Introduction to blockchain

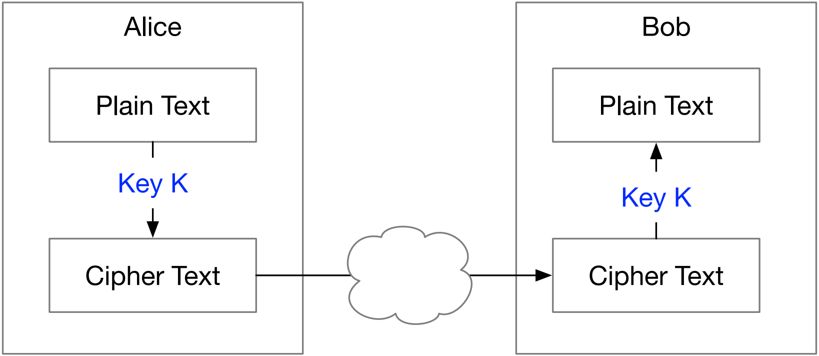
With bitcoin as an example

Zhehao Wang

Oct 2018

Symmetric cryptography

Consider symmetric cryptoraphy, where



The problem: how do I securely transfer *key K* over the network?

Asymmetric cryptography, RSA algorithm

Find 3 very large positive integers e, d, n s.t.

$$(m^e)^d \equiv m \pmod{n}, \quad \forall m, \ 0 \leq m \leq n$$

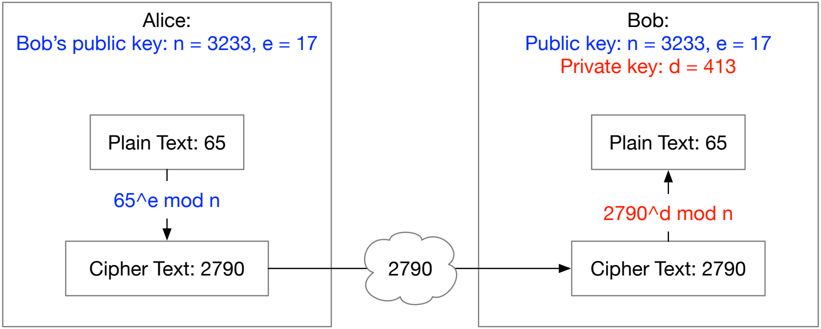
Knowing e, n or even m it's extremely hard to find d.

- Public key: n, e
- Private key: n, d
- Message: m

Operations:

- Encryption(m): $c = m^e \pmod{n}$
- Decryption(c): $m' = c^d \pmod{n} = (m^e)^d \bmod n$, $m' = m \pmod{n}$

RSA algorithm, example



Anyone who wants to talk to Bob can retrieve Bob's public key, use it to encrypt the message, and know that only holder of the corresponding private key can decrypt.

RSA algorithm in practice

In practice, key pairs are much longer.

```
$ ssh-keygen -t rsa -b 4096
```

Each key pair corresponds with an **identity**.

The two operations:

- Encryption/Decryption: Alice uses Bob's public key to encrypt, Bob uses his private key to decrypt
- Signing/Validation: Alice uses her own private key to sign, others use Alice's public key to verify

Vs symmetric encryption: more computation, but solves the problem of key transfer (and many others)

Related concepts: digital signature, certificate, public key infrastructure

To design a cryptocurrency

Imagine designing a cryptocurrency

- An account is a public/private key pair!
- If I pay someone (a public key identity), I sign with my private key: others can verify *I* made the payment, and I cannot refute later on

But how does anyone know I have enough money left to make the payment? *Double spending*

- The **centralized** way: we all agree on (and preinstall) having one party to trust, who keeps track of everyone's account balances
- The **decentralized** way: is it possible to have *all* of us keep track of account balances together?

To design a cryptocurrency - cont

Considerations:

- Distributed
- Trustless, no a-prori trust relationship established
- Consensus, everyone agrees on account balances

Why is this problem unique:

- I don't trust anyone I talk to! (think Raft, dynamo, etc) But we can still agree on something.

Distributed trustless consensus - intuition

Distributed consensus: leadership election, and **state-machine replication** (think raft)

If we can agree on the entire history of transactions, we would know if someone is trying to double spend. (think log structured merge tree (e.g. Cassandra) / journaling file system / git)

To agree on the history,

- what if we assume: there are more peers in the network who are honest, than those who are not
- we can have everyone tell everyone else their view of the entire history, and hope they converge...

Blockchain in bitcoin whitepaper

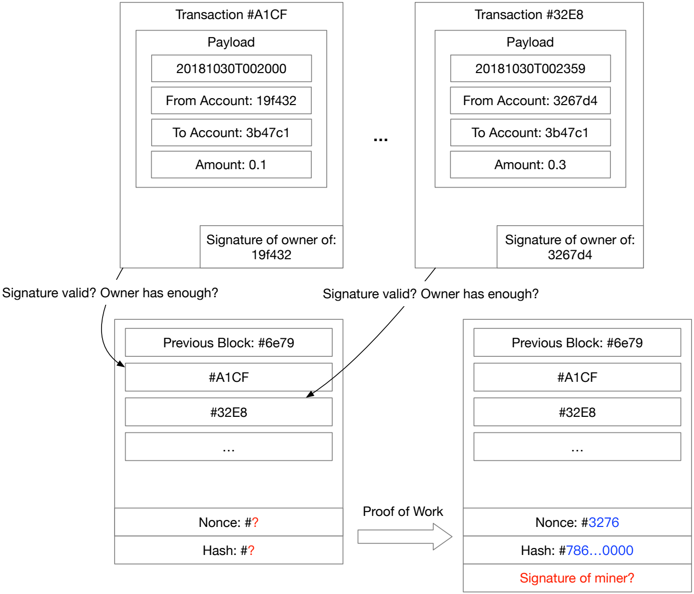
Blockchain makes a different assumption

- **honest peers own more computation power than those who are dishonest**

Consequently, if it takes computation power to grow the history, then honest peers can grow the history faster than dishonest ones

- Peers **trust the chain with the most computation power invested (i.e. the longest)**, and grow based on that
- In order to grow the history, peers perform a **proof of work**, which is computationally non-trivial

Building a block



Transaction #A1CF

Payload

20181030T002000

From Account: 19f432

To Account: 3b47c1

Amount: 0.1

Signature of owner of: 19f432

...

Transaction #32E8

Payload

20181030T002359

From Account: 3267d4

To Account: 3b47c1

Amount: 0.3

Signature of owner of: 3267d4

Signature valid? Owner has enough?

Signature valid? Owner has enough?

Previous Block: #6e79

#A1CF

#32E8

...

Nonce: #?

Hash: #?

Proof of Work

Previous Block: #6e79

#A1CF

#32E8

...

Nonce: #3276

Hash: #786...0000

Signature of miner?

Proof of work

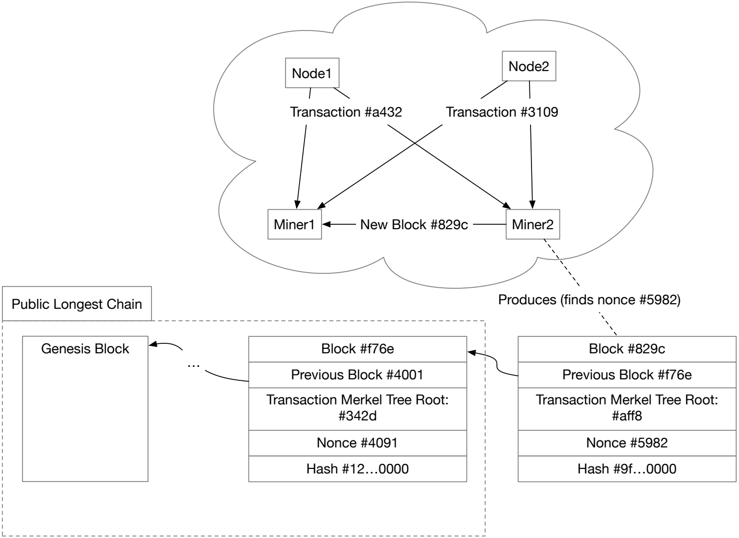
Find a nonce value to attach to a block, such that the hash of the entire block has an agreed number of 0s at its end

- A block is only considered valid and complete, if one such nonce is attached
- No known efficient algorithm exists to find one such value, (mostly) trials
- Alternatives exist

Proof-of-work "verifies" a block (in turn, grows a chain)

Peers who carry out proof-of-work are called miners.

Growing a chain



Genesis block: the start of the chain (e.g. allocating X coins to the creator)

Mining and incentive

Why should anyone verify someone else's transaction (mine)?

- Miners are awarded a fixed number of bitcoins for each block mined
- Transactions can attach a transaction fee
- Winner takes all

How are new coins introduced to the system?

- Genesis block
- Mining 'adds' new coins to the system (analogy: miners spend work to find gold nuggets to add to world gold circulation)

Operation

What if two blocks are created at the same time?

- A temporary fork. We can introduce an ordering mechanism, or miners may work on different forks, until one gets longer than the other and propagates.

As the system progresses,

- Number of 0s to satisfy proof-of-work increases (world's computation power increases)
- Mining reward **halves** (how to incentivize afterwards? Transaction fee)

Max number of coins is fixed (inflation-free!)

$$\lim_{n \to \infty} \sum_{i=1}^{n} \frac{1}{2^i} = 1$$

Tamper resistance and privacy

Temper resistance

- Assume dishonest peers work together to produce proof-of-work for a block in which a transaction source account double spends
- Dishonest peers have to grow the chain faster than the honest ones, so that the honest ones use their chain
- Subverting a previous block in the chain gets exponentially more difficult (grow everything afterwards, and faster)
- If you have this much computation power, it'd be more in your interest to mine, rather than to subvert

Privacy

- No tie between account $736f$ with a physical identity
- The network does not store anything related with physical identity

The trade-offs

Distributed and trustless come at a cost

- Global scale broadcast messages: vs unicast to a bank
- Proof-of-work: vs bank keeping records and trusting what they keep
- Transaction fee: pay to verifier, or pay to bank

But also at an advantage

- Less a-priori
- Anonymous
- Mathematically hard to tamper / counterfeit (?)

Analogy to a currency

- Durable
- Portable
- Divisible
- Fungible (like symmetric in an equivalence relation)
- Intrinsic value?

Extensions - smart contract

Summary