Ethereum whitepaper

<u>Next generation smart contract & decentralised application platform</u>, V. Buterin

- Satoshi solved distributed consensus by requiring proof of work from participants, as well as by implementing a public ledger
- Formally, a ledger can be thought of as a set of state transitions.

$$\text{APPLY}(S, TX) \rightarrow S' \text{ or ERROR}$$

transaction converts old state into new state or throws error

- the state is the collection of UTXO (unspent transaction outputs), with each UTXO having a denomination and owner.
- a transaction maps 1+ inputs (with sigs) to 1+ outputs

```
for each input in TX:
   if input UTXO not in S :  ERROR   ← can't send coins that don't exist
   if sig ≠ UTXO owner :  ERROR   ← can't send others' coins
If Σ inputs < Σ outputs : ERROR   ← can't create value
return S − (input UTXO) + (output UTXO)
```

- Miners create blocks, which are valid if
  - previous block exists and is valid
  - timestamp > timestamp(previous)
  - valid proof of work
  - all transactions are valid
- An Eve with >51% of the network's computational power could extend her own malicious forks.
- The blockchain history is stored in a Merkle tree
  - a node can compare a block header with a small part of the tree, which reduces required storage size
  - the simplified payment verification (SPV) protocol allows for 'light nodes' to verify POW on block headers and only download a small part of the tree.

- To build a consensus protocol, you can either build a new network or build it on top of the bitcoin blockchain
  - the former is difficult to implement and most applications will be too small
  - the latter is not scalable, as you cannot have 'light nodes'
- Bitcoin does have a scripting language that can:
  - create 'safety deposit boxes' that requires an additional key to open
  - implement merchant escrow
  - support cross-cryptocurrency exchange

  But it has important limitations:
  - not Turing-complete
  - lack of state: UTXOs are spent or unspent, which limits possibilities
  - Blockchain-blindness: a UTXO cannot see the nonce and prevHash, which could be good sources of randomness.

Ethereum

- Blockchain with a built-in Turing complete programming language
- The state in Ethereum is made up of accounts, with each account having a 20B address and containing four fields:
  - nonce
  - ether balance ← ether is the digital currency used to pay transaction fees
  - contract code
  - storage (empty by default)
- Accounts are either externally owned (controlled by private keys) else they are contract accounts (controlled by their contract code).

- The Eth equivalent of a Btc transaction is a message:
  - can be created externally or by a contract
  - can explicitly contain data
  - if the recipient is a contract account, they can return a response → messages can be used as functions
- 'Transactions' in Eth refers to the signed data package that stores the message, the recipient, the signature, the quantity of eth, data, and STARTGAS and GASPRICE

to prevent infinite loops, you must pay a certain amount to the miner per computation. The limit of how much you will pay is STARTGAS. If a transaction runs out of gas, state changes revert except for gas fees. Spare gas is returned to the sender.

- Contracts in Eth are created with a different transaction format.
- Contracts are first class citizens, capable of doing anything that an external individual can.

APPLY(S, TX) → S' works as follows:

1. Check if tx is well-formed with a valid sig
2. Subtract STARTGAS x GASPRICE from sender and increment the sender's nonce
3. Initialise GAS = STARTGAS, and subtract a certain amount of gas per byte to pay for the size of the tx.
4. Transfer the tx value to the receiving account. If it is a contract, run the contract.
5. Minor collects the fees.

Code execution

- The basis of Eth is Ethereum Virtual Machine (EVM) code
- Code is an infinite loop (incrementing a counter) until STOP or RETURN is seen.

- Each byte represents an operation. These operations can access:
  - the stack (32B)
  - memory, an infinitely expandable byte array } reset after computation ends.
  - the contract's storage, a key/value store where any item can be 32B
  - value, sender, data, block-header
- The code can also return a byte array ← data or more operations.

The Ethereum Blockchain

- In addition to the transactions, blocks also contain:
  - the most recent state ← not inefficient because ~~only a small part~~ will change between transactions
  - the block number and difficulty.

The block validation algorithm:

1. Check if the previous block exists and is valid
2. Check the timestamp, block number, difficulty, tx root, gas limit
3. Check the PoW
4. Set S[0] := STATE_ROOT of the previous block. ← no need to store blockchain history
5. APPLY(S[i], TX[i]) for i = 0, 1, ... n-1. S_FINAL := S[n] ← + miner reward
6. Check if S_FINAL is the same as the system's STATE_ROOT

Applications

- Token systems
- Financial derivatives, which require a data feed contract that can be pinged when needed.
- 'Decentralised dropbox': split the file and have your contract pay them as long as they can prove that they have the file.

- Decentralised Autonomous Organisations/corporations/communities (DAOs)
  - only a 67% majority of members can move funds / modify code.
  - there can be dividend-receiving shareholders and tradeable shares.
  - voting and liquid vote-delegation
- Decentralised data feeds: N parties input a datum, and everyone between the 25th-75th percentile gets a reward.
- Cloud computing: pay others to compute, with spotchecks built in.
- P2P gambling