Blockchains

How crypto-currencies work.

Luke Anderson

luke@lukeanderson.com.au

3rd May 2019

University Of Sydney

Overview

Crypto-Bulletin

Crypto-Bulletin

Atlassian Confluence under botnet attack

https://www.itnews.com.au/news/atlassian-confluence-under-botnet-attack-524349

Microsoft Security Risk Detection: 0day in VeryPDF Reader (Part 1)

https://www.vdalabs.com/2019/04/25/microsoft-security-risk-detection-0day-in-verypdf-reader-part-1/

A Programmer Solved a 20-Year-Old, Forgotten Crypto Puzzle

https://www.wired.com/story/a-programmer-solved-a-20-year-old-forgotten-crypto-puzzle/

New Jailbreak release for iOS 12

https://chimera.sh/

Introduction

Currencies: Desiderata

Suppose we are designing a currency. What properties does it need to have?

Divisibility: A unit of currency should be able to be subdivided into units with equal buying power. Gold is divisible, an iPod is not.

Fungibility: Individual units of currency can be exchanged for each other. Every $10 note is the same, while diamonds are different.

Scarcity: There should be a reasonable restriction on the projected availability of the currency.

Recognisability: It should not be difficult to verify that a piece of currency is genuine.

Limitations of Centralised Digital Currency

Example: Alice has $100 in her PayPal account, and wants to buy some item off Bob for $25.

1. Bob asks for payment from Alice.
2. Alice speaks to the PayPal server and asks to transfer $25 to Bob.
3. Alice tells Bob that the transaction has been processed.
4. Bob checks with the PayPal server and confirms he is now $25 richer.

Limitations of Centralised Digital Currency

Continuing example: Alice has $100 in her PayPal account, and wants to buy some item off Bob for $25.

Advantages:

- Transactions require minimal work from clients.
- Transactions can be reversed, in the case of fraudulent transactions.
- Transactions are secure, and double spending or cheating cannot occur.

Disadvantages:

- The PayPal servers are a single point of failure.
- PayPal can shift or move money at their discretion.
- Alice and Bob can't perform a transaction without the PayPal server.

Bitcoin

First, there was Bitcoin (BTC)

The first successful "Crypto Currency" payment system, and the first widespread implementation of a Blockchain technology.

- Published 2009 by Satoshi Nakamoto[1]
- Open-source software
- Peer-to-peer
- Worth real money!
  - Now:
    $7,550 AUD / $5,380 USD / $95b USD

  See https://blockchain.info/charts
- Trade via Exchanges
  - BTC-e
  - CoinJar
  - Mt. Gox (dead)

[1] A pseudonym for an anonymous person or group.

Overview of Bitcoin

Bitcoin uses public key encryption to secure transactions.

- The *public key* is like a bank account number.
- The *private key* is like a PIN / password.

A *blockchain* takes the place of a central server.

- Transactions are announced to the **peer-to-peer network**.
- **All transactions are visible** to nodes in the network.

Bitcoin *miners* are rewarded to operating the blockchain.

- The creator of the next block is awarded some newly minted Bitcoins.
- The creator of the next block is awarded the transaction fees for transactions processed during the last time period.

Beginnings of Bitcoin

A paper published in November 2008 titled *"Bitcoin: A Peer-to-Peer Electronic Cash System"* by the pseudonymous author Satoshi Nakamoto outlined the design of Bitcoin, and the first (open-source) Bitcoin client was released in January 2009.

Nobody knows who Satoshi is, all we know is that they

1. Created the Bitcoin client.
2. Mined many of the first Bitcoins (now worth millions of dollars).
3. Disappeared.

Bitcoin has proven surprisingly strong, both the client program and the protocol itself.

Properties of Bitcoins

Recall: Currency needs to be *scarce* and *divisible*.

Scarcity:

- There will be at most 21 million Bitcoins (ever).
- They will be released in a predictable fashion.

Divisibility

- Each Bitcoin is divisible 100 million smaller parts, called "Satoshis".
- 1 BTC $= 10^8$ Satoshis.

What is in a Bitcoin wallet?

Create a public/private keypair.

Public: The public key is similar to a bank account number: giving someone this number allows them to transfer Bitcoin to you. This used to create a Bitcoin *address*.

Private: The private key allows you to transfer money *away* from the corresponding address.

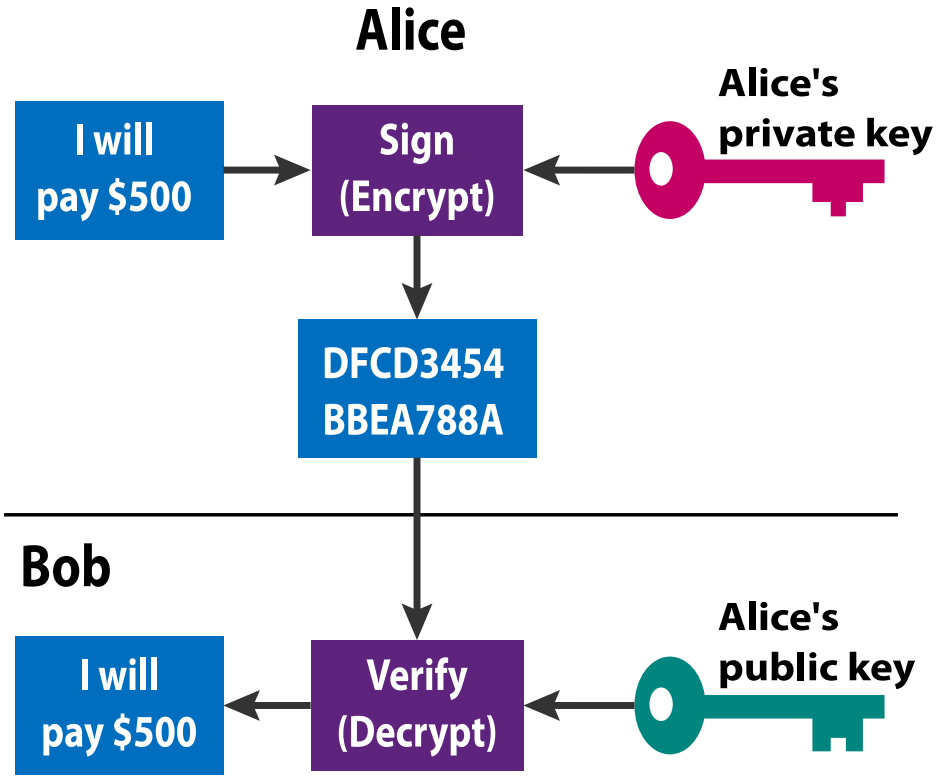
Whilst it is possible to use a single address forever, it's very common to use many addresses, even a unique address per transaction.

Address (Public key)	Bitcoin Balance
`1CkH8epnCee2jSnoYKVf2no8564LygpZcr`	1.5027
`1Lm9AuUUcazH54qFFW1Rt3V35mNvUCVFb1`	17.3723
`1D8L2KPG2U8mUqu6seE1GrYCRw2tkCxBHR`	0.2
`1NbLhL5xGS1YF8LEcXo588EPWSswgsvizb`	23.5643

Public-key Crypto & Signatures

An encryption & decryption key pair are generated.

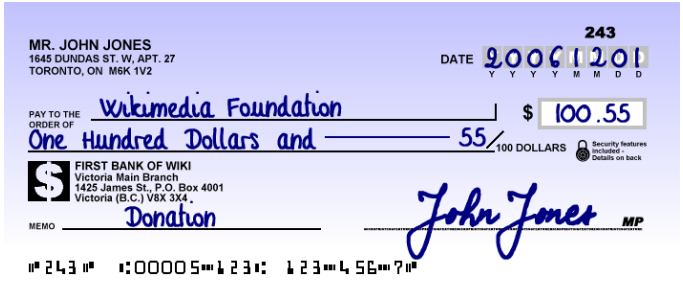
- **Blockchain address is public key.**
  Anyone can verify your signature with the public key
- **Blockchain wallet is private key**
  Only you can sign a transaction with your private key



Bitcoin Transactions

Bitcoin
transactions are like cheques.

- Sender
- Recipient
- Amount
- Signature



We don't accept cheques because
they can't be trusted.

One must *submit it to the bank* (a *central* authority) and wait for it to clear before accepting you have the money.

In Bitcoin, the sender submits a *transaction to the network* (a *distributed* authority).

If valid, the network will agree that the receiver is the new owner.

The Blockchain

The payee of a transaction must be able to "prove" that the previous owners of the bitcoin did not **double spend** it at any point.

In bitcoin, this is achieved by making everyone in the network aware *of all previous transactions*.

In order to accomplish this without a trusted third party,

1. Transactions must be publicly announced.
2. All participants must agree on a single history for the order of transactions.

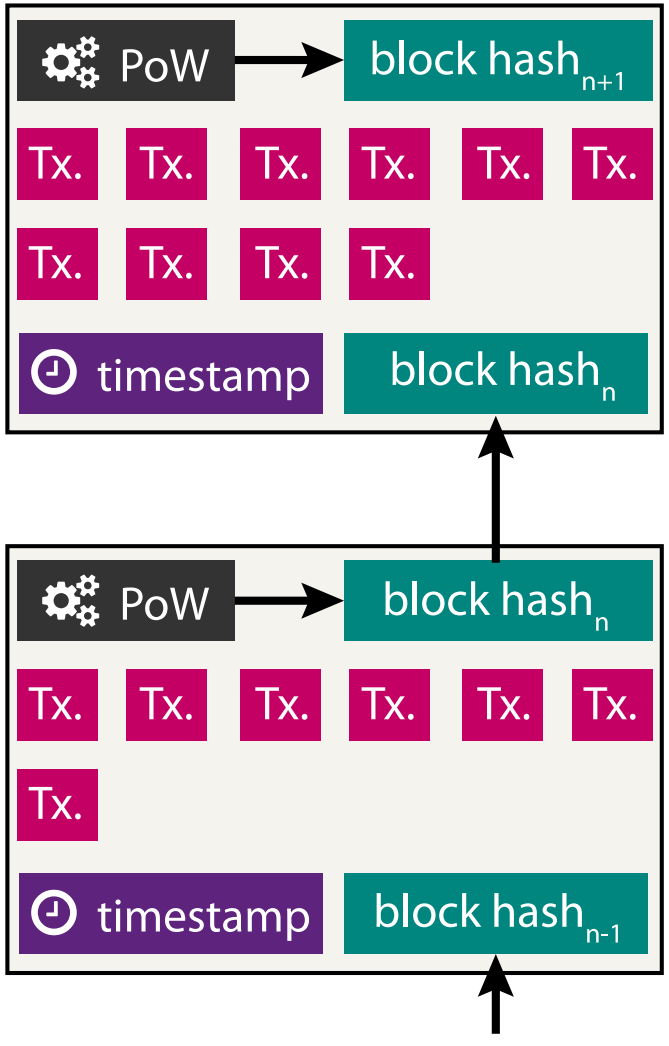
What exactly is a blockchain?

The concept of a *blockchain* incorporates a series of interdependant *blocks*, which store a consistent history of information.

Each block contains:

- Tx.: A set of transactions.
- **block hash**: block identifiers, linking them together.
- 🕘: time the block was "completed".
- ⚙: Proof of Work – what **miners** do.

Data is **appended only** – with a new block.



Blockchain: Desiderata

- **The blockchain must definitively record all transactions**
  It must show the order in which these transactions occurred, hence it must implement some form of timestamping that is agreed upon by all observers.
- **The blockchain must be difficult to modify**
  Specifically, it should be difficult to modify an existing blockchain such that past transactions may be modified, added, or removed.

The blockchain achieves both these goals by segregating work into chunks, called *blocks*, which are processed on average once every 10 minutes. This time limit, and the difficulty of modifying previous blocks, is accomplished via a *proof-of-work function*.

Blockchain: Timestamp Server

The blockchain can be thought of as implementing a *timestamp server*, which takes a group of items from timestep n, and combines them into a block B_{n}.

The hash of current block, H_{n}, is a function of both the block's contents and the previous hash $\mathrm{H}_{\mathrm{n}-1}$.

$$\mathrm{H}_{\mathrm{n}} = \mathsf{Hash}(\mathrm{H}_{\mathrm{n}-1} \parallel \mathrm{B}_{\mathrm{n}})$$

Thus, to modify a previous item, all hashes following the modification must be recalculated.

Blockchain: Proof-of-Work

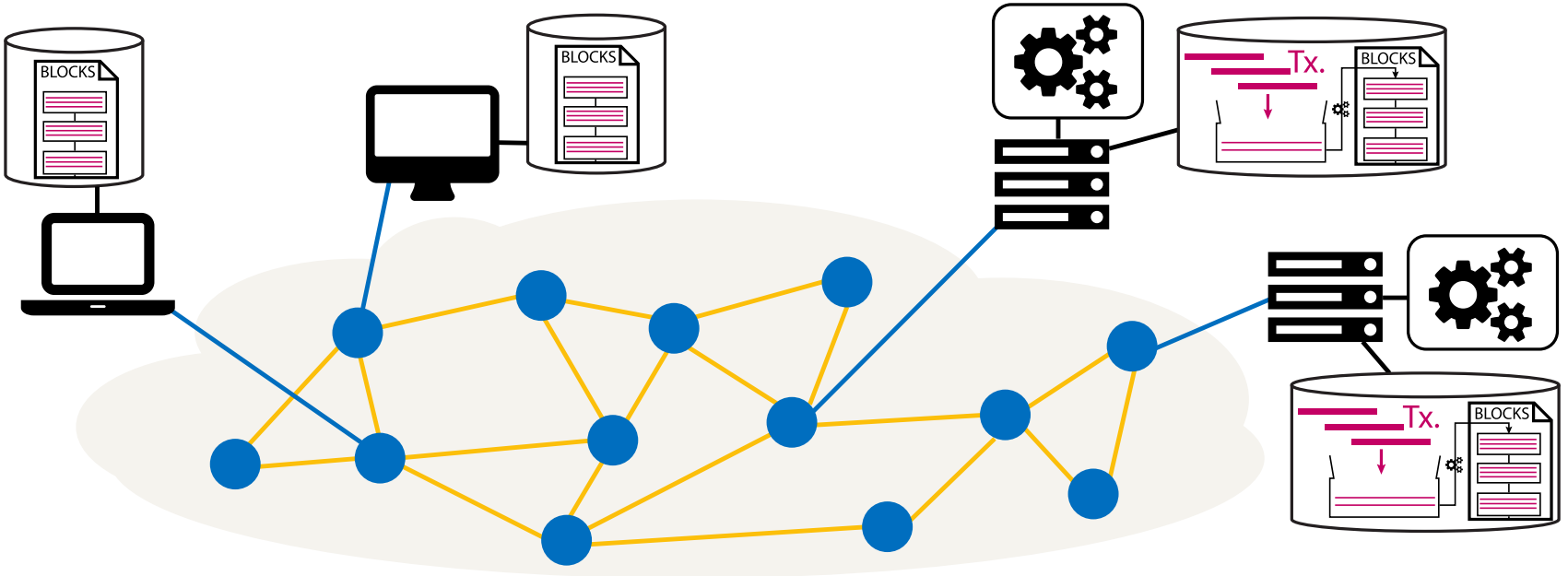
Pure hashing is fast (in fact, most hash algorithms are *designed* to be fast).

The proof-of-work involves making the hashing more time consuming, by including a nonce in the block $\mathrm{B_n}$.

The nonce has to be changed until the hash $\mathrm{Hash}(\mathrm{H_{n-1}} \parallel \mathrm{B_n})$ starts with a set number of zero bits.

Hash Format	XXXXXXXXXXXXX
Trivial	00XXXXXXXXXXX
Easy	0000XXXXXXXXX
Harder	000000XXXXXXX

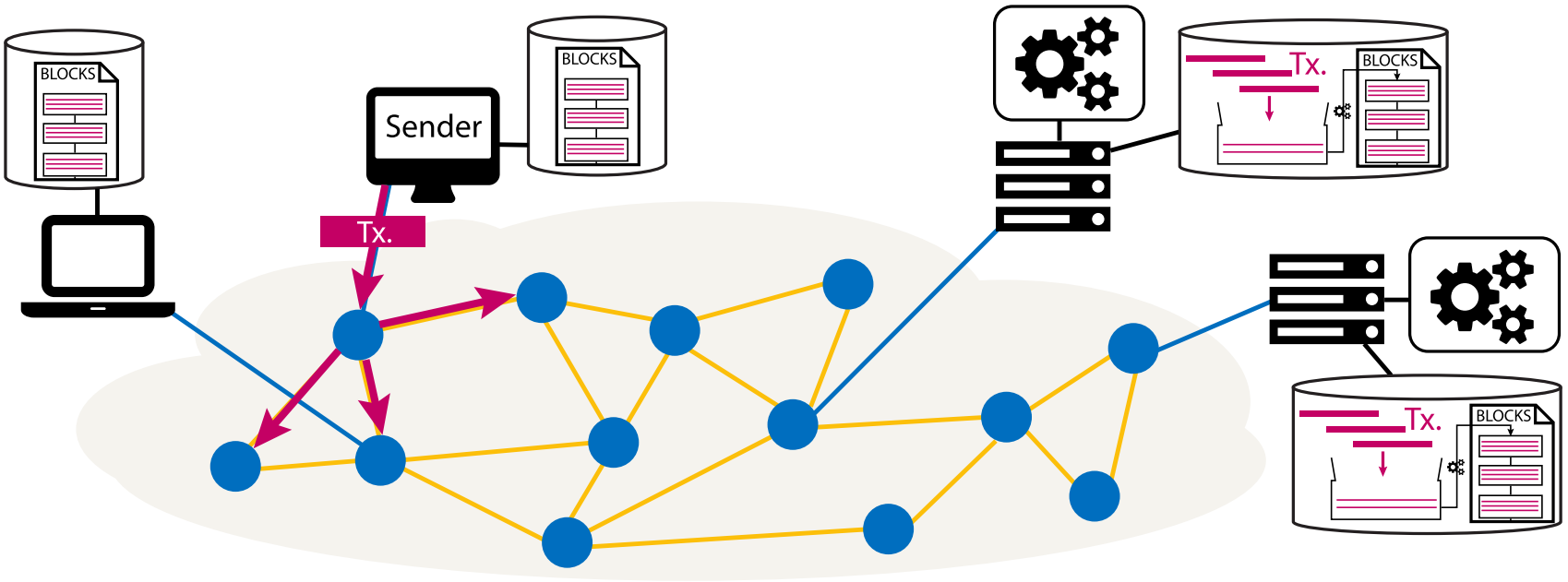
The Peer-to-Peer Network



All Bitcoin clients are sharing the latest block information via a P2P network.

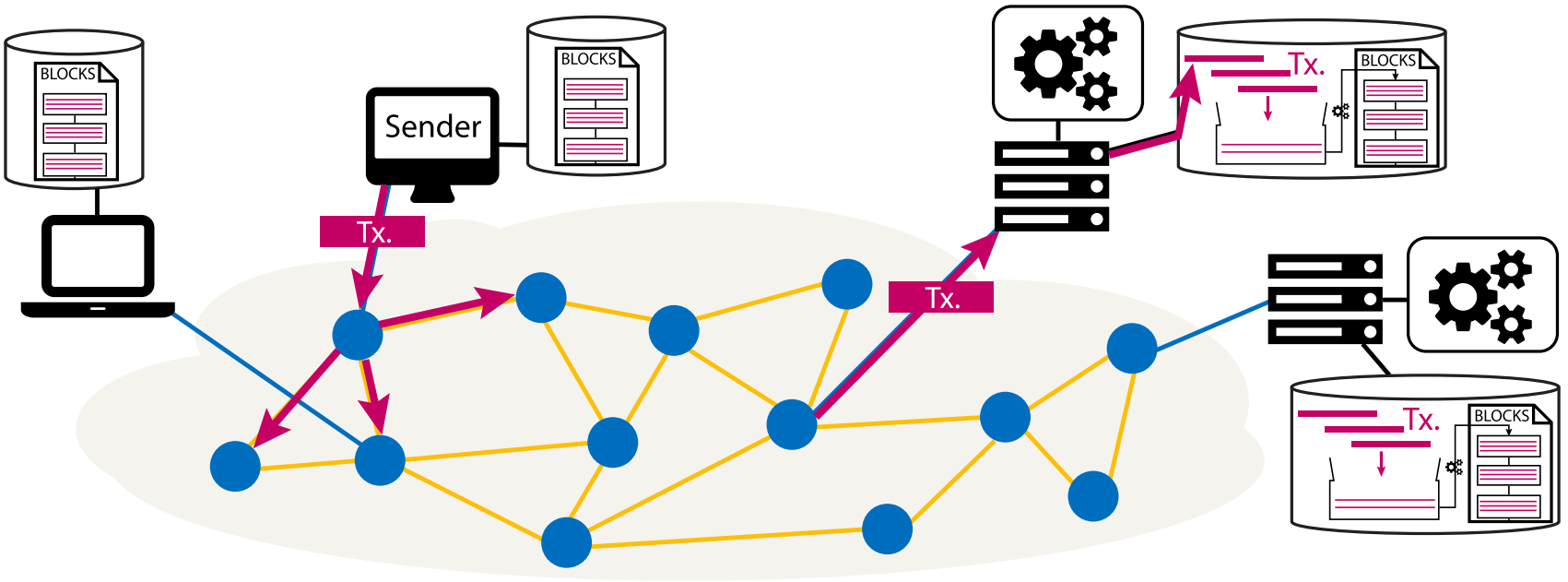
Some of those clients also *mine* blocks (⚙).
These *miners* are resposible for including transactions (Tx.)
New Tx. are announced on the network by clients and included in new blocks.

Step 1: Sending a Transaction



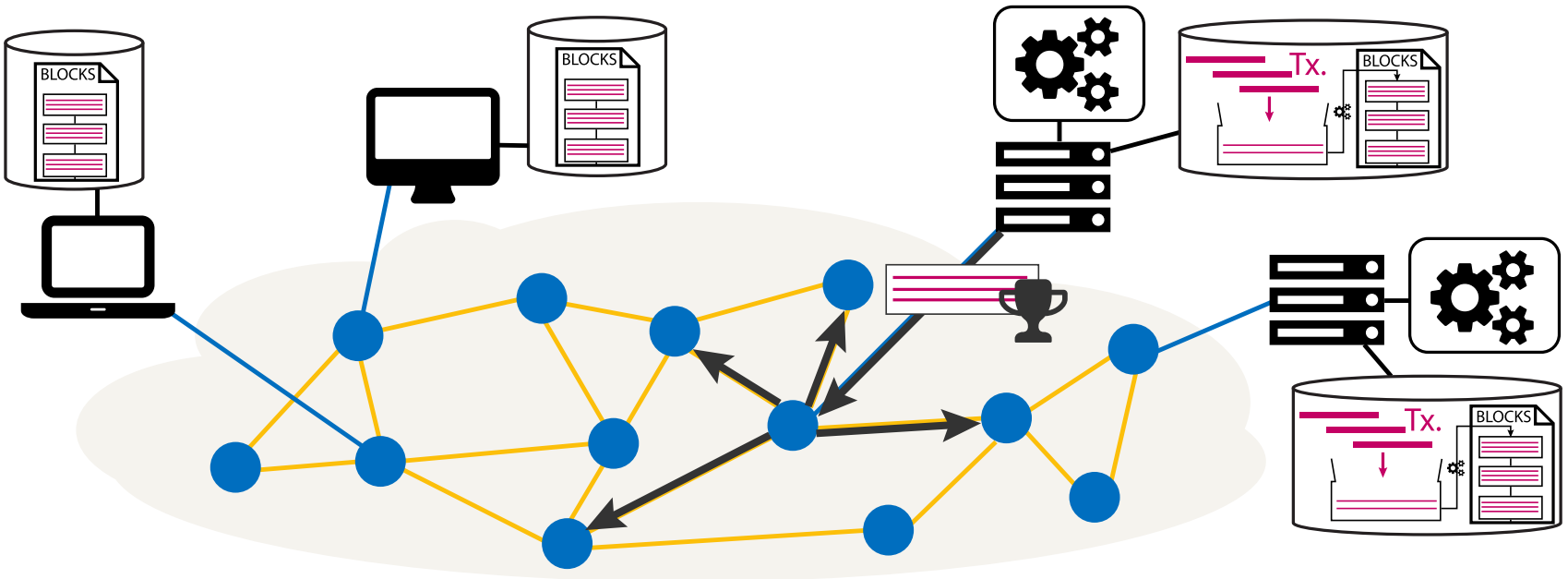
A sender authors a transaction Tx. and broadcasts it to the network.

Step 2: Miners Include Transaction



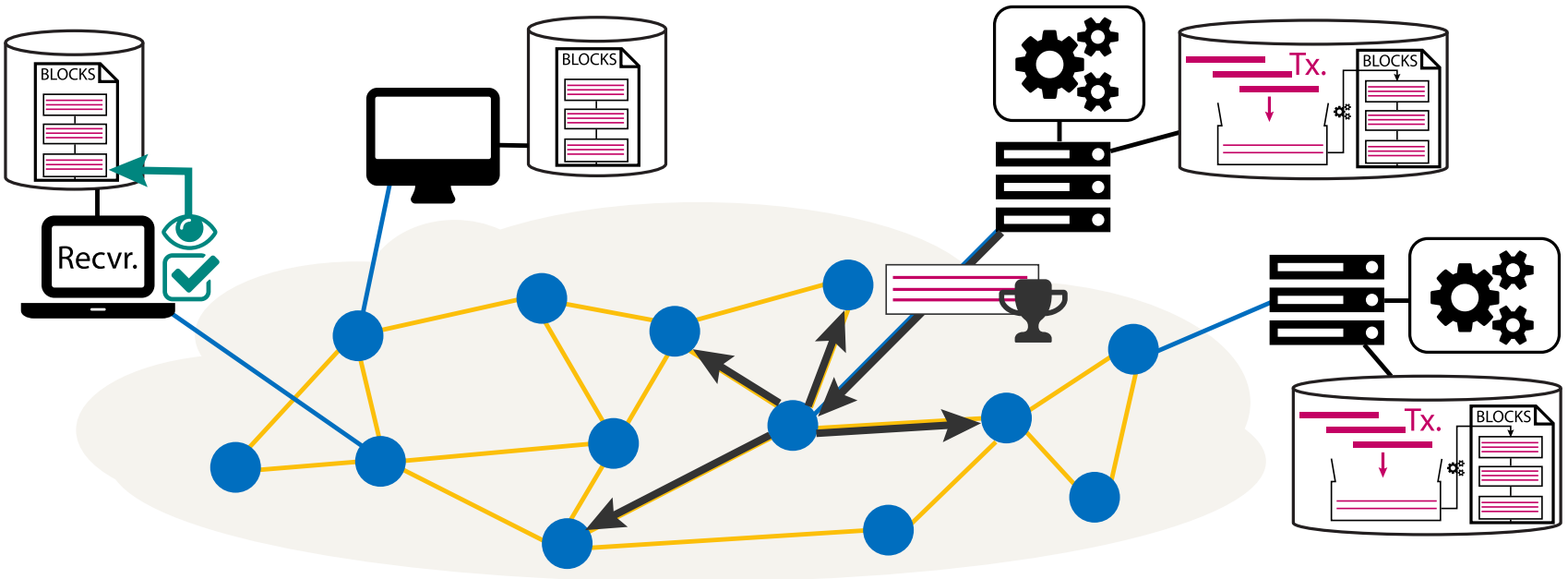
The miners include Tx. in the next block they work on.

Step 3: Includes Transaction



One miner **solves** 🏆 a block that includes Tx..
It announces the new block to the network.

Step 4: Receiver Verification



The receiver watches 👁 the blocks to see that Tx. is confirmed.
After a few blocks go by, the receiver accepts ☑ that the funds have cleared.

Transaction Confirmation

In proof-of-work, ties are resolved when one of the competing heads solves the next block.

The longest chain at any time is the most authoritative.[2]

To fork the blockchain, you need to command more than 50% of the computing power in the network.

If you're really unlucky, your transaction will end up in a dead branch, which opens you up to double spending attacks.

Transaction Confirmation
Wait a certain number of blocks before being 'sure' that your transaction will be permanent and irrefutable.
Six confirmed blocks ($\approx$ one hour) is generally considered very safe for Bitcoin.

[2] This is only true for proof-of-work.

Incentives for Mining

If you solve the proof-of-work and mine the next block, you receive two significant rewards:

The **mining reward** is essentially "free money in a block" rewarded to the successful miner.

- The only way to create new bitcoin.
- Started at 50 BTC per block, halves approximately every two years.
- Eventually will hit zero.

Transaction fees incentivise miners to include your transaction.

- A successful miner takes the transaction fees associated to each transaction in the block.
- These fees will drive mining in the long run, once the "mining reward" runs dry.

Proof-of-work Difficulty

The SHA256 algorithm outputs 256 bits which should be "approximately a random string of bits", meaning each bit has an equal probability of being a 0 or 1.

- The probability of any hash beginning with 10 zero bits is $2^{-10} \approx 0.1\,\%$.
- The probability of any hash beginning with 50 zero bits is $2^{-50} \approx 10^{-13}\,\%$.

Hash Format	XXXXXXXXXXXXX
Trivial	00XXXXXXXXXXX
Easy	0000XXXXXXXXX
Harder	000000XXXXXXX

Given the speed of hashing across the network (calculated by how fast blocks are mined), bitcoin will adjust how many zeros are required each 2016 blocks ($\approx$ two weeks), to try to make the next 2016 blocks take two weeks.

Mining hardware

Bitcoin mining (similar to password cracking) can be done *much* faster on dedicated hardware, rather than general-purpose CPUs.

Ordered by increasing specialisation, mining hardware is categorised as CPU, GPU, FPGA[3], or ASIC[4].

Type of hardware	Millions of Hashes / sec
Multi-Core CPU	35
Average GPU	200
Multi-Card GPU	2000
FPGA	400
$1,000 ASIC	10 000
$30,000 ASIC	14 000 000

An ASIC is entirely custom hardware, and is very expensive to start producing. Both ASICs and FPGAs consume far less power than CPUs or GPUs.

[3] Field Programmable Gate Array

[4] Application Specific Integrated Circuit

The Strength of Bitcoin

> "Entire classes of bugs are just missing. Bitcoin has fixed almost all flaws that aren't forced by design."
>
> — Dan Kaminksy

The bitcoin protocol is **open source**, and has been since the beginning.

Bitcoin's security resides in three places:

- The strength of public-key crypto (ECDSA) to protect accounts.
- The strength of SHA256 in the proof-of-work function.
- General secure programming.

Even if one of these were broken in the future, Bitcoin is made to be upgradeable.

Attacks on Bitcoin

Improper verification. Verification wasn't properly done on transactions before they entered the block chain and less than a week after discovery a fraudulent transaction resulted in 184 billion fake Bitcoins being created (reverted by the community).

Blockchain Forks. The blockchain temporarily forked into two independent chains due to a major software bug. The new Bitcoin client produced a transaction that wasn't accepted by the older client, splitting the blockchain and producing the first real world examples of "double spending".

Control of the Network. If an attacker had more than 50% of the computing power of the network, the attacker could perform double spending and also reject other people's transactions from receiving confirmations.

Blockchain Security

Hardware Key Solutions

Hardware is hard.

This makes it useful for key storage because breaking into hardware is difficult.
Unfortunately, developing and deploying hardware is also very difficult.

Hardware Key Solutions

Some challenges:

- Hardware still depends on software.
  - Need more sophisticated hardware to counteract
    - case study: Trezor
- Can give users a false sense of security
  - e.g. falling victim to card skimming
- Carrying hardware sucks
  - How many cards are in your wallet?
  - How many RSA tokens on your keychain?
  - **Solution:** standardisation & cooperation
    - case study: Yubikey
- Developing hardware is expensive

Hardware Key Solutions: Current State

Hardware solutions have a ways to go.

Need to consolidate solutions while allowing flexibility and upgradability.

Need to:

- Establish public and academic trust
  - Open-source is essential
  - Security through obscurity $\neq$ security
- Prevent firmware updates to prevent key theft
- Allow updates to enable future-proofing

Distributing Responsibility

Since we can't trust people to look after their own keys, we need ways to spread the risk to *multiple* people.

It's harder to hack all 5 individuals, or have 5 individuals all loose their keys, than one.

Copying the key to 5 people reduces likelihood of losing key, but increases likelihood of it being stolen or misued.

How do we distribute the responsibility of a key?

- Need to hack 2 or 3 people.
- If one person loses their key, the others can "reset" or "reissue" their key.

Distributing Responsibility: Multisigs

Multisignature is a technique to enforce complex key requirements.

In a multisig scenario, a group of n people have their own independent keys which they self-manage.

A piece of software places business-logic requirements on which keys must be present in order to perform an action.

Multisig in Bitcoin

Bitcoin has a built-in scripting language that allows software authors to determine rules.

For example:
"At least 4 of the 6 authorised keys must sign a transaction in order for the funds to be released."

Distributing Responsibility: Key Splitting

Multisigs are good when our protocol supports it, however there are many applications where multisig doesn't make sense.

Personal Crypto Fortune

"How do I give my family access to my diverse cryptocurrency fortune if I die?"

Normal circumstances:

- Only I have access.
- I have full control myself.

Extenuating circumstances:

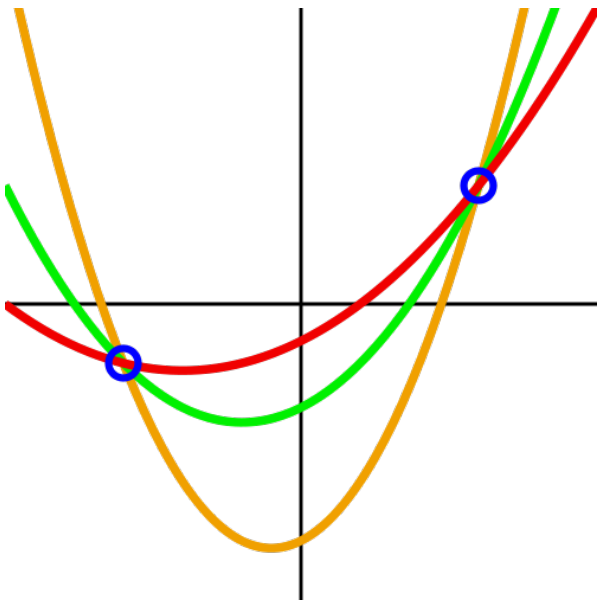
- Many trusted 3rd parties must come together
- A single rogue 3rd party cannot take control

Could be done with multisigs, but need a different implementation for each cryptocurrency.

Distributing Responsibility: Key Splitting

Shamir's Secret Sharing is an algorithm for dividing a secret into m pieces, where only n of them are required to reconstruct the original secret.

A polonomial of degree n can be uniquely defined by plotting $n + 1$ points on that polynomial.



Example

$$y = ax^2 + bx + c$$

Infinite possibilities for the coefficients with only 2 points.

3 points, and a, b, and c are uniquely determined.

Outsourcing Responsibility

So keeping crypto is hard, let someone else do it for you!

A lot of people store their cryptocurrency on exchanges.

Not actually *that* bad when you consider security vs. usability.

- Authenticated with:
  - Password
  - 2FA
  - E-mail address
- *Very* famililar
- *Very* easy to use

But people don't understand the risks.
Many of these companies have incompetent tech people, and security is *hard*.

Outsourcing Responsibility: Examples

Getting hacked is inevitable.

How you respond to a hack affects how it impacts you.

Anonymity/Pseudonymity

Blockchains are thought to be anonymous.
In general, they're not. They're **pseudonymous**.

Like all security, if your attacker is large enough, you cannot hide.

If someone can monitor the vast majority of the internet, bitcoin's anonymity is out the window.

Ethereum, however, is not.
It's P2P network protocol has built-in encryption with forward-secrecy.

Game Theory

Game Theory

Game theory is:

"the study of mathematical models of conflict and cooperation between intelligent rational decision-makers."

Many attacks and security considerations in decentralised systems are solved using game theoretic solutions.

~~Removing~~ Trust

Blockchain systems are often said to be "trustless" or that they "remove the need to trust a 3rd party".

This is straight up **false**.

Blockchains do not remove trust, they *move* trust

Instead of trusting a central entity (e.g. a bank) to perform some action with integrity, the users are trusting a whole network of computers to perform some action with integrity.

A network is a lot more complicated than a central authority, and has many points of failure.

Other Attacks

There are many other kinds of attacks on the network, such as the Denial-of-Service attacks seen on the Ethereum network.

By choosing incentive structures carefully (again, game theoretic), these can be avoided.

Ethereum EVM

The Ethereum EVM opcode: `EXTCODESIZE` costs a normal amount of 'gas', but requires significant resources to read from disks.

By spamming the network with transactions that called this 50,000 times per block, the network was brought to a halt in September 2016.

Stakeholder Influence

Since humans rely on other humans to make decisions, key personalities in the community can have a profound effect on decentralised systems.

For example, Vitalik Buterin has a *massive* influence on the price of ETH.

DAO Hack

When the DAO hack was ongoing, the price of DAO tokens crashed.

The moment Vitalik posted publically that the network would hard-fork and refund people's money, the price almost returned to normal.

Blockchain networks are about *human consensus*, not about *immutable code*.

Decentralisation

A major selling point of *public* blockchains is their decentralisation.

There are a number of different ways to measure the amount of centralisation when it comes to blockchain networks.

We will discuss the following metrics:

- Control of Mining Power
- Location of Network Nodes
- Diversity of Node Software

Decentralisation: Mining

Mining is a game of efficiency, the two extremes are:

Mining Farm

If you rent space to a hydro-electric power plant to house a warehouse full of miners, your cost of operations are low.
You can probably also afford to buy *ASICs*.

Individuals

If you live in an apartment in Sydney, power is very expensive and the economies of scale are not competitive.
There is less incentive to mine.

Miners control the network, and the economics encourage them to consolidate and *centralise*.

Decentralisation: Network

The nodes which form the underlying peer-to-peer network are often hosted in similar scenarios, such as:

- Hosted on Amazon's AWS
- Located in a particular country
  (e.g. Lots of mining in China)

Large-scale outages can have severe impacts on network availability as a whole.

The Great Firewall

Imagine that China blocks Bitcoin using the Great-Firewall...

All the mining power in China suddenly drops off the network.

The result:

- A poorly connected P2P network
- Very long block times
  Losing hash power = slower blocks...

...until the network adjusts.

Decentralisation: Software

A software monoculture can be a dangerous thing.

Blockchain networks operate according to well defined protocols, but software implementations differ.
Etherum has clients written in:

C++: `Aleth (cpp-ethereum)`

Go: `go-ethereum (geth)`

Rust: `parity`

Python: `pyethapp`

Java: `ethereumj`

Beyond Bitcoin

Why Blockchains?

Do you *really* need a blockchain?

Blockchain = Timestamp Server

At it's heart, a blockchain is just a **decentralised, append-only, timestamp server**.

It does **not**:

- *Remove* the need for trust
- Process transactions at scale (yet[5])
- Provide efficient computation or information transfer

It **can**:

- *Move* trust
- Improve data integrity
- Strengthen non-repudiation
- Provide a verifiable order of events

[5] See Ethereum's planned sharding technique

Ask: Why Blockchain?

When someone says they want to use blockchains, a good way to differentiate those on the *hype train* and those with a serious use case is to ask a simple question:

Why blockchains?

If you don't get a concise and sensible answer, they're probably on the hype train.

Public vs. Private Blockchains

Public: *Moves* trust from a centralised organisation to a decentralised group of (humans running) computers.
Publically accessible
Not necissarily publically readable

- Bitcoin
- Ethereum
- Monero

Private: Improves efficiency and data integrity across large organisations or groups of organisations.
Only **verified parties** *can participate in the network.*

- Eris Industries
- Rubix

Public Blockchains

Properties:

- Anyone can participate.
- *Sometimes*, anyone can read.
- Secured by many participants, the more the better.

Use Cases:

- Censorship avoidance
- Prevent corruption
- Move trust away from organisations

Developing Economies

In 1st world, organisations work, corruption not a concern.

In 3rd world, corruption is everywhere.

Biggest market = developing nations.

Private Blockchains

Properties:

- Only authorised parties can participate or read.
- a.k.a. “permissioned” blockchain.
- Secured by authorisation lists.
- Not very decentralised

Use Cases:

- More rigerous audit trails
- Prevent corruption/improve integrity
- Improve cooperation & transparency
- More efficient, append-only database

Namecoin: More complex

Namecoin was the first "altcoin" and extends Bitcoin functionality by acting as a distributed DNS system.

A transaction in Namecoin extends Bitcoin's transactions:

- Name Update

Users can register ".bit" domains (e.g. usyd.bit) by submitting a transaction with the "Name Update" command.

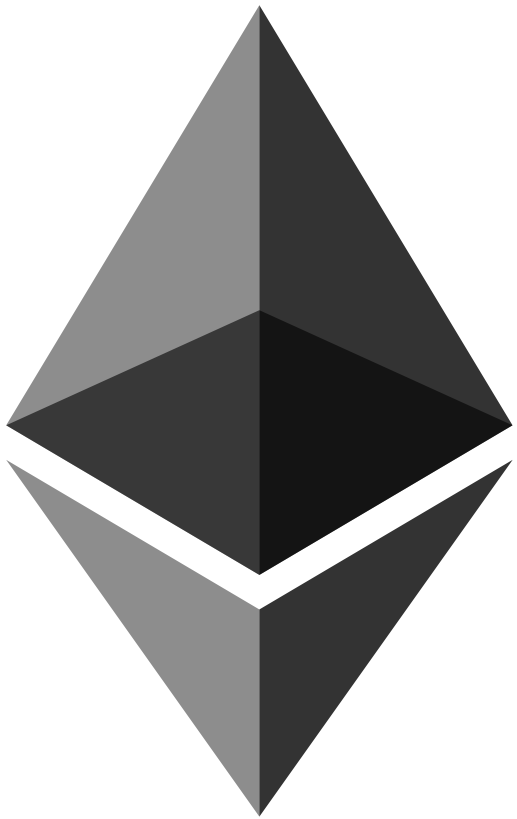
Ethereum: Arbitrarily Complex

Ethereum fully extends the concept of transactions, by creating “smart contracts”.

A contract is like an independent entity that is governed by it’s (full-turing) code. It has data storage, it’s own currency balance, and can be interacted with via transactions.

Now we can easily run arbitrary software on a blockchain without creating a whole new chain.

Ethereum has an internal currency, “ether”, which is burned as “gas” when executing contracts.



Smart Contracts

A **smart contract** is a program on the blockchain.

Like a regular program, a smart contract has:

- Code
- Inputs
- Outputs
- Data storage

Unlike a regular program, a smart contract is executed by *all* nodes in the network.

Anyone can send a message (input) to a smart contract, causing all nodes in the network to agree on the state change caused as a result of the message and code.

Smart Contracts

A smart contract:

- is fully defined by it's initial code.
  (code is public)
- cannot be modified.
  (unless it's code explicitly permits)
- can receive input from anyone on the network.

Since a smart contract could fully encapsulate business logic, they can be used to create:

decentralized autonomous organisations (DAOs).

Dapp: Lotto Pollo

The premise[6] is simple:

- Initial balance: 100 ether.
- Send at least 1 ether to participate
- If 24 hours elapse with no deposit:
  The last participant receives all Ether sent by all participants.

Before Ethereum, this would be a scam.

The entity running this could easily run away with all the money.

But now, our entity is a smart contract. It is governed by it's code, and controlled by nobody. It's completely fair and trustworthy.

http://lottopollo.com/

[6] https://www.reddit.com/r/ethereum/comments/3ivfyn/lotto_pollo_lottery_chicken/

Dapp: Slock.it

Slock.it allows you to rent or share anything - without middlemen.

1. You announce that you have an asset to rent, on the blockchain.
   e.g. a bicycle
2. Users pay a deposit to rent it
3. A custom electronic IoT-style[a] lock on the asset:
   3.1 Watches the blockchain.
   3.2 Realises that a user has hired it.
   3.3 Unlocks.
4. User returns the item, ends the rent and the lock re-engages.

[a] Internet of Things

Ethereum: Augur

Augur is a decentralised prediction market.

Users can bet on the outcome of real-world events, such as the outcome of an election.

But how to confirm the winner or loser?

We need a way of introducing real-world data into the blockchain...

Reputation Token

Holders of REP tokens *stake* their reputation (i.e. money), claiming that they will report the correct outcome at the end of the bet. If they act maliciously, other moderators can stake against them and take their money.

In this way, we can reliably introduce real-world data to the blockchain.

Smart Contract: Tokens

Imagine a simple smart contract that creates "tokens" and simply tracks the balance of each user.

```
1  contract FixedSupplyToken {
2      uint _totalSupply;
3      mapping(address => uint) balances;
4
5      function balanceOf(address tokenOwner) public view returns
           (uint balance) {
6          return balances[tokenOwner];
7      }
8
9      function transfer(address to, uint tokens) public returns
           (bool success) {
10         balances[msg.sender] = balances[msg.sender].sub(tokens
               );
11         balances[to] = balances[to].add(tokens);
12         return true;
13     }
14 }
```

Smart Contract: ERC20 Tokens

Using this smart contract, we can launch tokens **on top** of an underling blockchain, such as Ethereum.

When these tokens are implemented in a standard way, they can be transferred and traded, much like Ether or Bitcoin, however transactions are submitted to the Ethereum blockchain.

The standard implementation of these tokens is the ERC-20 standard.

Full ERC20 Contract

The smart contract shown on the previous slide is an incomplete implementation of the ERC-20 standard, adapted from theethereum.wiki.

ERC20

ERC20 is a software interface

Contains six functions:

- `totalSupply() constant returns (uint256 totalSupply)`
- `balanceOf(address _owner) constant returns (uint256 balance)`
- `transfer(address _to, uint256 _value) returns (bool success)`
- `transferFrom(address _from, address _to, uint256 _value) returns (bool success)`
- `approve(address _spender, uint256 _value) returns (bool success)`
- `allowance(address _owner, address _spender) constant returns (uint256 remaining)`

ERC20

And two events:

- `Transfer(address indexed _from, address indexed _to, uint256 _value)`
- `Approval(address indexed _owner, address indexed _spender, uint256 _value)`

ERC20 Examples

The following list are some example ERC20 tokens that are live on the public Ethereum blockchain:

OMG: OmiseGo
Financial technology for digital wallets
0xd26114cd6EE289AccF82350c8d8487fedB8A0C07

BAT: Basic Attention Token
A new system of digital advertising
0x0d8775f648430679a709e98d2b0cb6250d2887ef

GNT: Golem
Decentralised market for computing power
0xa74476443119A942dE498590Fe1f2454d7D4aC0d

REP: Reputation (Augur)
Decentralised prediction markets
0x1985365e9f78359a9B6AD760e32412f4a445E862

SNT: StatusNetwork
Decentralised messaging platform
0x744d70fdbe2ba4cf95131626614a1763df805b9e

The DAO – Overview

The DAO is a smart contract.

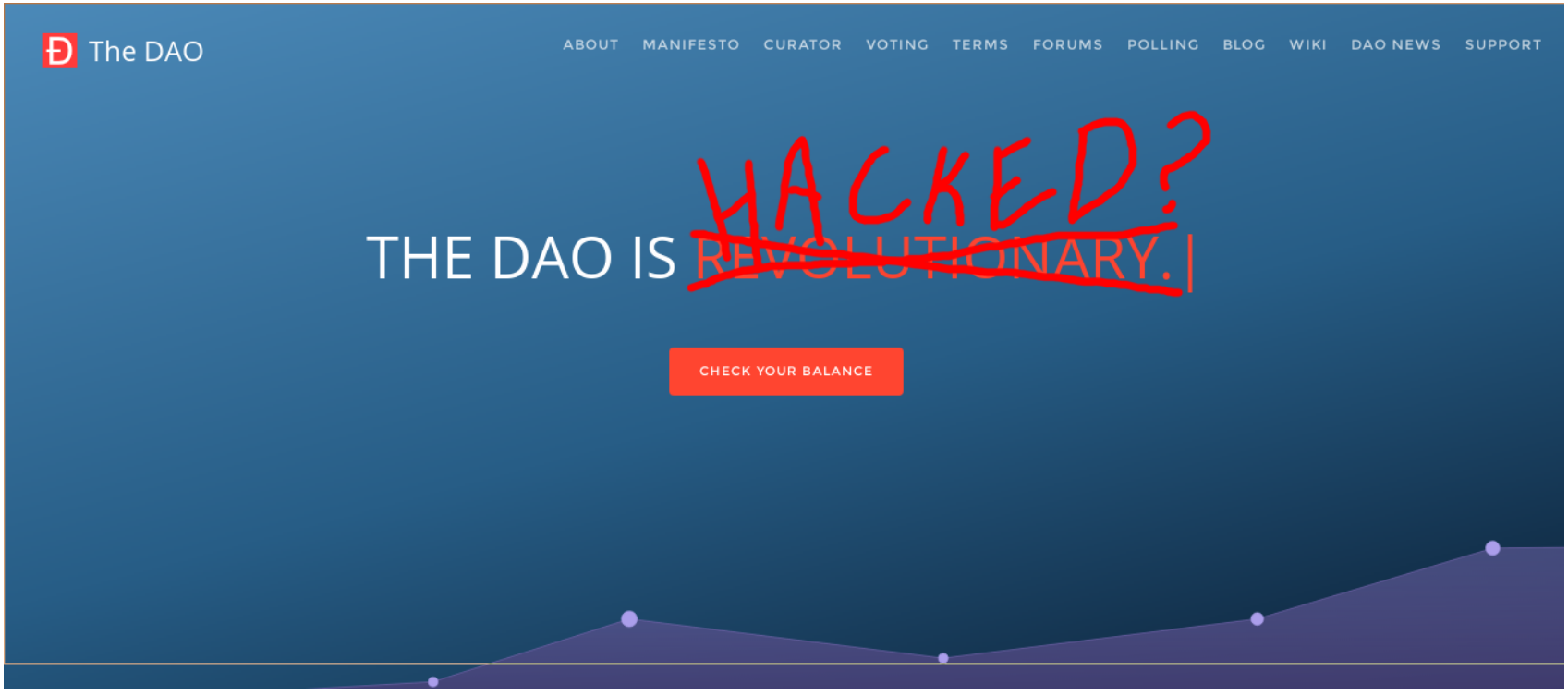
It was supposed to be a kind of crowd-funded decentralised venture capital firm.

"Largest" crowd funding campaign in history – about USD $150M equiv. Ether

More details: https://github.com/slockit/DAO

1. Early investors send ETH to contract during "creation phase" – like making shares in an IPO.
2. Proposals can choose to invest ETH in a project
3. Investors can withdraw/refund their own ETH by calling "Split DAO" function
4. Withdrawing investor goes to their *own* DAO, where they can refund themselves.

The DAO – Hacked?



A bug in the code allowed an attacker to recursively withdraw more funds to their own Split DAO, so someone did just that.

They stole about 3.5M ETH out of about 12.5M ETH that The DAO was holding.

The DAO – Hacked?

But is it a hack?

The DAO Terms

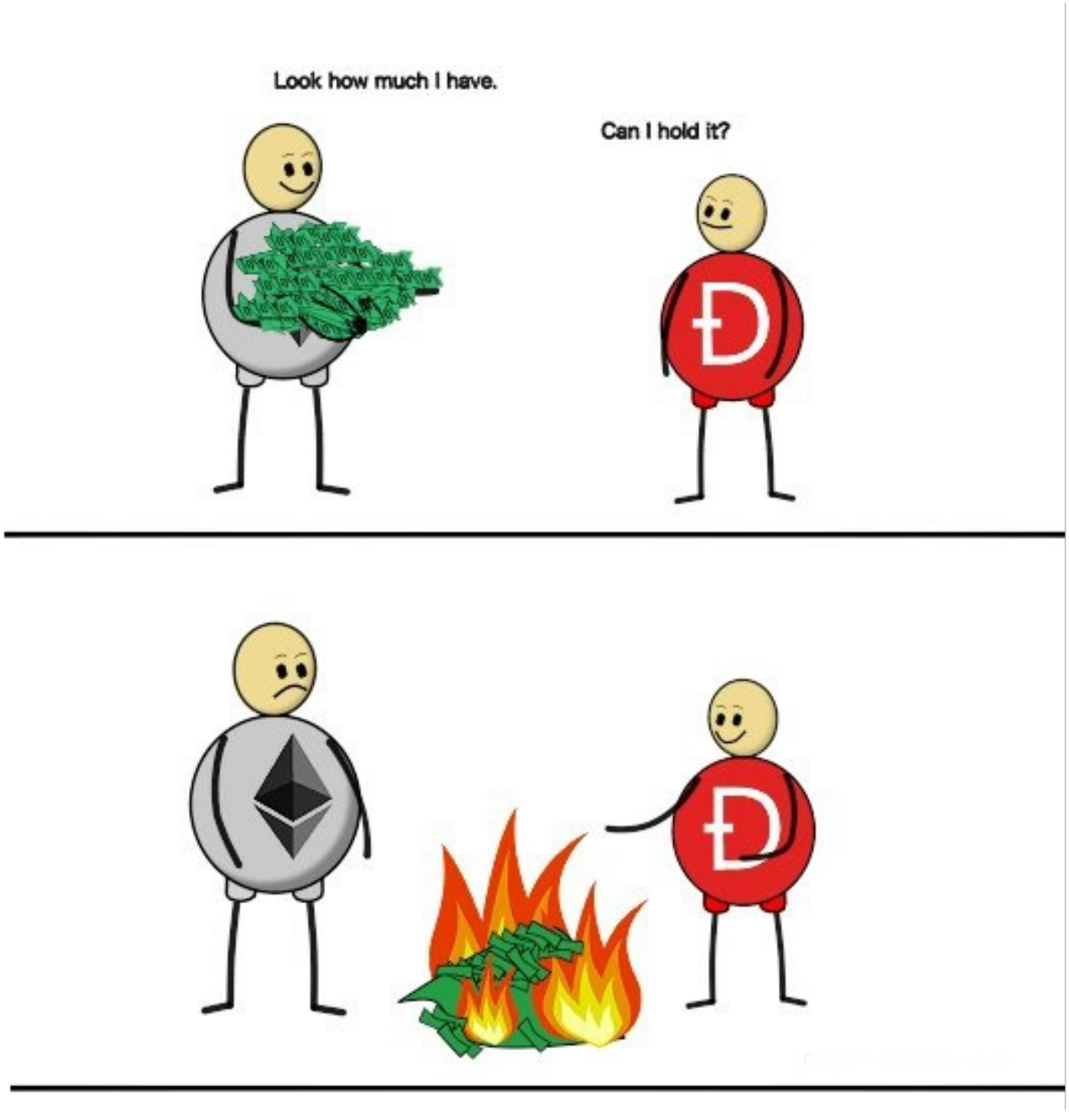
"The terms of The DAO Creation are set forth in the smart contract code existing on the Ethereum blockchain at: `0xbb9bc244d798123fde783fcc1c72d3bb8c189413`.

Nothing in this explanation of terms or in any other document or communication may modify or add any additional obligations or guarantees beyond those set forth in The DAO's code."

Apparently the hacker claims the exploit is a "feature" of the code, and who can argue?

Open letter "from the attacker" (not verified):
http://pastebin.com/CcGUBgDG

The DAO – Hacked?



Initial Coin Offerings

Where do "tokens" come from?

When a new blockchain system is developed, it usually starts with an initial allocation of token supply.

People purchase pre-allocated tokens before the blockchain is launched.

This process is called an initial coin offering (ICO), and is similar in some ways to an IPO.

"The Killer Dapp"

ICOs or "Crowdsales" are very popular use case of Ethereum.
To understand more about crowdsales take a look at:

`https://www.ethereum.org/crowdsale`

Large ICOs

There have been some very large ICOs, some notable ones are listed below:

DAO: The DAO
USD $152 Million

BNT: Bancor
USD $153 Million

XTZ: Tezos
USD $232 Million

TON: Telegram
USD $1.7 Billion

EOS: EOS
USD $4.1 Billion