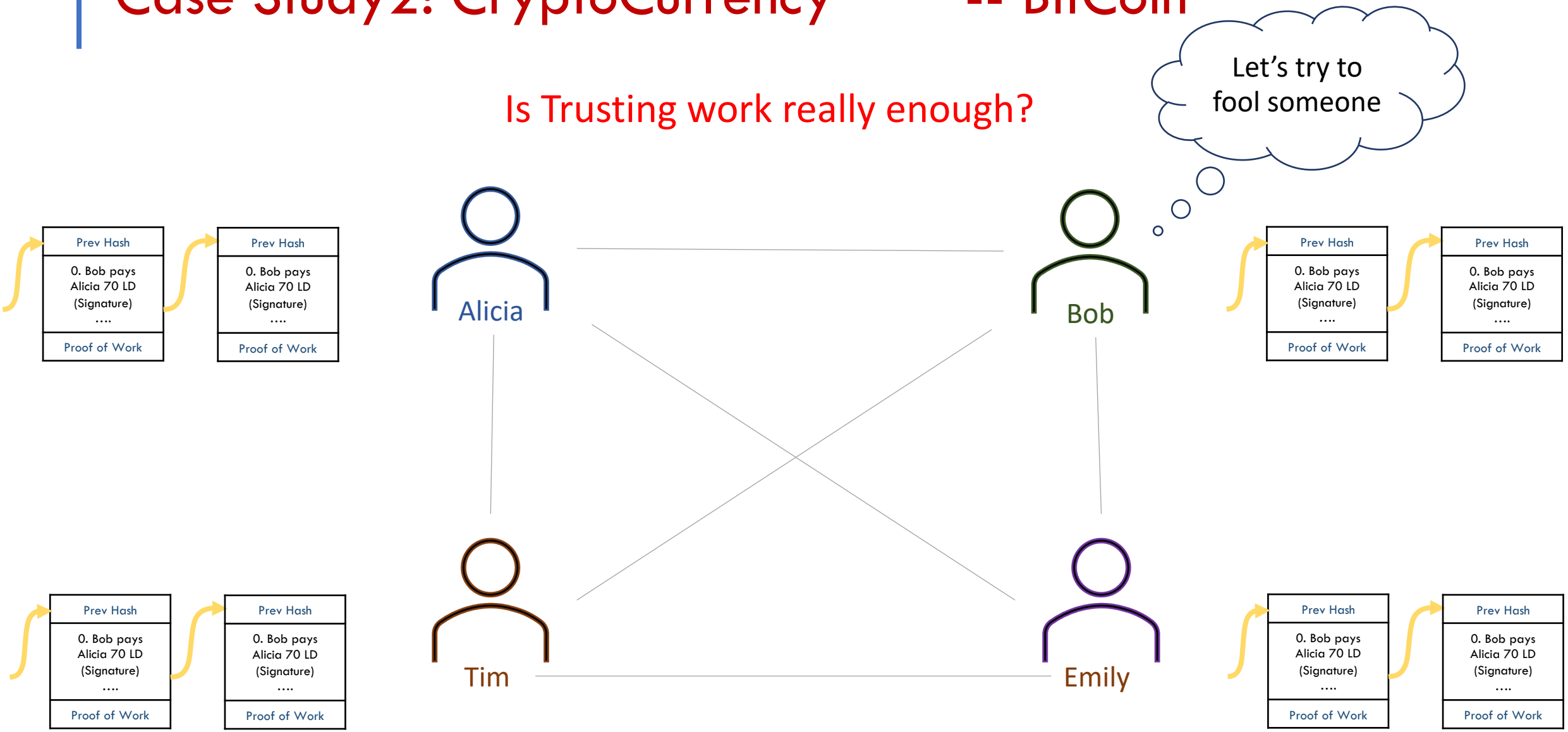

Prev Hash
0. Bob pays Alicia 70 LD
(Signature)
1. Alicia pays Bob 60 LD
(Signature)
Proof of Work
Prev Hash
2. Emily pays Tim 30 LD
(Signature)
3. Tim pays Bob 10 LD
(Signature)
Proof of Work
Prev Hash
4. Alicia pays Tim 30 LD
(Signature)
5. Bob pays Emily 40 LD
(Signature)
Proof of Work

Case Study2: CryptoCurrency -- BitCoin

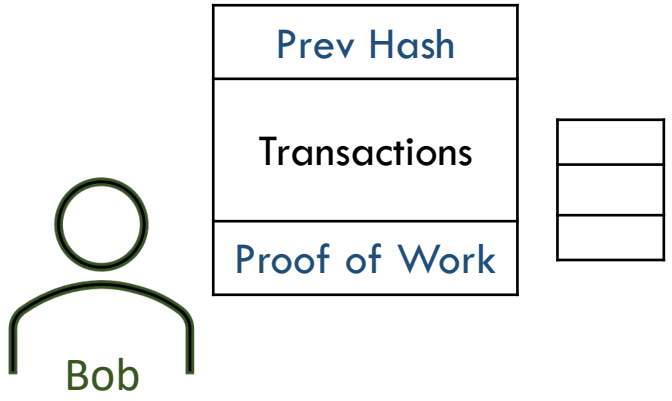


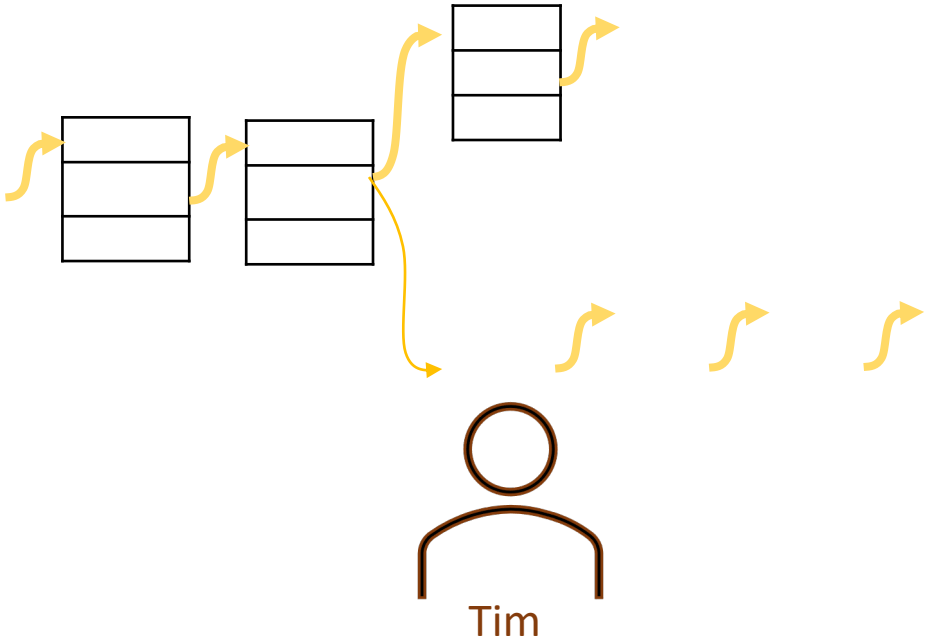
Case Study2: CryptoCurrency -- BitCoin

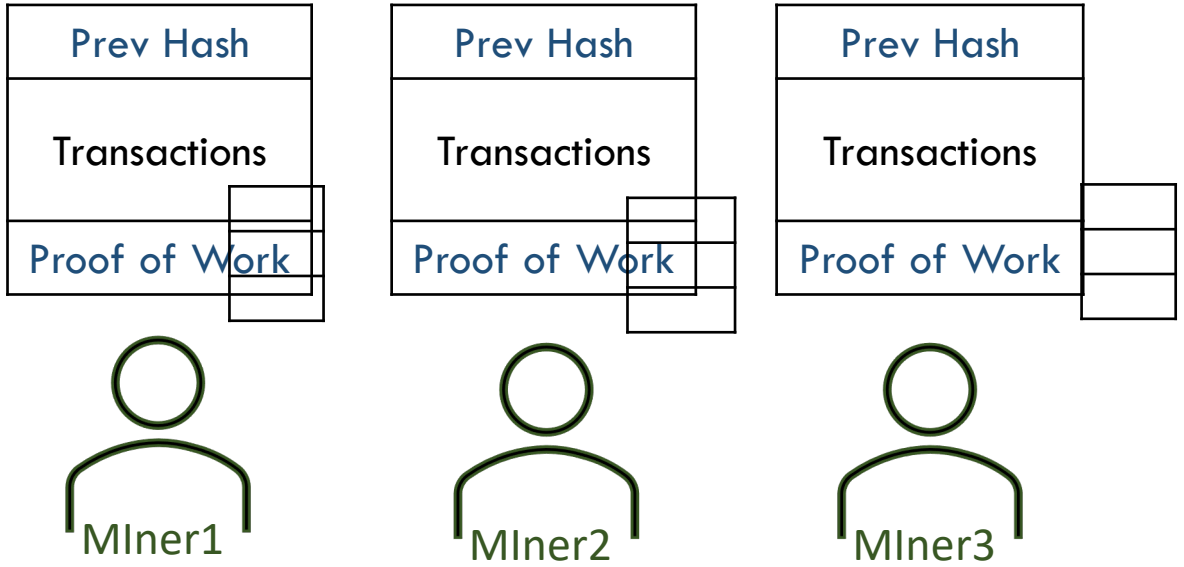
Is Trusting work really enough?



Blocks







- Always trust the longest chain.
- Unless Bob has close to 50% of all the computing resources, the probability of Bob can add blocks faster than everyone else is very small.

Blocks --- Average Block Time

BTC: 10 Minutes

ETH: 15 Seconds

XRP: 3.5 Seconds

LTC: 2.5 Minutes

Case Study2: BitCoin

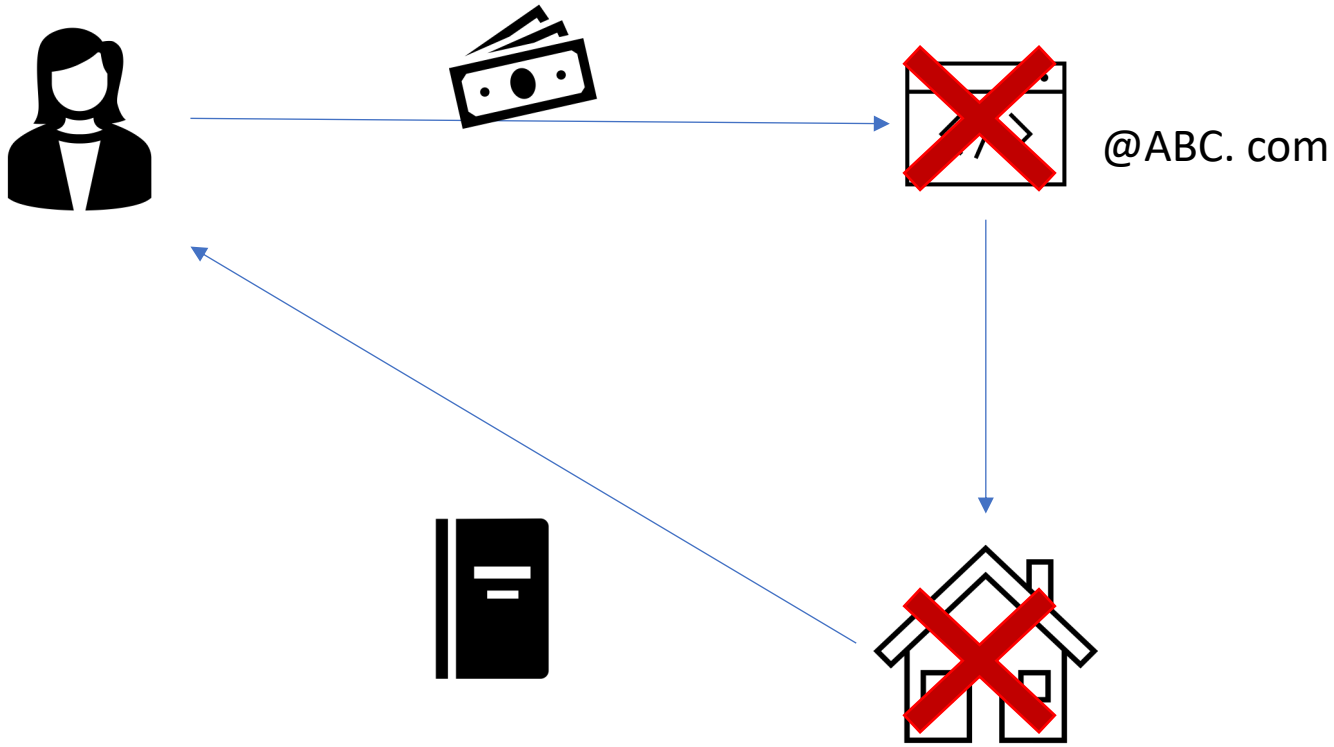
Permissioned or permissionless?

- Digital Signatures, Timestamp Server
- The ledger is the currency
- Decentralize
- Proof of work
- Block chain

Case Study3

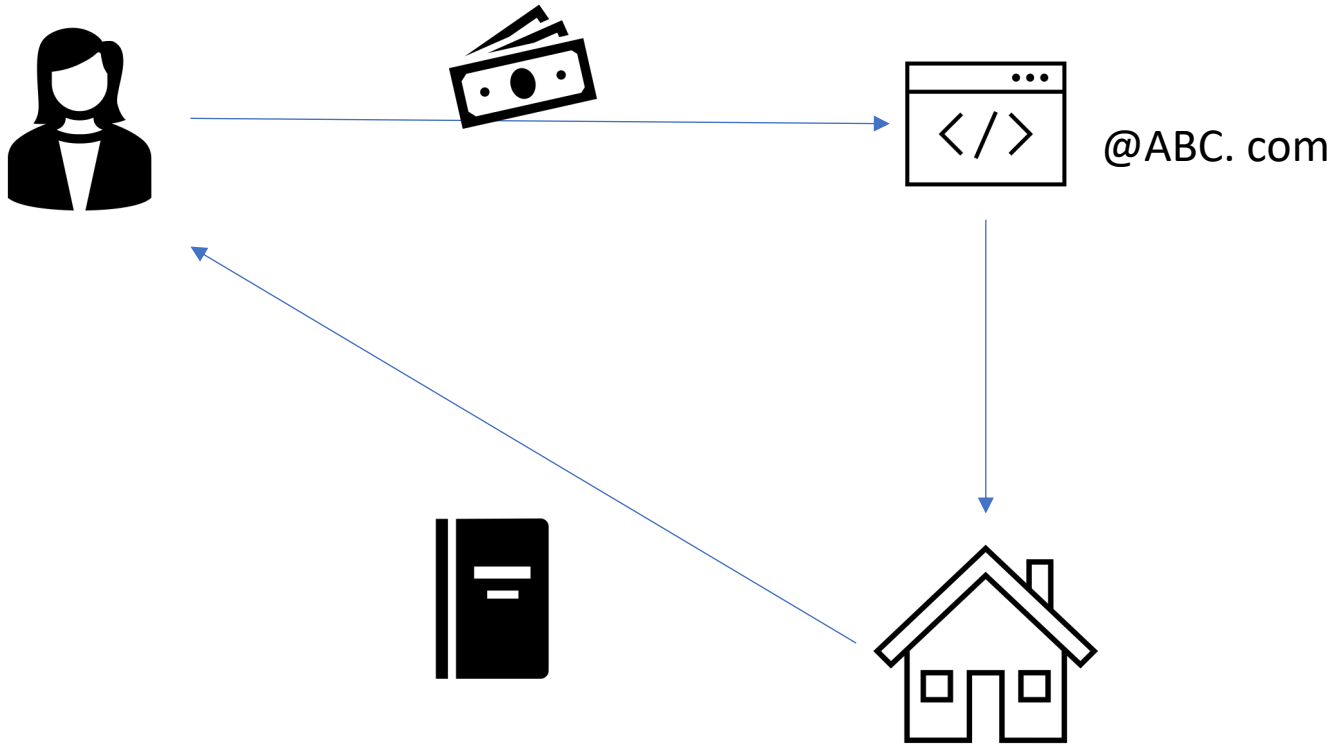
- Question:

How to make sure person A make the payment only when the book is received?



Case Study3: Smart Contract

If … then …..



Case Study3: Smart Contract

```
pragma solidity 0.6.11;

contract VendingMachine {

    // Declare state variables of the contract
    address public owner;
    mapping (address => uint) public cupcakeBalances;

    // When 'VendingMachine' contract is deployed:
    // 1. set the deploying address as the owner of the
contract
    // 2. set the deployed smart contract's cupcake balance to
100
    constructor() public {
        owner = msg.sender;
        cupcakeBalances[address(this)] = 100;
    }

    // Allow the owner to increase the smart contract's
cupcake balance
    function refill(uint amount) public {
        require(msg.sender == owner, "Only the owner can
refill.");
        cupcakeBalances[address(this)] += amount;
    }

    // Allow anyone to purchase cupcakes
    function purchase(uint amount) public payable {
        require(msg.value >= amount * 1 ether, "You must pay
at least 1 ETH per cupcake");
        require(cupcakeBalances[address(this)] >= amount, "Not
enough cupcakes in stock to complete this purchase");
        cupcakeBalances[address(this)] -= amount;
        cupcakeBalances[msg.sender] += amount;
    }
}
```

- Smart Contract written in code, stored inside a block chain
- Everyone can access and interact with the contract (verify it)

Case Study3: Smart Contract

Permissioned or permissionless?

- Smart Contract written in code, stored inside a block chain
- Everyone can access and interact with the contract (verify it)
- Consensus about outcome
- Send & receive coins

Q&A

Reference

1. EpidemicAlgorithms for ReplicatedDatabaseMaintenance, A. Demers,D. Greene,C.WesIrish, J. Larson. S. Shenker,H. Sturgis,D. Swinehart,and D. Terry, ACM 1987
2. Bimodal multicast, K. Birman, M.Hayden, O. Ozkasa, M. Budiu, Y.Minsky, ACM 1999, https://dl-acm-org.proxy.library.cornell.edu/doi/pdf/10.1145/312203.312207
3. Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining, R. Van Renesse, K. Birman, W. Vogels, ACM 2003, https://dl-acm-org.proxy.library.cornell.edu/doi/pdf/10.1145/762483.762485
4. Bitcoin: A Peer-to-Peer Electronic Cash System, Satoshi Nakamoto, https://bitcoin.org/bitcoin.pdf
5. Bitcoin and Cryptocurrency Technology: A comprehensive introduction. Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, Steven Goldfeder
6. What is a Merkle Tree? Beginner's Guide to this Blockchain Component. Brian Curran, https://blockonomi.com/merkle-tree/
7. Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies, Florian Tschorsch Björn Scheuermann,
8. The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication, Marko Vukolić, https://link.springer.com/chapter/10.1007/978-3-319-39028-4_9
9. The Tangle: an Illustration Introduction. https://blog.iota.org/the-tangle-an-illustrated-introduction-4d5eae6fe8d4/
10. Video Tutorial of Bitcoin: https://www.youtube.com/watch?v=bBC-nXj3Ng4
11. Introduction to Smart Contracts by Ethereum: https://ethereum.org/en/developers/docs/smart-contracts/