Lesson 1 - Blockchain Theory

Practical Details

All lessons will be conducted via zoom.
The format will usually be 45 mins of theory followed by 45 mins practical
You can work in teams

How to ask questions ?

We have channels for questions

- Sli.do : https://app.sli.do/event/mGCCEA24h1VE579HSL4mav
- Discord technical questions

Put your questions in channels rather than asking individuals

How do practicals work ?

The lessons will typically be split 50/50 into theory and practical

During the practical half of the lesson you can work on exercises and ask questions in the support channel

The tools we will be using are

- Remix
- VSCode (in Gitpod)
- Hardhat / Foundry

About us

Free Developer Workshops

- Basic
- Enterprise
- Advanced EVM
- Zero Knowledge Proofs

Business Workshops

Website : https://extropy.io

Email : info@extropy.io

Twitter : @extropy

Decentralised Systems

Problems with centralised systems

Monetary System

- Bank closure / insufficient capital reserves
- greek debt crisis in 2015 ? banks closed and people lost savings, insurance schemes meant nothing, lead to an increase in Bitcoin use in Greece
- Availability of banks
- Inflation - money supply controlled by central authority
- Merchant accounts may be shut down
- Control of money for political reasons - wikileaks funding shutdown

There are layers of access control built into our banking systems to prevent fraudulent transactions, effectively security is achieved by closing the network.

Goals of decentralisation

- Participation
- Diversity
- Conflict resolution
- Flexibility
- Moving power to the edge (user)

Typical Blockchain components

Gossip network

**

Shared public ledger

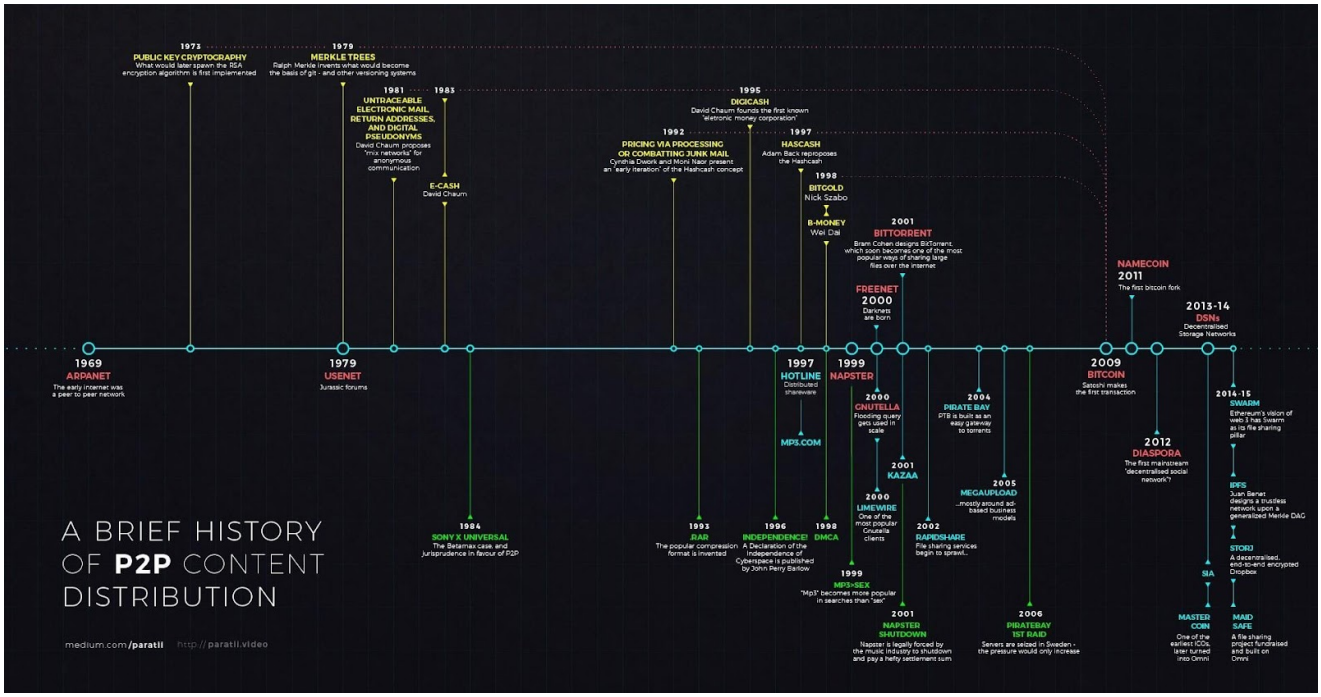
Cryptography

These components give the blockchain

- Transparency and verifiable state based on consensus
- Resilience
- Censorship resistance
- Tamper proof interactions

Timeline of Cryptographic systems

**

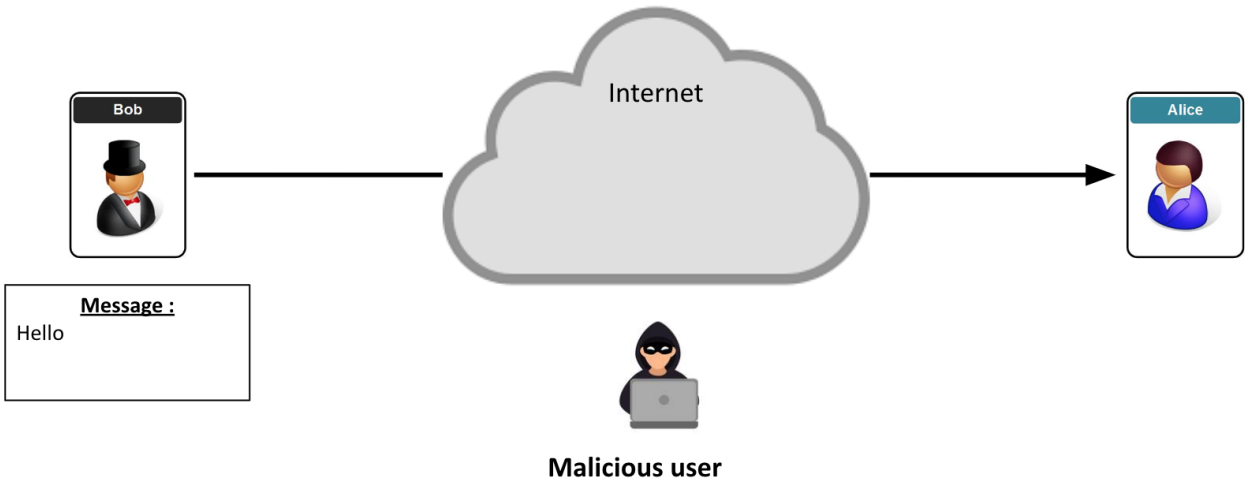


1970s

Problem = Security !

Secure Communication over Insecure Channel

Problem 1 : How do I ensure that my message has not been modified ?



1. Privacy

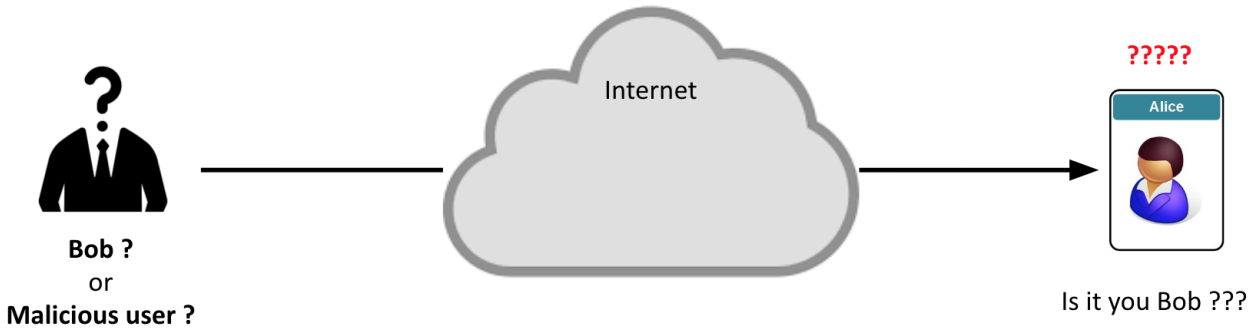
- How do I ensure that my message has not been modified ?

2. Authenticity

- How do I ensure that the message comes from a legitimate person ?

Secure Communication over Insecure Channel

Problem 2 : How do I ensure that the message comes from a legitimate person ?

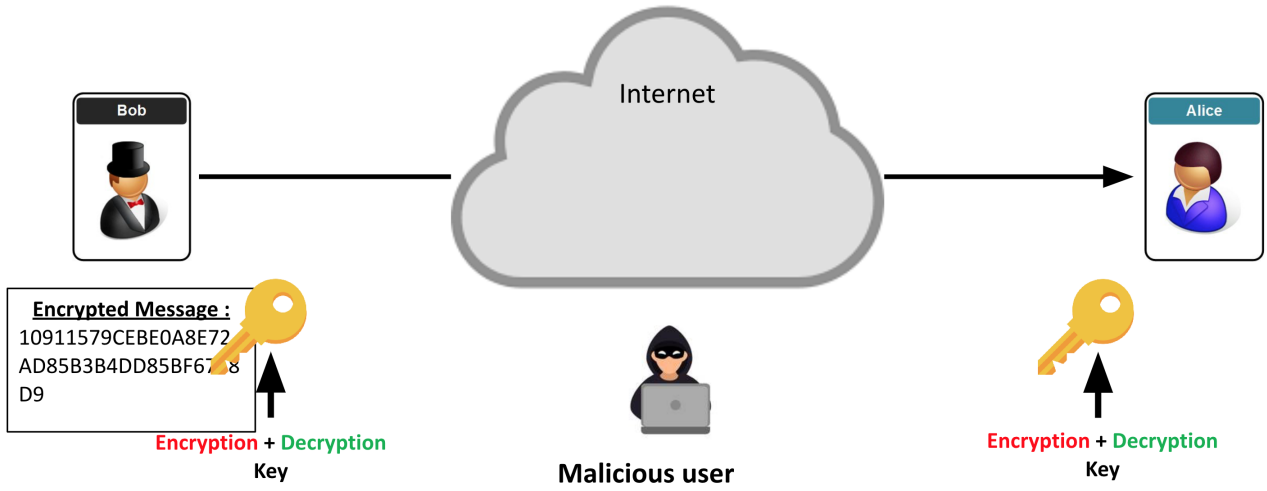


1st Solution : Symmetric Cryptography !

- Alice and Bob share the same key.
- One key for both encryption and decryption of messages

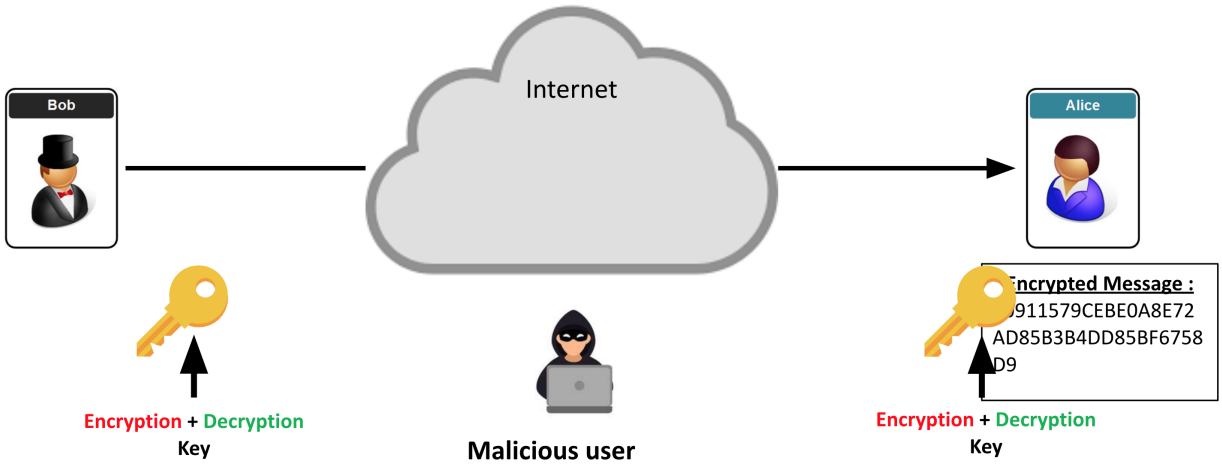
Secure Communication over Insecure Channel

Symetric Cryptography : encryption and decryption keys are the same



Secure Communication over Insecure Channel

Symetric Cryptography : encryption and decryption keys are the same



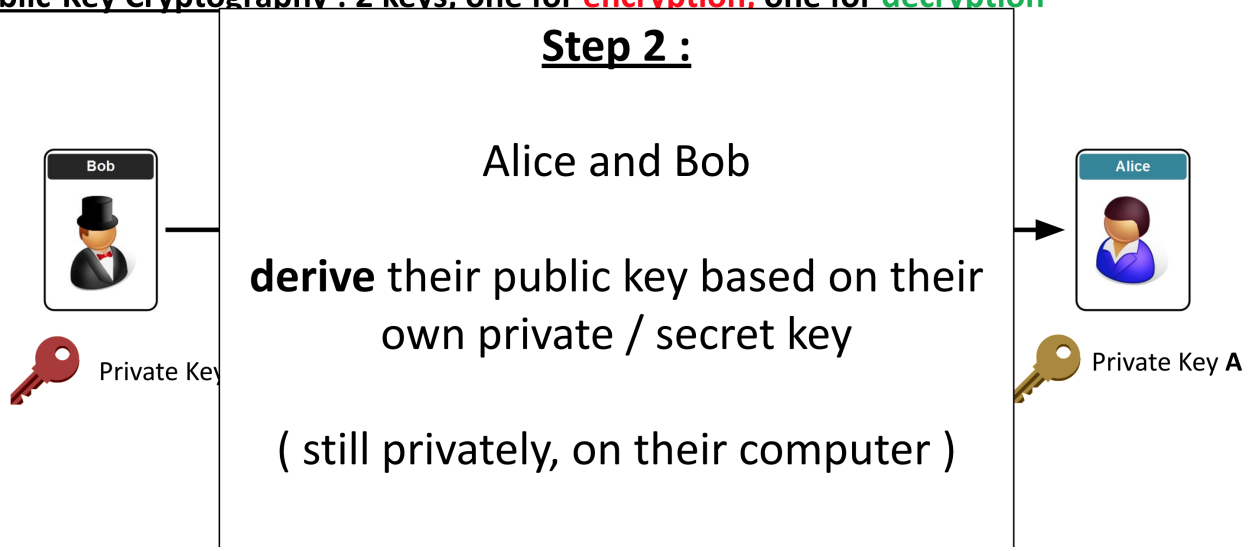
But what about key management ?

Can Alice and Bob share a key

- Without meeting
- Across a potentially hostile network

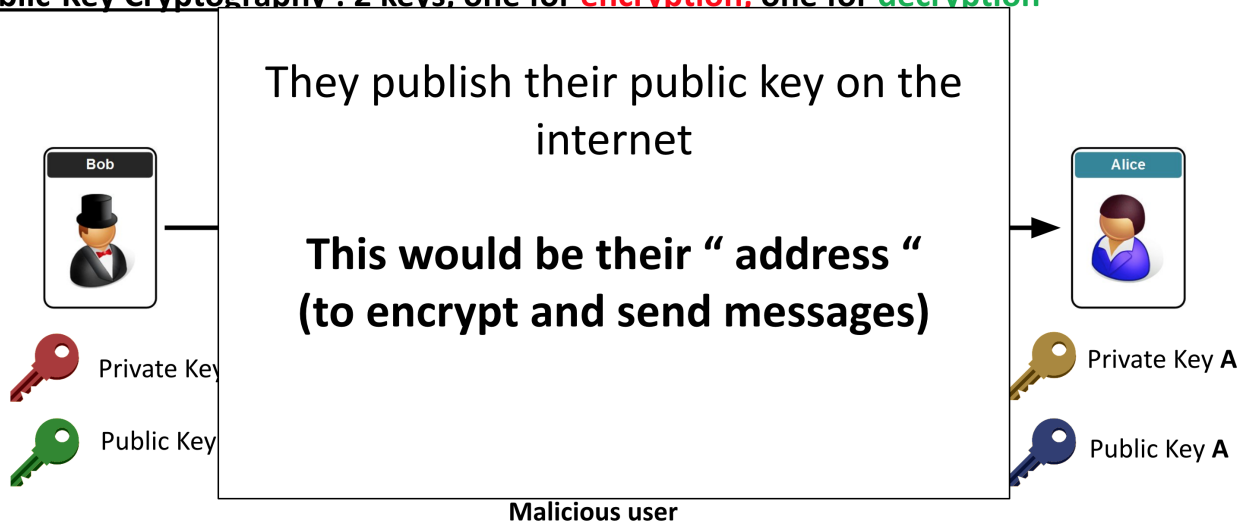
Secure Communication over Insecure Channel

Public-Key Cryptography : 2 keys, one for encryption, one for decryption



Secure Communication over Insecure Channel

Public-Key Cryptography : 2 keys, one for encryption, one for decryption



Secure Communication over Insecure Channel

Public-Key Cryptography : 2 keys, one for encryption, one for decryption

Secure Communication over Insecure Channel

Public-Key Cryptography : 2 keys, one for encryption, one for decryption

Public Key Encryption solves :

Problem 1 : Key Management

- If I use a 3rd party to share my key, do I trust him ?

Problem 2 : Integrity

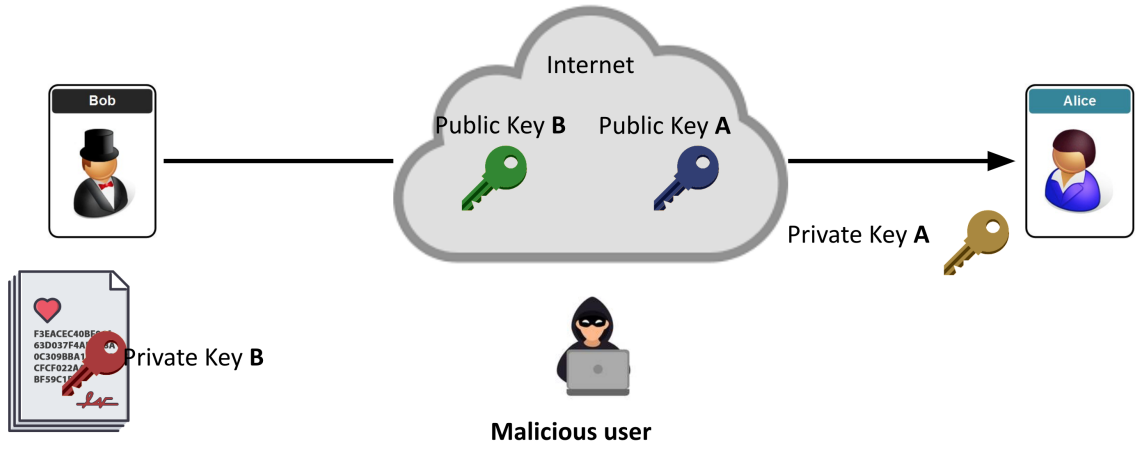
- How do I ensure that my message has not been modified ?

Problem 3 : Authenticity

- How do I ensure that the message comes from a legitimate person ?

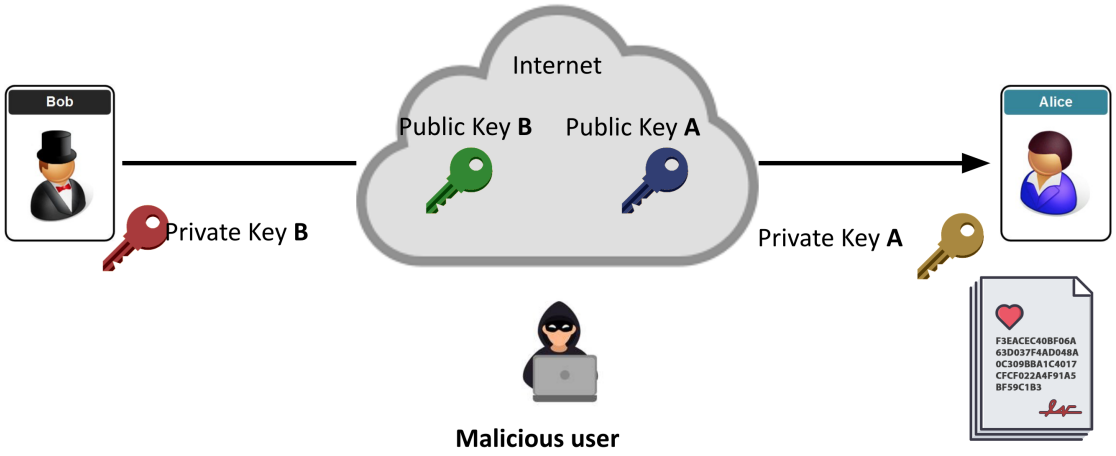
Secure Communication over Insecure Channel

Digital Signature : Encrypt > Sign > Decrypt > Verify



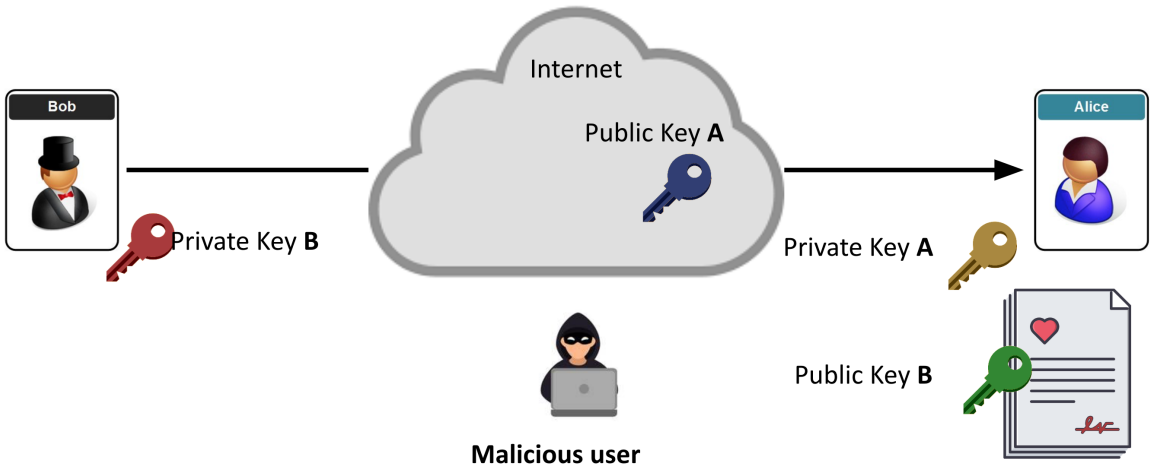
Secure Communication over Insecure Channel

Digital Signature : Encrypt > Sign > Decrypt > Verify



Secure Communication over Insecure Channel

Digital Signature : Encrypt > Sign > Decrypt > Verify



Digital Signature : 4 properties

- **Authentic** : when Alice verifies the message with Bob's public key, she know that he signed the message.
- **Unforgeable** : only Bob knows his private key.
- **Not Reusable** : the signature is a function of the document. It can't be transferred to any other document.
- **Unalterable :** if there is any alteration to the document, the signature can no longer be verified with Bob's public key.

Hash Function = Digital Fingerprint

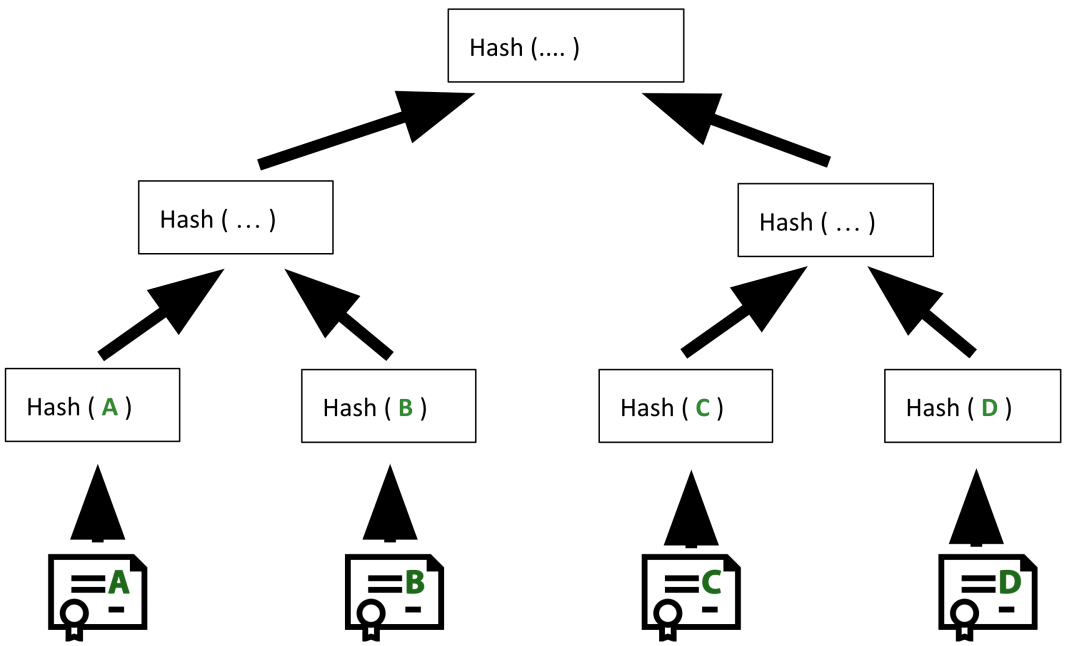
Example :

(the example uses SHA256, try here > https://bit.ly/2YYMbF8

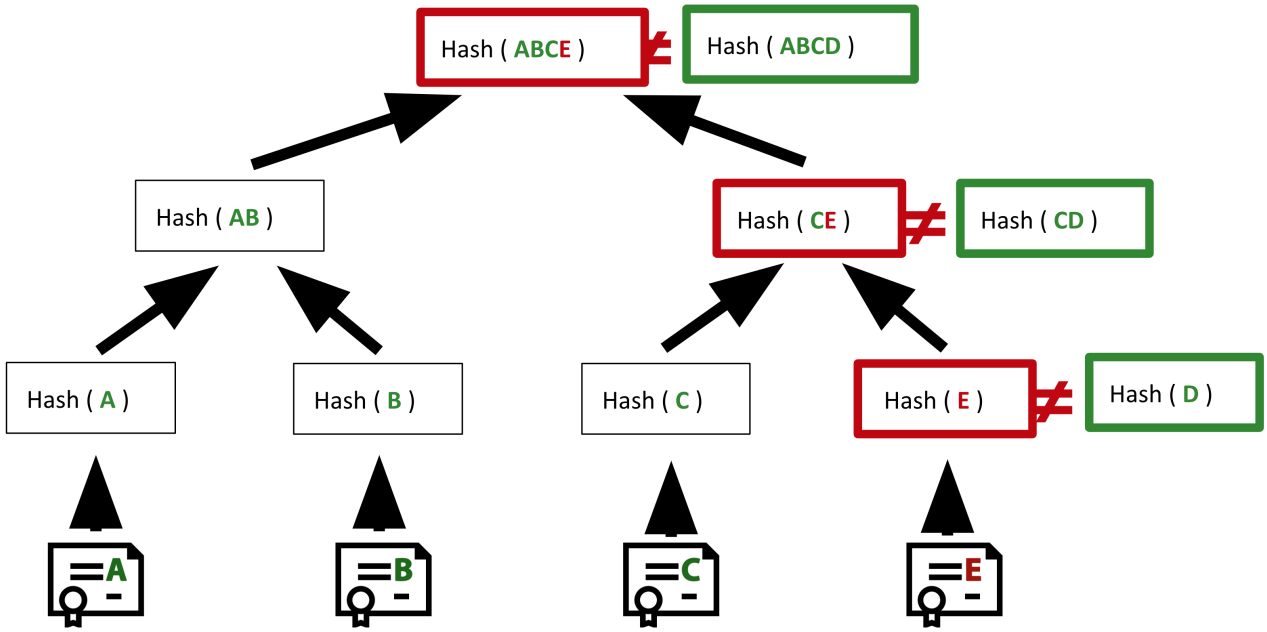
Input		Output
Hash(Loughborough)	→	acb208d3ac02ab6d5a4…
L " upper case		
Hash(loughborough)	→	1c2cc85d86f480bca5df0…
l " lower case		

A small change changes the whole digital fingerprint

Merkle Tree (the basic)



Merkle Tree (the basic)



That is the cryptographic background, how did people try to use this technology ?

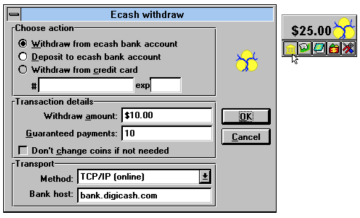
The development of

- Electronic cash
- Timestamping
- P2P Systems
- Consensus systems

Electronic cash

1980's

E-Cash



1983

Blind Signature for Untraceable Payments
David Chaum

Untraceable Electronic Cash
Chaum, Fiat and Naor

Extends to Offline payments

1988

David Chaum founds the company

Digicash

1990

Starts with online transactions

1990's

Electronic cash
TimeStamping
Consensus Algorithm

Haber & Stornetta
How to TimeStamp a Digital Document

1991

1992

Bayer, Haber & Stornetta
Improving the efficiency and reliability of digital time-stamping

Benaloh and De Mare
Efficient Broadcast TimeStamping

Dwork and Naor
Pricing via Processing or Combatting Junk Mail

1993

1996

How to Make a Mint: the Cryptography of Anonymous Electronic Cash - NSA

HashCash
Adam Back

Haber & Stornetta
Secure Name for BitStrings

1997

1998

B-Money
Wei Dai

Bit-Gold
Nick Szabo

Auditable, Anonymous Electronic Cash
Sander & Ta-Shma

Massias, Avila and Quisquater
Design of a Secure Timestamping Service with minimal trust requirement

1999

Peer-to-Peer networks emerge

SoulSeek

Freenet

GNutella

2000 **2001** **2004**

BitTorrent

eDonkey

FastTrack / KaZaa

Further Attempts at Electronic Cash

“the one thing that’s missing is a reliable e-cash, whereby on the internet you can transfer funds from A to B without A knowing B or B knowing A” - Milton Friedman 1999

1998 - b-money - Wei Dai (http://www.weidai.com/bmoney.txt)
1998 - Bit Gold - Nick Szabo (https://nakamotoinstitute.org/bit-gold/)

Satoshi Nakamoto is the name used by the presumed pseudonymous person or persons who developed bitcoin, authored the bitcoin white paper, and created and deployed bitcoin's original reference implementation

Early Bitcoin History

August 2008 - domain name bitcoin.org registered
October 2008 - A Peer-to-Peer Electronic Cash System posted to a cryptography mailing list
January 2009 - Software implementation released as open source

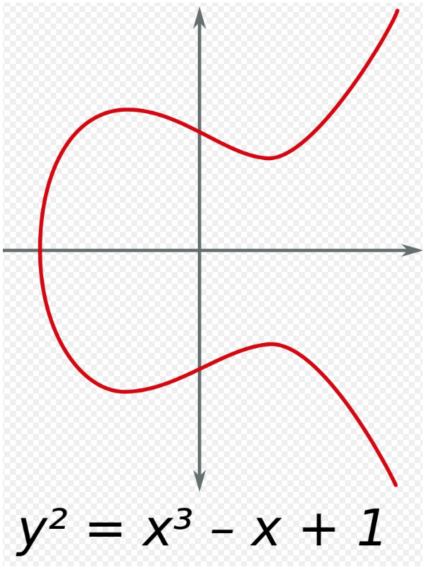
2010, the first known commercial transaction using bitcoin occurred when programmer Laszlo Hanyecz bought two Papa John's pizzas for 10,000 BTC

Blockchain events since 2009

2014 - Ethereum created
2017 - ICO Boom / Alternatives to Ethereum
2018 - Crypto winter
2020 - DeFi summer
2021 - Rise of NFTs / Gaming
2022 - Ethereum moves to Proof of Stake

Key Cryptography in Ethereum

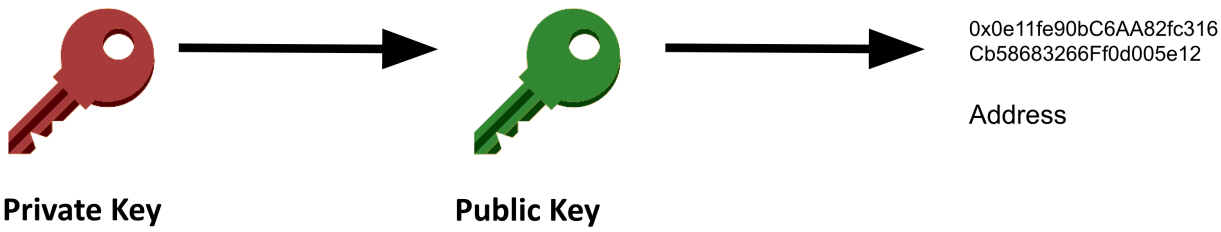
Key Cryptography used in Ethereum



Ethereum uses ECDSA (Elliptic Curve Digital Signature Algorithm)
It uses the SECP256k1 curve.

Elliptic curves have a shorter key length for the same level of security as RSA

Keys and addresses in Ethereum



For a given private key, p_r, the Ethereum address $A(p_r)$ (a 160-bit value) to which it corresponds is defined as the rightmost 160-bits of the Keccak-256 hash of the corresponding ECDSA public key:

Tool to visualise blocks and hashes

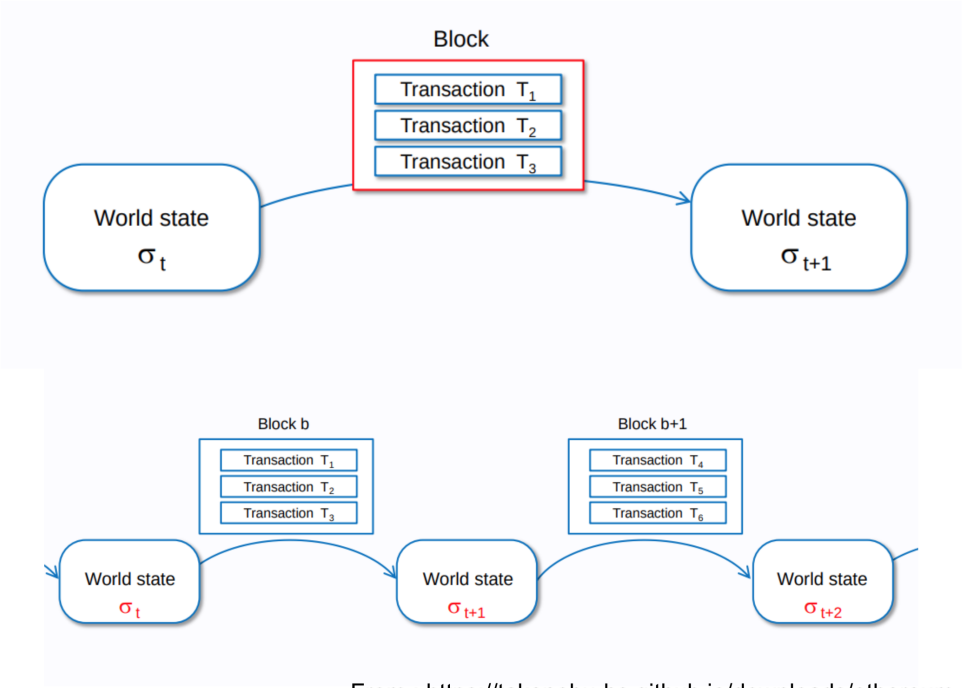
Blockchain Tool

Blockchain components in more detail

- A peer-to-peer (P2P) network connecting participants and propagating transactions and blocks of verified transactions, based on a standardized "gossip" protocol
- Messages, in the form of transactions, representing state transitions
- A set of consensus rules, governing what constitutes a transaction and what makes for a valid state transition
- A state machine that processes transactions according to the consensus rules
- A chain of cryptographically secured blocks that acts as a journal of all the verified and accepted state transitions

- A consensus algorithm that decentralizes control over the blockchain, by forcing participants to cooperate in the enforcement of the consensus rules
- A game-theoretically sound incentivization scheme (e.g., proof-of-work costs plus block rewards) to economically secure the state machine in an open environment
- One or more open source software implementations of the above ("clients")

Blockchain as a state machine



From : https://takenobu-hs.github.io/downloads/ethereum_evm_illustrated.pdf

Blockchain Network

From : Bitcoin book

Bitcoin is structured as a peer-to-peer network architecture on top of the internet. The term peer-to-peer, or P2P, means that the computers that participate in the network are peers to each other, that they are all equal, that there are no "special" nodes, and that all nodes share the burden of providing network services.

The network nodes interconnect in a mesh network with a "flat" topology.

There is no server, no centralized service, and no hierarchy within the network.

Nodes typically

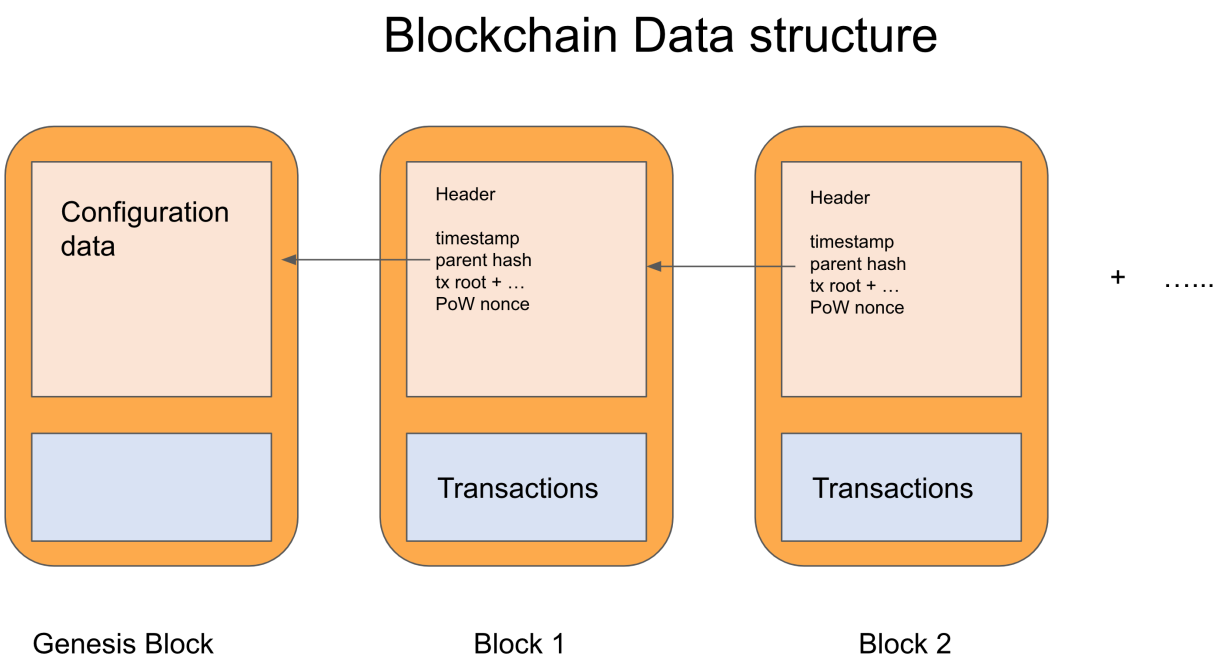
- Accept and transmit transactions (if valid)

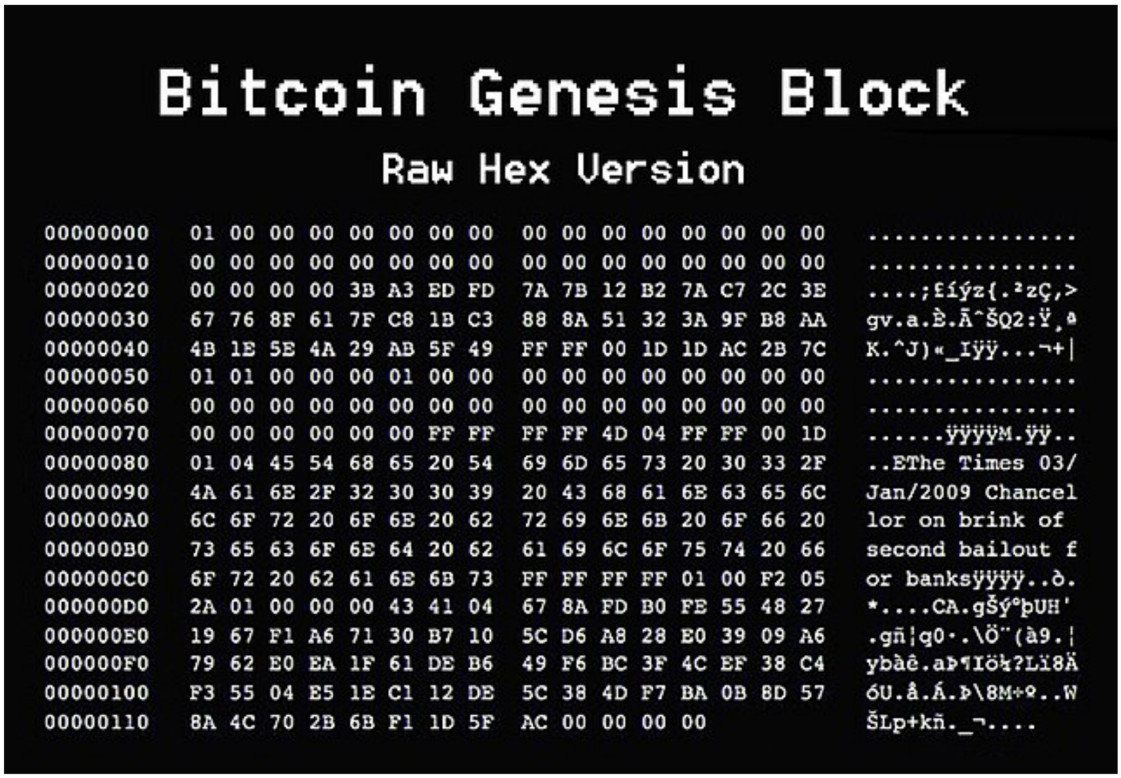
- they keep a mempool of pending transactions

- Provide network discovery and routing functions
- the connections are not based on geographical proximity but proximity in a hash table
- connections to misbehaving nodes will be dropped
- Accept blocks and update their ledger

Node discovery

- Via DNS seed nodes
- Via locally stored list





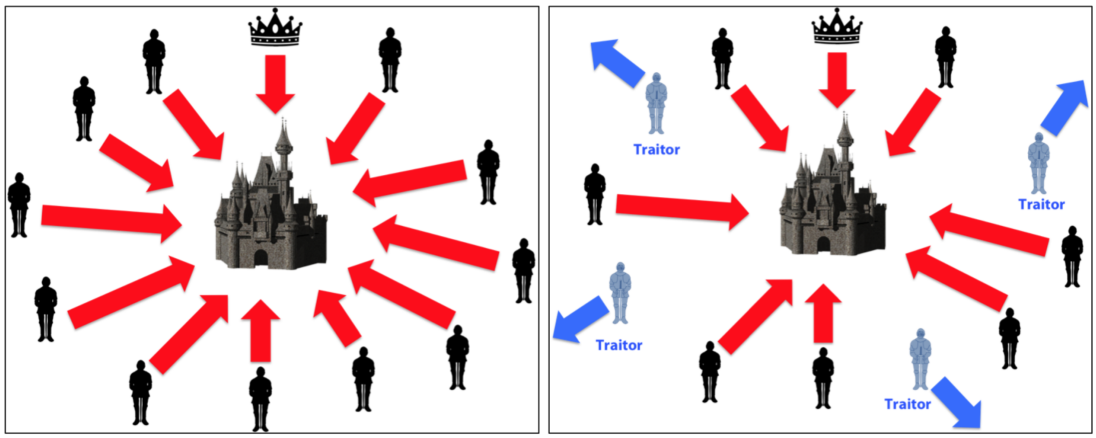
By MikeG001 - Own work, CC BY-SA 4.0,

Consensus in systems

How can we agree on the state of a system ?

Byzantine Fault tolerance

Byzantine fault tolerance (BFT) is the dependability of a fault-tolerant computer system to such conditions where components may fail and there is imperfect information on whether a component has failed.



Coordinated Attack Leading to Victory **Uncoordinated Attack Leading to Defeat**

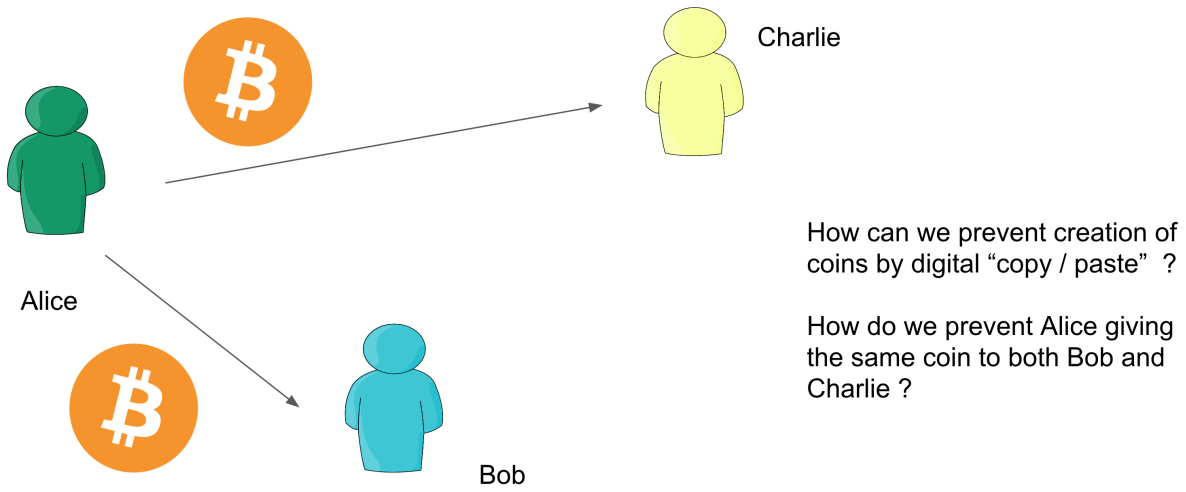
Byzantine Generals' Problem, Image by Debraj Ghosh

The Double Spending Problem

"The double spending problem is a potential flaw in a cryptocurrency or other digital cash scheme whereby the same single digital token can be spent more than once, and this is possible because a digital token consists of a digital file that can be duplicated or falsified."

https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3090174

The double spend problem



Consensus Mechanisms

Strictly speaking there are 2 parts, sybil resistance and consensus, although most people talk of both together.

Examples

Practical Byzantine Fault Tolerance (pBFT) Castro and Liskov 1999
Nakamoto Consensus (Proof of Work and Nakamoto Consensus) 2008

There are now many “Proof of ….

Stake / Authority / History / Burn / Elapsed Time / Spacetime ….”

Proof of Kernel Work (Extropy and others)**

Summary

Blockchains should be seen in the context of decentralisation
Cryptographic techniques have been crucial for solving the problems in implementing decentralised systems
Early electronic cash systems came part way to a practical implementation that could solve the double spend problem and achieve consensus across an open network.
Ethereum successfuly transitioned from Proof of Work to Proof of Stake in 2022