CS5412 Recitation
Blockchain
Block Chain
Block Chain

- Block chain is an open, distributed ledger that can record transactions between two parties efficiently in a verifiable 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.
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.
• 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
  • ...

```
H( ) H( )
```

```
H( ) H( )
```

```
H( ) H( )
```

```
H( ) H( )
```

```
(data) (data) (data) (data)
```
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 continues to tamper with other blocks farther up the tree, the change will eventually propagate to the top, where it won't be able to tamper with the hash pointer that all users stored.
Merkle Tree

• 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 Study 1: Block chain in IoT Food Industry

• To track the process of food production
• Blockchain helps to create a digital certificate for 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 Study 2: CryptoCurrency

-- BitCoin

Diagram showing the exchange of currencies among Alicia, Tim, Bob, and Emily without involving a bank.
Case Study 2: CryptoCurrency -- BitCoin

Ledger

<table>
<thead>
<tr>
<th>Person</th>
<th>Action</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alicia</td>
<td>pays</td>
<td>Bob $20</td>
</tr>
<tr>
<td>Tim</td>
<td>pays</td>
<td>Bob $30</td>
</tr>
<tr>
<td>Emily</td>
<td>pays</td>
<td>Tim $10</td>
</tr>
</tbody>
</table>
Case Study 2: CryptoCurrency  

-- digital signature

**Ledger**

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

Anyone can add a line
Case Study 2: CryptoCurrency -- digital signature

Ledger

Alicia pays Bob $20
Tim pays Bob $30
Emily pays Tim $10
Case Study 2: CryptoCurrency

-- digital signature

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

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

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

Difference messages
Completely different signatures

\[
x \cdot 001010
\]
\[
x \cdot 001110
\]
Case Study 2: CryptoCurrency  -- digital signature

Scheme:

- \((sk, pk) := generateKeys(keysize)\) The `generateKey` method takes a key size and generate a key pair of secret key (sk), and public verification key (pk).

- \(sig := sign(sk, message)\) the `sign` method takes a message and a secret key (sk) as input and outputs a signature for message under sk.

- \(isValid := verify(pk, message, sig)\) 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.
Two properties of signature:

1. Valid signature must verify, and its verification needs to be deterministic.
   \[
   \text{verify}(pk, \text{message}, \text{sign}(sk, \text{message})) = \text{true}
   \]

2. Signatures are existentially unforgeable

---

Case Study 2: CryptoCurrency

- \((sk, pk) := \text{generateKeys(keysize)}\)
- \(\text{sig} := \text{sign}(sk, \text{message})\)
- \(\text{isValid} := \text{verify}(pk, \text{message}, \text{sig})\)
Case Study 2: CryptoCurrency -- digital signature

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

$2^{256}$ possible signatures

Verify$(pk, \text{Message}, \text{256 bit signature}) = T/F$
Case Study 2: CryptoCurrency

-- digital signature

Signature(Message, sk) = Signature

Verify(pk, Message, signature) = T/F
Case Study 2: CryptoCurrency -- BitCoin

- Digital Signatures, Timestamp server
- The ledger is the currency
Case Study 2: CryptoCurrency

-- BitCoin

Where is this? Who control this?

Ledger

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

Transaction history ⇔ Ledger
Case Study 2: CryptoCurrency -- BitCoin

Centralized -> decentralized

Alicia pays Bob $20
Case Study 2: 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") = 000011000100010001000010000010110 ....

Message/File

SHA256("Diaryfarm1") = 11011001100001111100110010001000111 ....
Proof of Work & Verification

Ledger
Alicia pays Bob $20

1,2,3,4,5,....... SHA256

Probability of the below requirement?

P = \frac{1}{2^{30}} \approx \frac{1}{1,000,000,000}

30 zeros

000000000000000000000000011
000100010000100001000001
01100000010011101010010
01010100100101100110010
11....
Case Study 2: BitCoin

- Digital Signatures, Timestamp Server
- The ledger is the currency
- Decentralize
- Broadcast transactions
0. Bob pays Alicia 70 LD (Signature)
1. Alicia pays Bob 60 LD (Signature)

Proof of Work

2. Emily pays Tim 30 LD (Signature)
3. Tim pays Bob 10 LD (Signature)

Proof of Work

4. Alicia pays Tim 30 LD (Signature)
5. Bob pays Emily 40 LD (Signature)

Proof of Work
Case Study 2: CryptoCurrency -- BitCoin

<table>
<thead>
<tr>
<th>Prev Hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>Miner1 gets 10 LD</td>
</tr>
<tr>
<td>0. Bob pays Alicia 70 LD (Signature)</td>
</tr>
<tr>
<td>1. Alicia pays Bob 60 LD (Signature)</td>
</tr>
<tr>
<td>514546234</td>
</tr>
<tr>
<td>30 zeros</td>
</tr>
</tbody>
</table>

00000000000000000000000110110101110101001001001011....

30 zeros

Bob

Emily

Tim

Alicia

Miner1

Miner2

Miner3
Case Study 2: CryptoCurrency -- BitCoin

Is Trusting work really enough?

Let’s try to fool someone

Prev Hash
0. Bob pays Alicia 70 LD (Signature) ....
Proof of Work

Prev Hash
0. Bob pays Alicia 70 LD (Signature) ....
Proof of Work

Prev Hash
0. Bob pays Alicia 70 LD (Signature) ....
Proof of Work

Prev Hash
0. Bob pays Alicia 70 LD (Signature) ....
Proof of Work

Prev Hash
0. Bob pays Alicia 70 LD (Signature) ....
Proof of Work

Prev Hash
0. Bob pays Alicia 70 LD (Signature) ....
Proof of Work
• 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 Study 2: BitCoin

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

Permissioned or permissionless?
Case Study 3

• Question:

How to make sure person A make the payment only when the book is received?
Case Study 3: Smart Contract

If ... then .....
Case Study 3: Smart Contract

- Smart Contract written in code, stored inside a block chain
- Everyone can access and interact with the contract (verify it)
Case Study 3: Smart Contract

• 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


7. Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies, Florian Tschorsch Björn Scheuermann


10. Video Tutorial of Bitcoin: https://www.youtube.com/watch?v=bBC-nXj3Ng4