



# How to integrate blockchain technology to businesses



Professor Fatih Ozaydin  
[fatih@tiu.ac.jp](mailto:fatih@tiu.ac.jp)

# Outline

---

- ▶ Brief introduction to Blockchain
  - ▶ Background
  - ▶ Essentials
- ▶ Blockchain in Business
- ▶ Cons, Threats & Solutions

# Background

## ▶ Assets

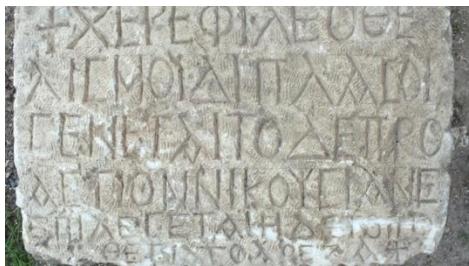
- ▶ Tangible: Has physical form
  - ▶ e.g. stones, land, buildings
- ▶ Intangible: No physical (solid) form
  - ▶ e.g. digital, intellectual



Yap Stone. img src:

## ▶ Ledger

- ▶ To record the transfer of assets

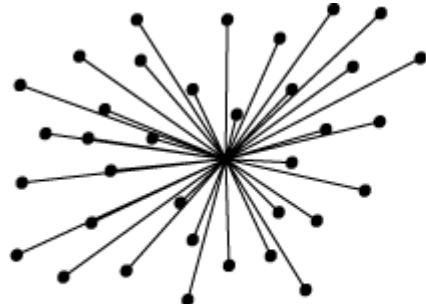


General Ledger List - Current						
View Using:						
Set GL Date Range:						
<input checked="" type="checkbox"/> Set GL Date Range	<input type="checkbox"/> Transaction Id	<input type="checkbox"/> Specific Transaction Id				
To: 07/02/2010	<input type="checkbox"/> Transaction Id Range					
	<input type="checkbox"/> Reference Number					
	<input type="checkbox"/> DR amount Between					
	<input type="checkbox"/> CR amount Between					
	<input type="checkbox"/> Specific Jnl Type					
	<input type="checkbox"/> Specific Acct#					
Transaction Id	Jnl	Date	Reference	Dept	Acct	DR
						CR
2010060001	AP	06/09/10 222	000 01010		673.00	673.00 1099 Te
2010060002	AP	06/09/10 222	000 01010		673.00	0.00 Void for
2010060003	AP	06/09/10 222	000 02620		673.00	673.00 Void for
2010060004	FB	06/16/10 012246582	000 04010		3400.00	3400.00 FAMREC
2010060005	FB	06/16/10 012246582	001 04020		0.00	668.10 FAMREC
2010060006	AR	06/16/10 012246582	001 04010		0.00	668.10 FAMREC
2010060007	AR	06/16/10 012246582	001 04010		0.00	668.10 FAMREC
2010060008	AR	06/16/10 All Account	000 01210		6500.00	6500.00 Cash rec
2010060009	AR	06/16/10 Adjustment	000 01210		0.00	0.00 Cash rec
2010060010	PRI	06/23/10 Batch 228	000 05630		450.00	0.00 ABC Es
2010060011	PR	06/23/10 Batch 228	000 01010		0.00	376.95 ABC Ne

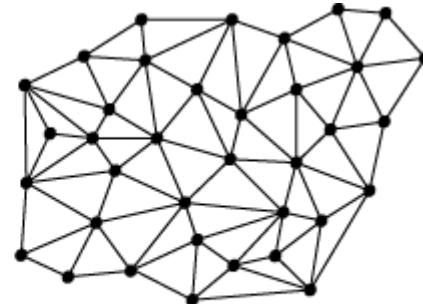
# Background

---

- ▶ **Ledger:**
  - ▶ Centralized
  - ▶ Distributed (shared)
    - ▶ Each node keeps a copy
    - ▶ Requires technology, e.g. Cryptography and effort



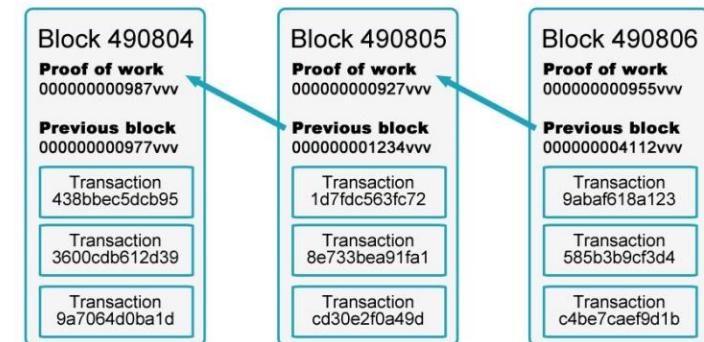
centralised



distributed

# Background

- ▶ An append-only distributed ledger
  - ▶ Transaction Records are stored in blocks
  - ▶ Blocks form a chain: The Ledger
  - ▶ Each node has a copy of the ledger:
  - ▶ Consensus required (longest chain)
    - ▶ Permissionless
    - ▶ Permissioned
  - ▶ With mining: Proof of Work
  - ▶ Without Mining: Proof of Stake, Practical Byzantine Fault Tolerance



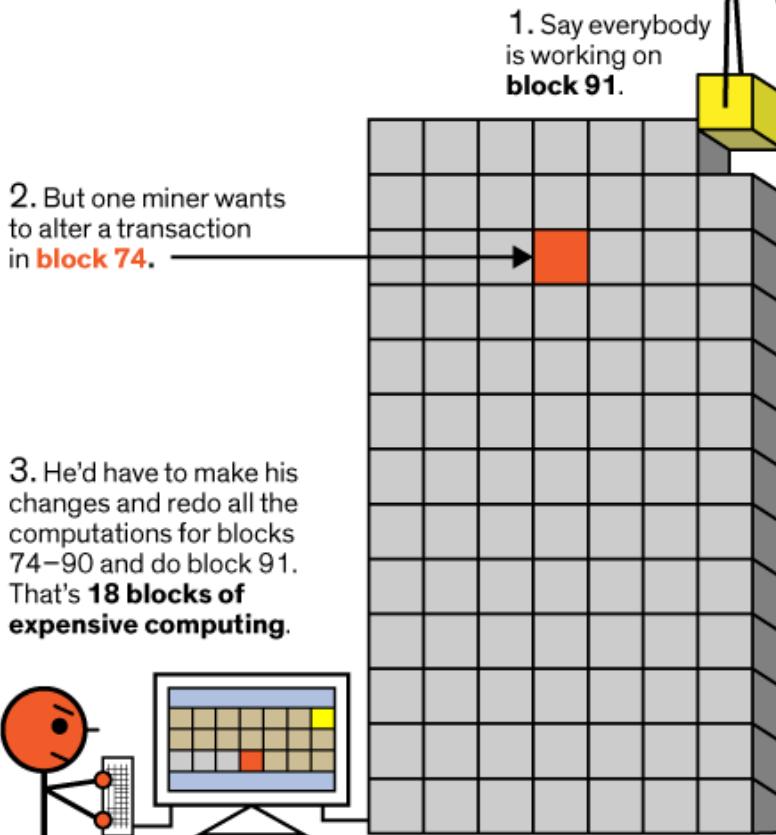
# With mining: Proof of Work

---

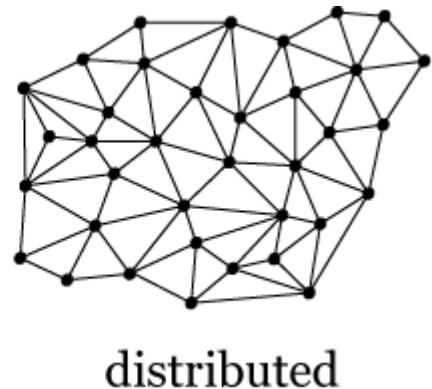
- ▶ Miners: Peers that maintain and update the Blockchain
  - ▶ Solve a complex mathematical problem (cryptography)
  - ▶ Send transactions to other nodes for verification
  - ▶ Paid a reward (such as, some Bitcoin)
- ▶ If all miners agree on solution → Block is added to Chain

# Why You Can't Cheat at Bitcoin

Secu



4. What's worse, he'd have to do it all **before** everybody else in the Bitcoin network finished **just the one block (number 91)** that they're working on.



Immutable:  
Write & Read-Only

Illustration: Mark Montgomery, IEEE Spectrum

# Blockchain

---

- ▶ A system consisting of
  - ▶ Transactions
  - ▶ Immutable Ledgers
  - ▶ Distributed (decentralized) Peers
  - ▶ Encryption Processes
  - ▶ Consensus Mechanisms
- ▶ (Optional) Smart Contracts ?

# Smart Contracts

---

- ▶ Aims to minimize the requirement of “trust”
- ▶ Actually, neither ‘smart’, nor ‘contract’
- ▶ Software, deployed onto a Blockchain
  - ▶ Transaction instructions triggered by events  
**If X, then Y**
  - ▶ Example
    - If** the tuition fee is paid, **then** approve the registration
  - ▶ Agreements follow **If-then** logic,
    - ▶ Can be replaced with Smart Contracts!

# Blockchain Applications

---

- ▶ **Cryptocurrencies**
  - ▶ Bitcoin,
  - ▶ Altcoins: Ripple, Ethereum, Monero, Zcash, etc.
- ▶ **Managing Transactions in Business; can improve**
  - ▶ procure-to-pay,
  - ▶ accounts receivable
  - ▶ accounts payable,
  - ▶ general ledger,
  - ▶ reconciliation,
  - ▶ payroll

# Business Blockchains

---

- ▶ Set up by
  - ▶ A single company or
  - ▶ A group of specified & known companies
  
- ▶ Does it make sense for our business?
  - ▶ Number of participants
  - ▶ Complexity of business
  - ▶ Long-term record keeping & regulatory compliance
  - ▶ Real-time transfer of assets or payments

# Real Life Business Blockchain Examples

---

- ▶ **Retail**
  - ▶ Warranteer, Blockpoint, Loyyal
- ▶ **Supply Chains and Logistics**
  - ▶ IBM Blockchain, Provenance, Blockverify, OriginTrail
- ▶ **Insurance**
  - ▶ Accenture
- ▶ **Healthcare**
  - ▶ MedicalChain, MedRec, Gem
- ▶ **Real Estate**
  - ▶ BitProperty, Deedcoin (1% Comission)

# How to integrate Blockchain to our Business

---

- ▶ **9-Component Framework by HashEx**
  - ▶ 1. Business Objective
  - ▶ 2. Benefits
  - ▶ 3. Concept
  - ▶ 4. Architecture
    - ▶ Consensus Rules,
    - ▶ Data Privacy, outsider?
    - ▶ Complexity of executable algorithms

- 
- ▶ 5. Roadmap
    - ▶ Description of Minimum Viable Product (MVP)
    - ▶ MVP vs fully functional product
    - ▶ Platform selection for MVP implementation
    - ▶ Prototyping
    - ▶ Prototype viability check
    - ▶ Describing the fully functional product
    - ▶ Selecting the platform
    - ▶ Implementation

- 
- ▶ 6. Minimum Viable Product (MVP)
  - ▶ 7. MVP vs Fully Functional Product
  - ▶ 8. Prototype
  - ▶ 9. Cases

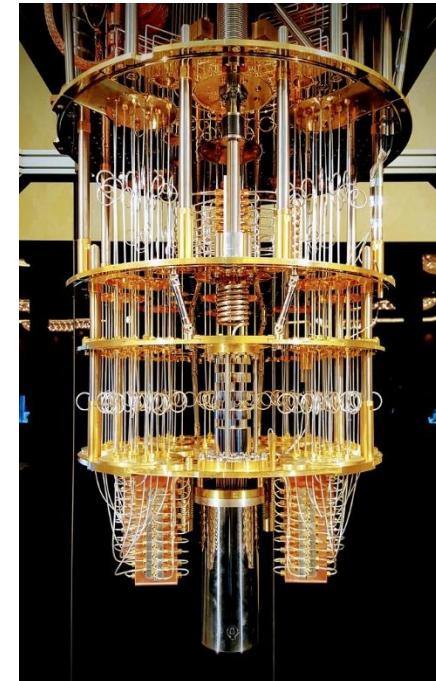
# Applications

---

- ▶ Smart Contracts (Ethereum)
- ▶ Paying Employees (BitWage)
- ▶ Electronic Voting (BitShares)
- ▶ IBM Blockchain
- ▶ Creating own cryptocurrency
  - ▶ Possibly using CryptoLife, WalletBuilders, etc.

# Cons, Threats, Solutions & Opportunities

- ▶ Security depends on complex mathematical functions
  - ▶ Difficult to solve (for classical computers)  
Mining requires computational power (electricity)  
BitCoin Currently consuming ~3 GWatts/year (~ Ireland)
  - ▶ Motivation to hack others' computers & servers
  - ▶ Easy to solve for Quantum Computers
    - ▶ A serious threat in ~10 years time
- ▶ Solution to both security & environmental issues:
  - ▶ Quantum Blockchain
- ▶ Opportunity: New Science Minister, Takuya Hirai



# Sources & Read More

---

- ▶ <https://mastanbtc.github.io/blockchainnotes/>
- ▶ <https://en.bitcoin.it/wiki/Contract>
- ▶ <https://www.forbes.com/sites/bernardmarr/2018/05/14/30-real-examples-of-blockchain-technology-in-practice/>
- ▶ <https://tradeix.com/essential-blockchain-technology-concepts/>
- ▶ <https://medium.com/hashex-blog/how-to-integrate-blockchain-464d241a6681>
- ▶ <http://fortune.com/2017/08/22/walmart-blockchain-ibm-food-nestle-unilever-tyson-dole/>
- ▶ <https://bitshares.org/technology/delegated-proof-of-stake-consensus/>
- ▶ <https://execed.economist.com/blog/industry-trends/5-applications-blockchain-your-business>
- ▶ <https://www.walletbuilders.com/>
- ▶ <https://dev.cryptolife.net/>
- ▶ <https://hashex.org/>
  
- ▶ Blockchain Demo: <https://anders.com/blockchain/blockchain.html>

# **Blockchain & Business Application**

## **Lecture 2: Introduction to Cryptography**

**Most of these slides are from:**

**Harvard Professor Boaz Barak's  
«COS 43 Cryptography» Lecture Notes at Princeton**

**This part covers only «Classical Cryptography»**

**But we will later discuss the threat by quantum computers,  
and the idea of «Quantum Cryptography»  
as well as «Quantum Blockchain»**

# Cryptography

History of 2500- 4000 years.

Throughout most of this history:

cryptography = “secret writing”:

“Scramble” (**encrypt**) text such that it is hopefully unreadable by anyone except the intended receiver that can **decrypt** it.

Recurring theme: (until 1970's)

- n Secret code invented
- n Typically claimed “unbreakable” by inventor
- n Used by spies, ambassadors, kings, generals for crucial tasks.
- n Broken by enemy using **cryptanalysis**.

*“Human ingenuity cannot concoct a cipher which human ingenuity cannot resolve.”*

Edgar Alan Poe, 1841

# Crypto History

**1587:** Ciphers from Mary of Scots plotting assassination of queen Elizabeth broken; used as evidence to convict her of treason.

**1860's (civil war):** Confederacy used good cipher (Vigenere) in a bad way. Messages routinely broken by team of young union cryptanalysts; in particular leading to a Manhattan manufacturer of plates for printing rebel currency.

**1878:** New York Tribune decodes telegram proving Democrats' attempt to buy an electoral vote in presidential election for \$10K.

**1914:** With aid of partial info from sunken German ships, British intelligence broke all German codes.  
Cracked telegram of German plan to form alliance with Mexico and conquer back territory from U.S. As a result, U.S. joined WWI.

**WWII:** Cryptanalysis used by both sides. Polish & British cryptanalysts break supposedly unbreakable **Enigma** cipher using mix of ingenuity, German negligence, and mechanical computation. Churchill credits cryptanalysts with winning the war.

# Crypto History

**1976:** Diffie and Hellman propose new, more ambitious, notion of “public key cryptography” based on simple to state, hard to solve, computational problem.

*“We stand today on the brink of a revolution in cryptography”*

**1977:** Rivest, Shamir and Adleman (RSA) propose another public key crypto candidate.

**1977-:** Schemes stay unbroken despite attacks with unprecedented manpower and computer cycles.

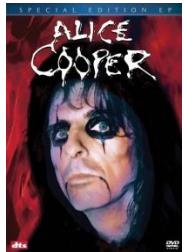
**1980's-:** Web of reductions – even more ambitious notions: CCA secure encryption, CMA secure signatures, zero knowledge, electronic cash, electronic elections and auctions, privacy preserving data mining, .... , fully homomorphic encryption (**2009**).

**Today:** Breaking cryptography not considered top cyber security threat.

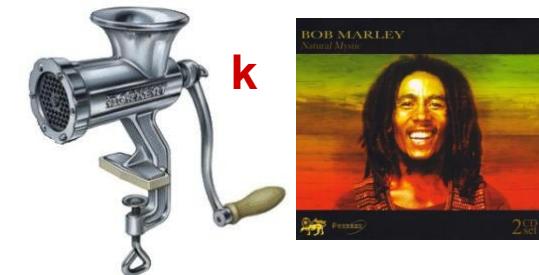
# Encryption Schemes

Alice wants to send Bob a secret message.

$m:$  AMEX 1234567890



$m' = D(c)$  AMEX 1234567890



They agree in advance on 3 components:

- n Encryption algorithm: E
- n Decryption algorithm: D
- n Secret key: k

To encrypt plaintext  $m$ , Alice sends  $c = E(m,k)$  to Bob.

To decrypt a ciphertext  $c$ , Bob computes  $m' = D(c,k)$ .

- n A scheme is valid if  $m' = m$
- n Intuitively, a scheme is secure if eavesdropper can not learn  $m$  from  $c$ .

# Example 1: Caesar's Cipher

**Key:**  $k = \text{no. between 0 and 25}$ .

**Encryption:** encode the  $i^{\text{th}}$  letter as the  $(i+k)^{\text{th}}$  letter.  
(working mod 26:  $z+1=a$ )

**Decryption:** decode the  $j^{\text{th}}$  letter to the  $(j-k)^{\text{th}}$  letter.

**Plain-text:** S E N D      R E I N F O R C E M E N T

**Key:** 2

**Cipher-text:** U G P F      T F K P H Q T E G O G P V

**Problem:** only 26 possibilities for key – can be broken in short time.

**Kerchoff's Principle (1883):** System should be secure even if algorithms are known, as long as key is secret.

In other words: “**security through obscurity**” does not work.

# Example 2: Substitution Cipher

**Key: k = table mapping each letter to another letter**

A	B	C		Z
U	R	B		E

**Encryption and decryption: letter by letter according to table.**

**# of possible keys:  $26!$  ( $= 403,291,461,126,605,635,584,000,000$ )**

**However – substitution cipher is still insecure!**

**Key observation:** can recover plaintext using statistics on letter frequencies.

Here Up On Le Grand Arose With A Grave And Stately Air And Brought  
L I V I T C S W P I Y V E W H E V S R I Q M X L E Y V E O I E W H R X E X I P F E M V E W H K V S T Y L X  
Me The Beetle From A Glass Case In Which It Was Enclosed It Was A Be  
Z I X L I K I I X P I J V S Z E Y P E R R G E R I M W Q L M G L M X Q E R I W G P S R I H M X Q E R E K I

I – most common letter

I=e    L=h    X=t

LI – most common pair

V=r    E=a    Y=g

XLI – most common triple

# Example 3- Vigenere (Belaso, 1553)

“Multi-Caesar Cipher” – A **stateful** cipher

**Key:**  $k = (k_1, k_2, \dots, k_m)$  list of  $m$  numbers between 0 and 25

**Encryption:**  $i^{\text{th}}$  letter encoded as Caesar w/ key  $\rightarrow k_{(n \bmod m)}$

**Decryption:** In the  $n^{\text{th}}$  letter may be decoded as Caesar w/ key  $= k_2$  :  $i \rightarrow i + k_2 \pmod{26}$

**Important Property:** Can no longer break using letter frequencies alone.

$m^{\text{th}}$  letter encoded as Caesar w/ key  $= k_m$  :  $i \rightarrow i + k_m \pmod{26}$

‘e’ will be mapped to ‘e’+ $k_1$ , ‘e’+ $k_2$ , ..., ‘e’+ $k_m$  according to location.

$m+1^{\text{th}}$  letter encoded as Caesar w/ key  $= k_1$  :  $i \rightarrow i + k_1 \pmod{26}$

Considered “unbreakable” for 300 years (broken by Babbage, Kasiski 1850’s)

# Example 3- Vigenere (Belaso, 1553)

“Multi-Caesar Cipher” – A **stateful** cipher

**Key:**  $k = (k_1, k_2, \dots, k_m)$  list of  $m$  numbers between 0 and 25

**Encryption:**  $n^{\text{th}}$  letter encoded w/ key= $k_{(n \bmod m)}$  :  $i \rightarrow i + k_{(n \bmod m)} \pmod{26}$

**Decryption:** In the natural way

**Breaking Vigenere:**

LIVITCSWPIYVEWHEVSRIQMXLEYVEOI E W H R X E X I P F E M V E W H K V

**Step 1:** Guess the length of the key  $m$

**Step 2:** Group together positions  $\{1, m+1, 2m+1, 3m+1, \dots\}$

$\{2, m+2, 2m+2, 3m+2, \dots\}$

...

$\{m-1, 2m+m-1, 3m+m-1, \dots\}$

# Example 3- Vigenere (Belaso, 1553)

“Multi-Caesar Cipher” – A **stateful** cipher

**Key:**  $k = (k_1, k_2, \dots, k_m)$  list of  $m$  numbers between 0 and 25

**Encryption:**  $n^{\text{th}}$  letter encoded w/ key= $k_{(n \bmod m)}$  :  $i \rightarrow i + k_{(n \bmod m)} \pmod{26}$

**Decryption:** In the natural way

**Breaking Vigenere:**

LIVITC  
SWPIYV  
EWHEVS  
RIQMXL  
EYVEOI  
EWHRXE  
XIPFEM  
VEWHKV

**Step 1: Guess the length of the key  $m$**

**Step 2: Group together positions  $1, m+1, 2m+1, 3m+1, \dots$**

$\{2, m+2, 2m+2, 3m+2, \dots\}$

...

$\{m-1, 2m+m-1, 3m+m-1, \dots\}$

**Step 3: Frequency-analyze each group independently.**

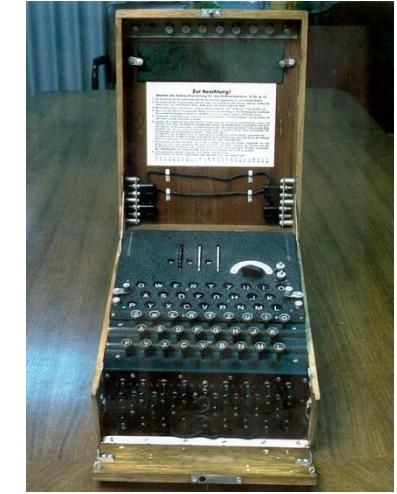
# Example 4 - The Enigma

A mechanical stateful cipher.

Used by Germany in WWII for top-secret communication.

**Roughly:** composition of 3-5 substitution ciphers implemented by wiring.

Wiring on rotors moving in different schedules, making cipher **stateful**



**Key:**

- 1) Wiring of machine (changed infrequently)
- 2) Daily key from code books
- 3) New operator-chosen key for each message

Tools used by Poles & British to break Enigma:

- 1) Mathematical analysis combined w/ mechanical computers
- 2) Captured machines and code-books
- 3) German operators negligence
- 4) Known plaintext attacks (greetings, weather reports)
- 5) Chosen plaintext attacks

# Post 1970's Crypto

Two major developments:

## 1) Provably secure cryptography

Encryptions w/ mathematical proof that are **unbreakable\***

\* Currently use conjectures/axioms,  
however defeated all cryptanalysis effort so far.

## 2) Cryptography beyond “secret writing”

Public-key encryptions

Digital signatures

Zero-knowledge proofs

Anonymous electronic elections

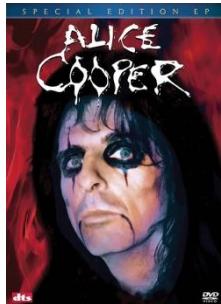
Privacy-preserving data mining

e-cash

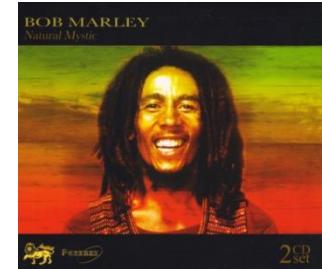
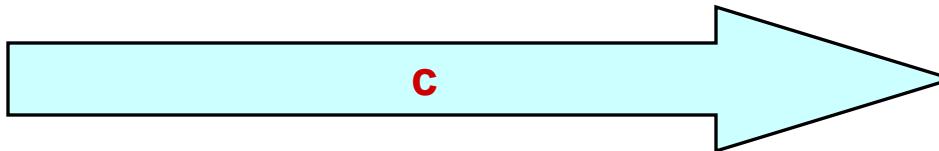
...

# Review of Encryption Schemes

Alice wants to send Bob a secret message.



$$c = E(m, k)$$



$$m' = D(c, k)$$

- n Encryption algorithm:  $E$
- n Decryption algorithm:  $D$
- n Secret key:  $k$

To encrypt  $m$ , Alice sends  $c = E(m, k)$  to Bob.  
To decrypt  $c$ , Bob computes  $m' = D(c, k)$ .

**Q:** Can Bob send Alice the secret key over the net?

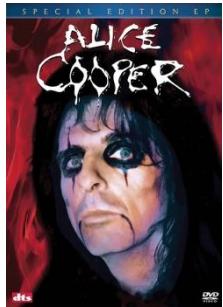
**A:** Of course not!! Eve could decrypt  $c$ !

**Q:** What if Bob could send Alice a “crippled key”

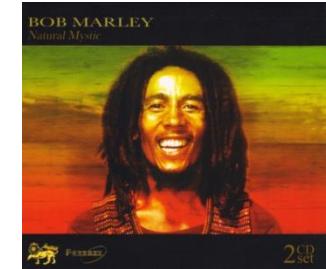
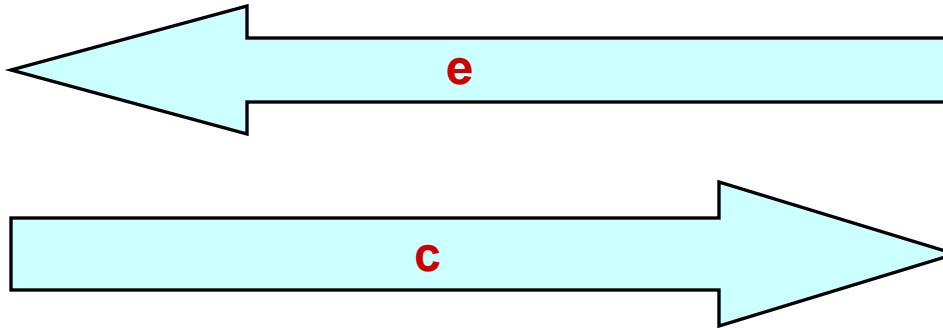
useful only for encryption but no help for decryption

# Public Key Cryptography [DH76,RSA77]

Alice wants to send Bob a secret message.



$$c = E(m, e)$$



$$\text{choose } d, e$$
$$m' = D(c, d)$$

- n Encryption algorithm:  $E$
- n Decryption algorithm:  $D$
- n Key: Bob chooses two keys:
  - Secret key  $d$  for decrypting messages.
  - Public key  $e$  for encrypting messages.

To encrypt  $m$ , Alice sends  $c = E(m, e)$

Even if Eve knows the key  $e$ !

Should be safe to send  $e$  "in the clear"

n A scheme is valid if  $m' = m$

n Intuitively, a scheme is secure if eavesdropper can not learn  $m$  from  $c$ .

# Other Crypto Wonders

**Digital Signatures.** Electronically sign documents in unforgeable way.

**Zero-knowledge proofs.** Alice proves to Bob that she earns <\$50K without Bob learning her income.

**Privacy-preserving data mining.** Bob holds DB. Alice gets answer to one query, without Bob knowing what she asked.

**Playing poker over the net.** Alice, Bob, Carol and David can play poker over the net without trusting each other or any central server.

**Distributed systems.** Distribute sensitive data to 7 servers s.t. as long as <3 are broken, no harm to security occurs.

**Electronic auctions.** Can run auctions s.t. *no one* (even not seller) learns anything other than winning party and bid.

**Fully homomorphic encryption.** Encrypt  $E(m)$  in a way that allows anyone to compute  $E(f(m))$  for every function  $f$ .

# Cryptography & Security

Prev slides: Have provably secure algorithm for every crypto task imaginable.

**Q: How come nothing is secure?**

**A1: Not all of these are used or used correctly:**

- „ Strange tendency to use “home-brewed” cryptosystems.
- „ Combining secure primitives in insecure way
- „ Misunderstanding properties of crypto components.
- „ Strict efficiency requirements for crypto/security:
  - ☞ The cost is **visible** but benefit **invisible**.
  - ☞ Many provably secure algs not efficient enough
- „ Easy to get implementation wrong – many subtleties
- „ Compatibility issues, legacy systems,

# Public-Key Algorithms

**Encryption in which each party publishes a public part of their key and keep secret a private part of it**

- RSA (by Rivest, Shamir, Adleman) »

# Public-Key Algorithms (1)

## Downsides of keys for symmetric-key designs:

- Key must be secret, yet be distributed to both parties
- For N users there are  $N^2$  pairwise keys to manage

## Public key schemes split the key into public and private parts that are mathematically related:

- Private part is not distributed; easy to keep secret
- Only one public key per user needs to be managed

## Security depends on the chosen mathematical property

- Much slower than symmetric-key, e.g., 1000X
- So use it to set up per-session symmetric keys

# RSA (1)

**RSA is a widely used public-key encryption method whose security is based on the difficulty of factoring large numbers**

**Key generation:**

- Choose two large primes, p and q
- Compute  $n = p \times q$  and  $z = (p - 1) \times (q - 1)$ .
- Choose d to be relatively prime to z
- Find e such that  $e \times d = 1 \pmod{z}$
- Public key is  $(e, n)$ , and private key is  $(d, n)$

**Encryption (of k bit message, for numbers up to n):**

$$\text{Cipher} = \text{Plain}^e \pmod{n}$$

**Decryption:**

$$\text{Plain} = \text{Cipher}^d \pmod{n}$$

# RSA (2)

## Small-scale example of RSA encryption

For  $p=3$ ,  $q=11 \rightarrow n=33$ ,  $\phi=20 \rightarrow d=7$ ,  $e=3$

Plaintext (P)		Ciphertext (C)		After decryption	
Symbolic	Numeric	$P^3$	$P^3 \pmod{33}$	$C^7$	$C^7 \pmod{33}$
S	19	6859	28	13492928512	19
U	21	9261	21	1801088541	21
Z	26	17576	20	1280000000	26
A	01	1	1	1	01
N	14	2744	5	78125	14
N	14	2744	5	78125	14
E	05	125	26	8031810176	05

Sender's computation

Encryption:  $C = P^3 \pmod{33}$

Receiver's computation

Decryption:  $P = C^7 \pmod{33}$

# One-Time Pads (1)

**Simple scheme for perfect secrecy:**

- XOR message with secret pad to encrypt, decrypt
- Pad is as long as the message and can't be reused!
  - It is a “one-time” pad to guarantee secrecy

Message 1: 1001001 0100000 1101100 1101111 1110110 1100101 0100000 1111001 1101111 1110101 0101110

Pad 1: 1010010 1001011 1110010 1010101 1010010 1100011 0001011 0101010 1010111 1100110 0101011

Ciphertext: 0011011 1101011 0011110 0111010 0100100 0000110 0101011 1010011 0111000 0010011 0000101

Pad 2: 1011110 0000111 1101000 1010011 1010111 0100110 1000111 0111010 1001110 1110110 1110110

Plaintext 2: 1000101 1101100 1110110 1101001 1110011 0100000 1101100 1101001 1110110 1100101 1110011

Different secret pad decrypts to the wrong plaintext

# One-Time Pads (2)

Alice sending Bob a one-time pad with **quantum crypto**.

- Bob's guesses yield bits; Trudy misses some
- Bob can detect Trudy since error rate increases

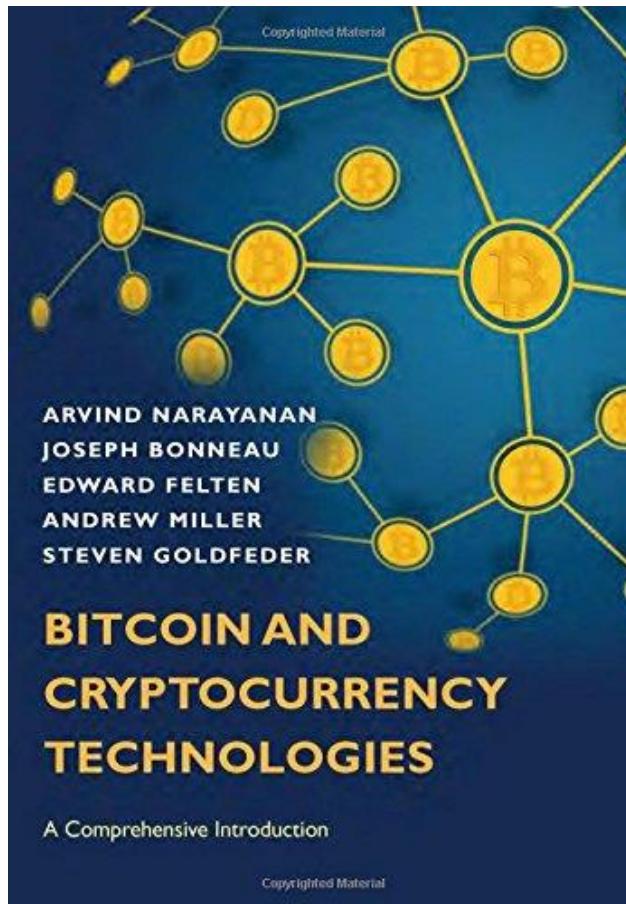
Bit number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Data	1	0	0	1	1	1	0	0	1	0	1	0	0	1	1	0	What Alice sends
	↖	↑	↑	↖	←	↖	↗	↗	↔	↓	→	↗	↗	↔	↔	↑	
	+	+	✗	✗	✗	+	+	✗	+	✗	+	✗	✗	✗	+	✗	Bob's bases
	↑	↑	↗	↖	↖	↓	→	↗	↔	↗	→	↗	↗	↖	→	↗	What Bob gets
	No	Yes	No	Yes	No	No	No	Yes	Yes	No	Yes	Yes	Yes	No	Yes	No	Correct basis?
		0		1				0	1		1	0	0		1		One-time pad
	✗	↔	+	✗	↔	+	↔	✗	+	↔	✗	✗	✗	+	✗	✗	Trudy's bases
	x	0	x	1	x	x	x	?	1	x	?	?	0	x	?	x	Trudy's pad

# Blockchain & Business Application

Lecture:

Cryptography & Cryptocurrencies

# Book's Chapter 1: Introduction to Cryptography & Cryptocurrencies



- Currencies require
  - Ways to control supply
  - Security to prevent cheating
    - Central banks do both
    - Security makes it more difficult for attacker not possible
    - Law enforcement needed!
- Cryptocurrencies (CC)
  - Pure technological methods → Cryptography
  - Requiring no authority

- Two basic primitives to construct CC
  - Cryptographic hash function
    - Hash function?
    - Hash pointers
  - Digital Signatures

# Hash Function Properties

- Input: Any string of any size.
- Output: Fixed size.
  - This lecture: 256-bit
- Efficiently computable:
  - Reasonable amount of time
  - n-bit string: Running time  $O(n)$ .

# Cryptographic Hash Function Properties

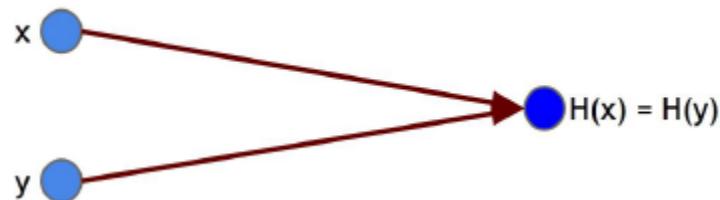
- Three additional properties:
  - 1) Collision-resistance
  - 2) Hiding
  - 3) Puzzle-friendliness
    - This is not general but required for CCs.

# Cryptographic Hash Function Properties 1

- Three additional properties:
  - 1) Collision-resistance
  - 2) Hiding
  - 3) Puzzle-friendliness

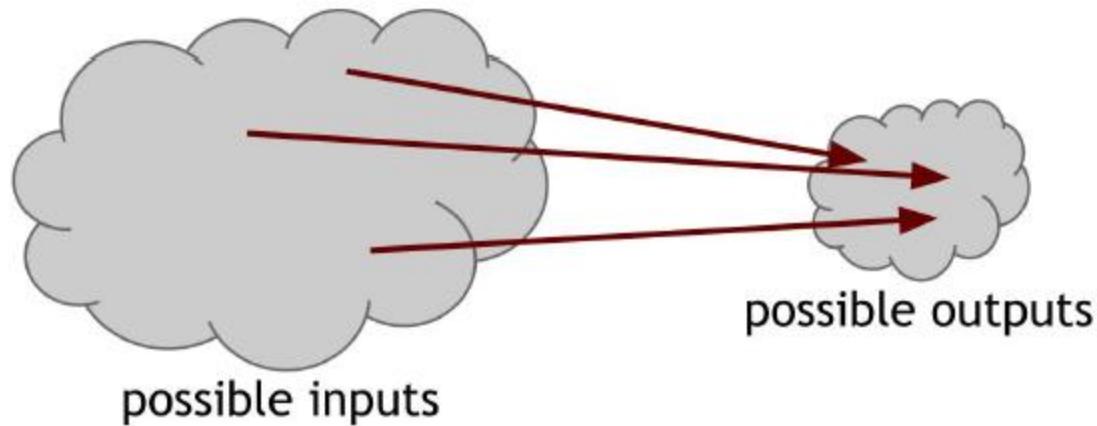
# Collision-resistance

**Collision-resistance:** A hash function  $H$  is said to be collision resistant if it is infeasible to find two values,  $x$  and  $y$ , such that  $x \neq y$ , yet  $H(x)=H(y)$ .



- The claim:
  - Nobody *can* “find”
  - Not “*it does not exist*”

# Collision-resistance



- For a hash function with a 256-bit output size:
  - pick  $2^{256} + 1$  distinct values
  - Compute their hashes
  - Check if any pair is equal. Result? ***Pigeonhole Principle***

# Collision-resistance

- For a hash function with a 256-bit output size:
  - Pick  $2^{256} + 1$  distinct values:
    - Will find collision **100%**
  - What about smaller number of values?
  - Pick  $2^{130} + 1$  distinct values:
    - Will find collision **99.% !!!**

*Birthday Paradox*

# Collision-resistance

- 100%, but can we really find collision?
- Calculate  $2^{256} + 1$  w.c. or, on avg.  $2^{128}$  Hashes
- If a computer calculates **10K Hashes / sec**
- Needs  $2^{27}$  years
- Finding one in reasonable some time..  
Infinitesimally smally small probability!

# Collision-resistance

- Counter-example

$$H(x) = x \bmod (2^{256})$$

$$x_1 = 3$$

$$x_2 = 3 + 2^{256}$$

$$H(x_1) = H(x_2) = 3$$

- So what? Research required for Hash func's!

# Collision-resistance

- ***Business Application:***
  - Online file storage systems
  - Upload, wait, download back.
  - Is it the same?
- Don't keep the original file
- Keep its Hash!

# Cryptographic Hash Function Properties 2

- Three additional properties:
  - 1) Collision-resistance
  - **2) Hiding**
  - 3) Puzzle-friendliness

# Hiding

- if we're given the output of the hash function  
 $y = H(x)$
- *there's no feasible way to figure out what the input,  $x$ , was*
- Deterministic functions → Decrease prob.
- Non-deterministic functions → Totally secure!

# Hiding

- ***Business Application:***
- **Commitments:** Digital analog of
  - taking a value,
  - sealing it in an envelope,
  - putting that envelope out on the table
- You've committed yourself
- Secret from everyone

# Hiding

- two algorithms:
  - **com := commit( msg, nonce )** The commit function takes a message and secret random value, called a *nonce*, as input and returns a commitment.
  - **verify( com, msg, nonce )** The verify function takes a commitment, *nonce*, and message as input. It returns true if  $\text{com} == \text{commit}( \text{msg} , \text{nonce} )$  and false otherwise
- Require the following two security properties hold:
  - Hiding : Given **com** , it is infeasible to find **msg**
  - Binding : It is infeasible to find two pairs  $(\text{msg}, \text{nonce})$  &  $(\text{msg}', \text{nonce}')$  such that  $\text{msg} \neq \text{msg}'$  &  $\text{commit}( \text{msg}, \text{nonce} ) == \text{commit}( \text{msg}', \text{nonce}' )$

# Hiding

- These algorithms actually behave like sealing & opening an envelope
- Every time you commit to a value, choose a new random value *nonce*.
- In cryptography, the term nonce is used to refer to a value that can only be used once.

# Cryptographic Hash Function Properties 3

- Three additional properties:
  - 1) Collision-resistance
  - 2) Hiding
  - **3) Puzzle-friendliness**

# Puzzle-friendliness

- A hash function  $H$  said to be puzzle-friendly if:
  - For every possible  $n$ -bit output value  $y$ ,
  - if  $k$  is chosen from a distribution with high min-entropy,
  - then it is infeasible to find  $x$  such that  $H(k \parallel x) = y$  in time significantly less than  $2^n$ .

# (min-entropy)

- A measure of how predictable an outcome is
- High: the distribution is very spread out.
- Means:
  - when we sample from the distribution, there's no particular value that's likely to occur.
- A concrete example:
  - if  $r$  is chosen uniformly from among 256-bit strings
  - then any particular string was chosen with probability  $1/2^{256}$  : infinitesimally small value.

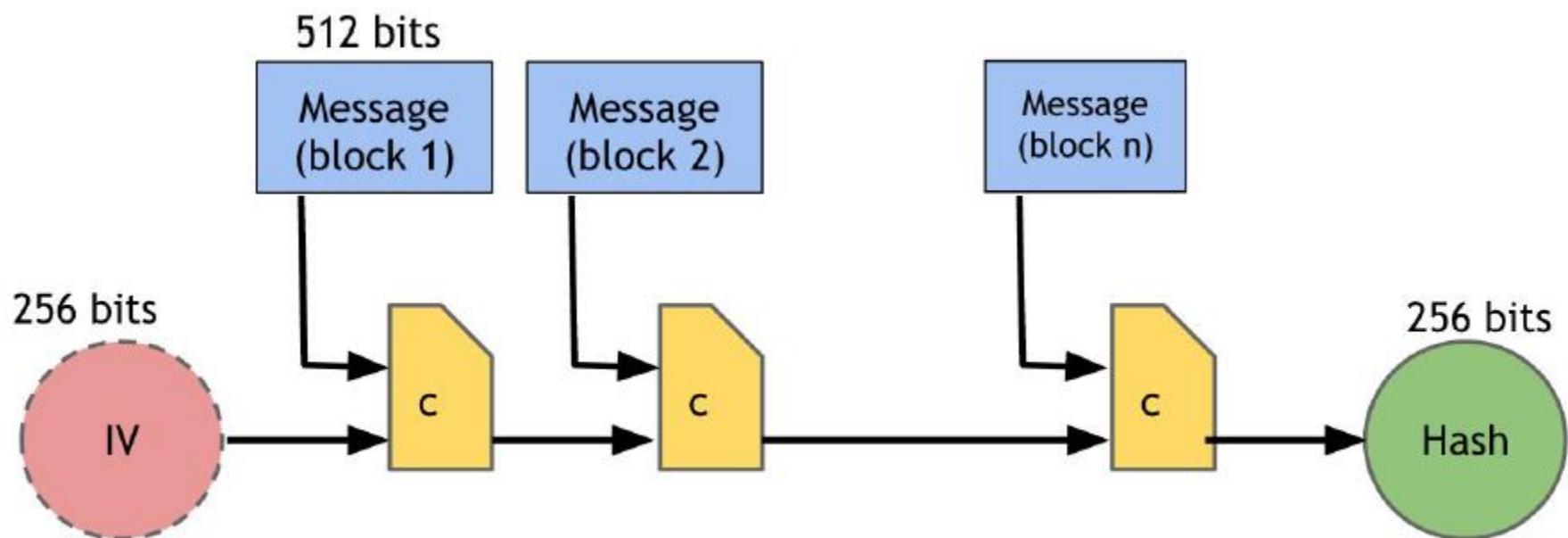
# Puzzle-friendliness

- ***Business Application:***
  - Search puzzles
  - It has no *shortcut (honeybee)*
  - No method to solve
  - Just search the whole space

# SHA-256

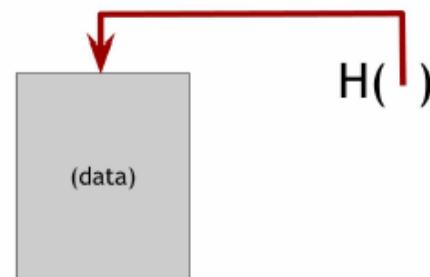
- A particular Hash function
- Bitcoin uses!
- Input size: fixed or arbitrary?
- We require arbitrary
  - Merkle-Damgard transform
- Compression function .
  - Proven: If the underlying compression function is collision resistant, then the overall hash function is collision resistant as well.

# How SHA-256 works (simplified)



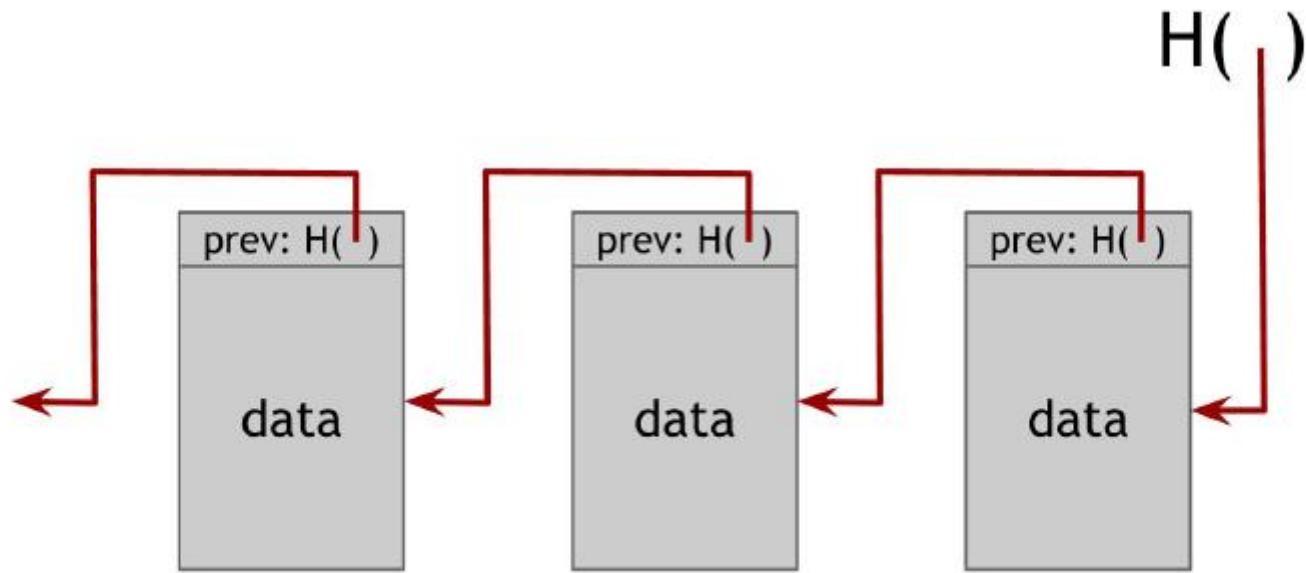
# Hash Pointers & Data Structures

- Regular pointer gives you a way to retrieve the information
- A hash pointer also gives you a way to verify that the information hasn't changed



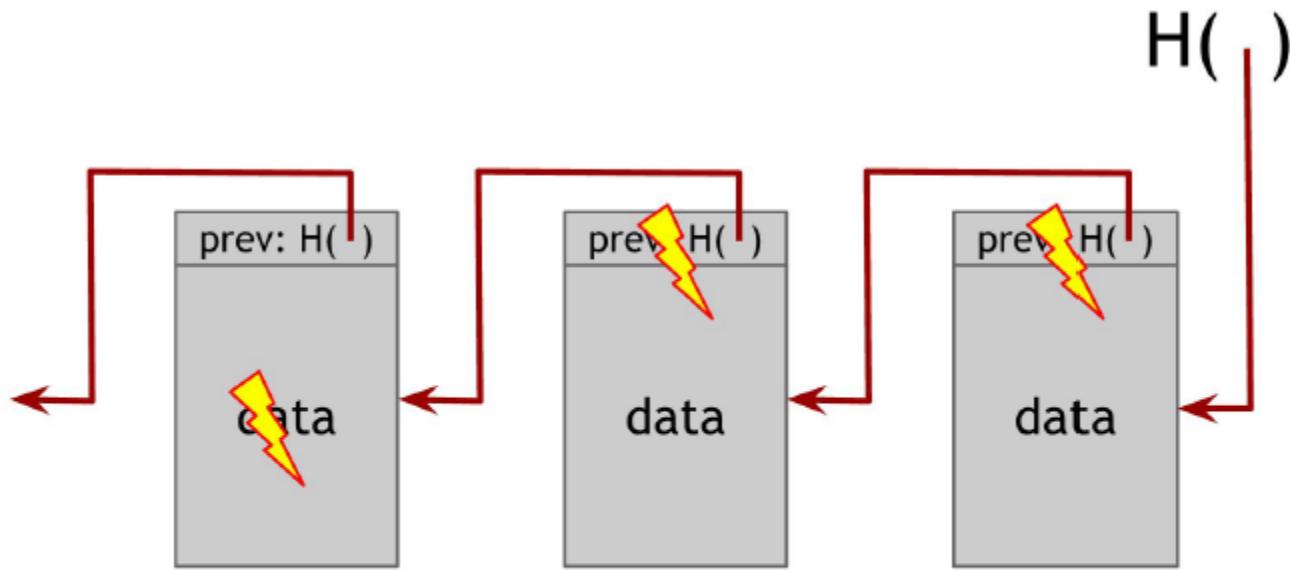
**Figure 1.4 Hash pointer.** A hash pointer is a pointer to where data is stored together with a cryptographic hash of the value of that data at some fixed point in time.

# Blockchain



**Figure 1.5 Block chain.** A block chain is a linked list that is built with hash pointers instead of pointers.

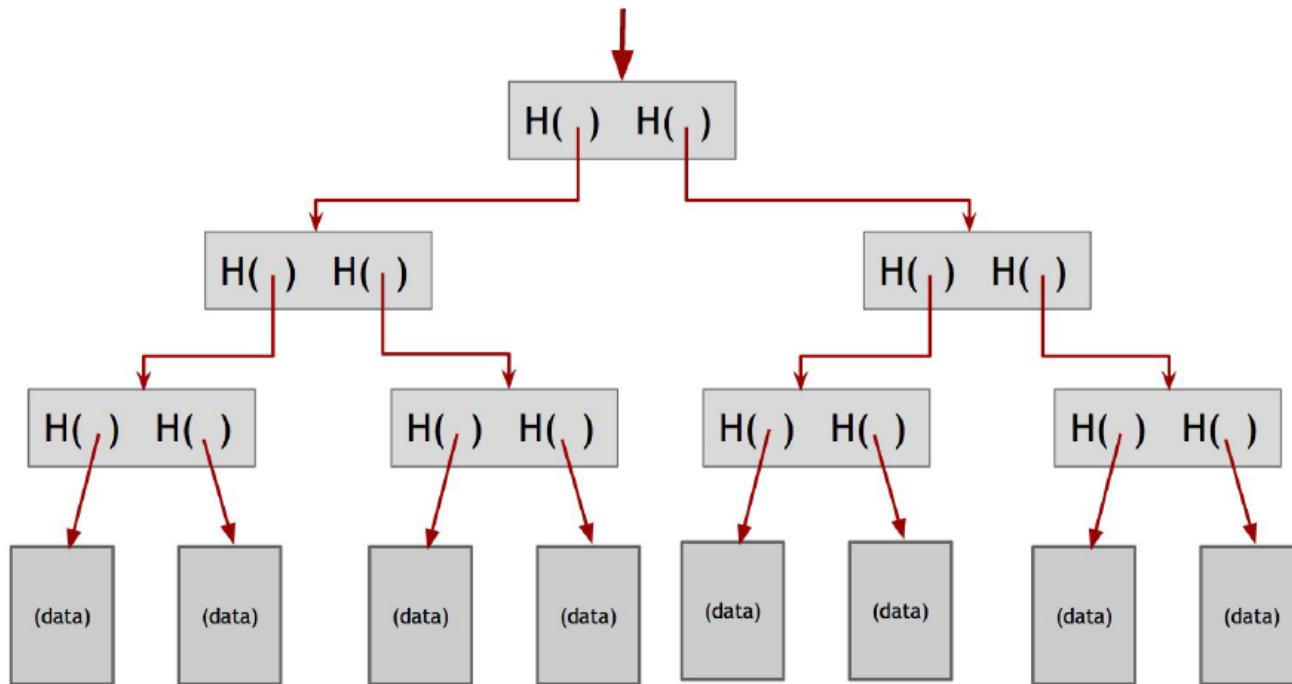
# Blockchain



**Figure 1.6 Tamper-evident log.** If an adversary modifies data anywhere in the block chain, it will result in the hash pointer in the following block being incorrect. If we store the head of the list, then even if the adversary modifies all of the pointers to be consistent with the modified data, the head pointer will be incorrect, and we will detect the tampering.

# Merkle Trees

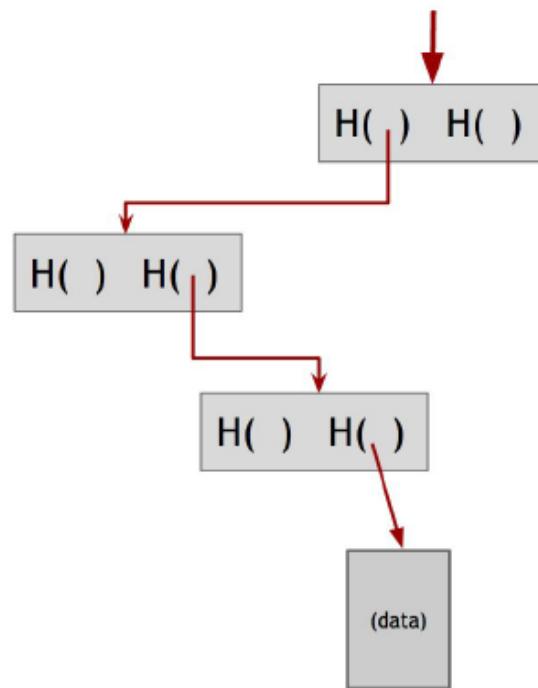
- A binary tree with hash pointers



**Figure 1.7 Merkle tree.** In a Merkle tree, data blocks are grouped in pairs and the hash of each of these blocks is stored in a parent node. The parent nodes are in turn grouped in pairs and their hashes stored one level up the tree. This continues all the way up the tree until we reach the root node.

# Merkle Trees & Proof of Membership

- *n nodes in tree: only  $\log(n)$  items to show*



**Figure 1.8 Proof of membership.** To prove that a data block is included in the tree, one only needs to show the blocks in the path from that data block to the root.

# Proof of Non-membership

- Sorted Merkle Trees!
- Verify the non-membership requires
  - Only logarithmic time&space!

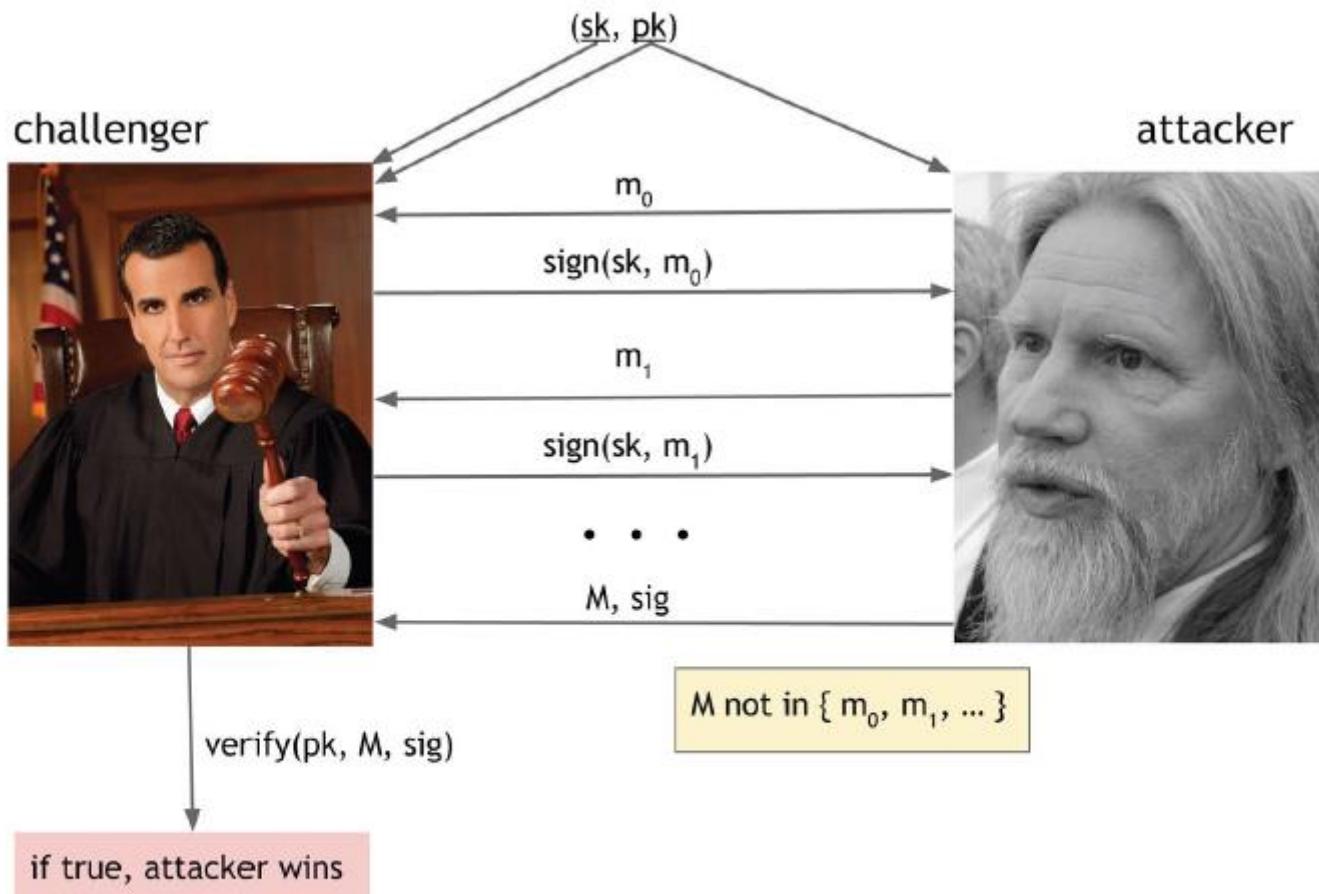
- Two basic primitives to construct CC
  - Cryptographic hash function
    - Hash function?
    - Hash pointers
  - **Digital Signatures**

# Digital Signatures

- Handwritten signature:
  - Only you can make your signature, but anyone who sees it can verify that it's valid.
  - Signature to be tied to a particular document
- Digital signatures?

- Three algorithms:
  - $(\text{sk}, \text{pk}) := \text{generateKeys}(\text{keysize})$
  - $\text{sk}$ : secret key, used to sign messages.
  - $\text{pk}$ : public verification key.
  - $\text{sig} := \text{sign}(\text{sk}, \text{message})$
  - $\text{isValid} := \text{verify}(\text{pk}, \text{message}, \text{sig})$
- We require that following two properties hold:
  - Valid signatures must verify  
 $\text{verify}(\text{pk}, \text{message}, \text{sign}(\text{sk}, \text{message})) == \text{true}$
  - Signatures are existentially unforgeable

# Unforgeability



**Figure 1.9 Unforgeability game.** The adversary and the challenger play the unforgeability game. If the attacker is able to successfully output a signature on a message that he has not previously seen, he wins. If he is unable, the challenger wins and the digital signature scheme is unforgeable.

# Practical Concerns

- Algorithm → Digital Signature  
Science → Business Application → Science
- Randomness
  - Ultimate requires non-deterministic (quantum)
- Message size!
  - Sign the Hash of the msg, not msg!

# Business Application: Signature in Bitcoin

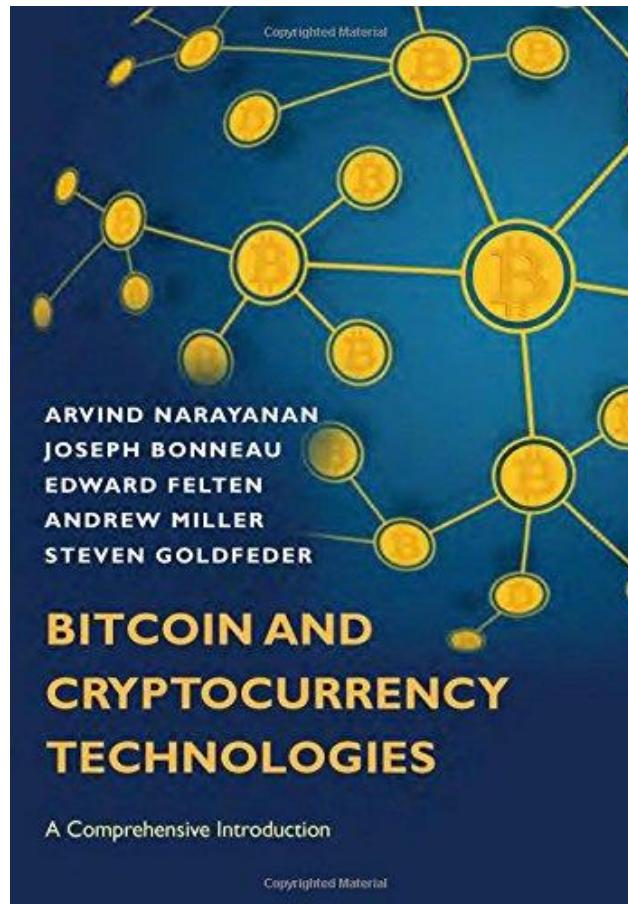
ECDSA: Elliptic Curve Digital Signature Algorithm

- US Government standard
- Improvement over DSA
- Bitcoin uses ECDSA over secp256k1
  - 128 bit
  - To break,  $2^{128}$  cryptographic operations required

# Blockchain & Business Application

Lecture:  
Consensus Protocols

# Chapter 2: How Bitcoin Achieves Decentralization



# Centralization vs. Decentralization

- the Internet, a famously decentralized sys.
- E-mail/Msgs: decentralized or centralized?
  - Simple Mail Transfer Protocol (SMTP)
  - Facebook, LinkedIn?
- Decentralization is not all or nothing!
- Bitcoin: Decentralized
  - Exchange, wallet softwares?

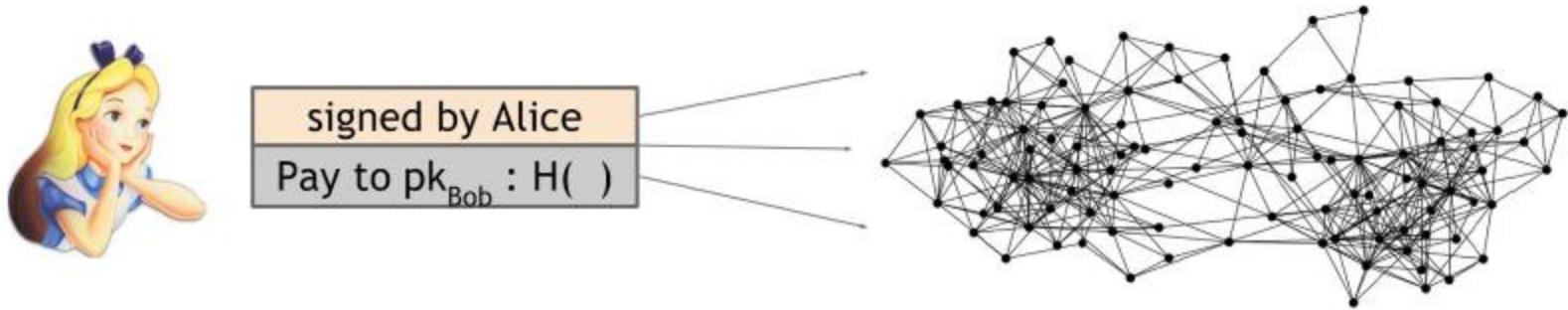
# How the Bitcoin protocol achieves decentralization

1. Who maintains the ledger of transactions?
  2. Who has authority over which transactions are valid?
  3. Who creates new bitcoins?
  4. Who determines how the rules of the system change?
  5. How do bitcoins acquire exchange value?
- 
- Questions 1-3: technical; focus of this chapter

- Different aspects of Bitcoin fall on different points on the centralization/decentralization spectrum.
  - The peer-to-peer network is close to purely decentralized: Anybody can run a Bitcoin node: fairly low barrier to entry.
  - Bitcoin mining: also *technically* open to anyone, but it requires a very high capital cost.
    - High degree of centralization / concentration of power
  - Updates to the software that Bitcoin nodes run

# Decentralization → Distributed Consensus

- Distributed consensus protocol:
- There are  $n$  nodes that each have an input value. Some of these nodes are faulty or malicious.
- A distributed consensus protocol has the following two properties:
  - It must terminate with all honest nodes in agreement on the value
  - The value must have been generated by an honest node



**Figure 2.1 Broadcasting a transaction** In order to pay Bob, Alice broadcasts the transaction to the entire Bitcoin peer-to-peer network.

- Where is Bob?
- a variety of users are broadcasting these transactions to the network:
- the nodes must agree on exactly which transactions were broadcast &
- the order in which these transactions happened
- single, global ledger for the system

- At any given point, all the nodes in the peer-to-peer network have a **ledger**
- consisting of a sequence of blocks,
  - each containing a list of transactions,
  - that they've reached consensus on.

- Additionally, each node has a pool of outstanding transactions
  - that it has heard about but have not yet been included on the block chain
  - consensus has not yet happened
- 
- Each node might have a slightly different version of the outstanding transaction pool

# How do nodes come to consensus on a block

- One way:
  - at regular intervals, say every 10 minutes,
  - every node in the system proposes its own outstanding transaction pool to be the next block.
  - nodes execute some *consensus protocol*,
    - each node's input is its own proposed block
  - some nodes may be malicious
    - put invalid transactions into their blocks
  - assume others are honest

- If the consensus protocol succeeds, a valid block will be selected as the output.
- Even if the selected block was proposed by only one node, it's a valid output if the block is valid.
- some valid outstanding transaction may not get included in the block,
- it could just wait and get into the next block
- This is similar to Bitcoin, but not exactly same

# Problems of this approach

- Consensus: A hard problem nodes might
  - crash / be malicious
- Network is highly imperfect
  - Not a complete graph but peer-to-peer
  - Poor internet connection
- Huge latency
  - Distributed to all over the Internet

- Bitcoin protocol must reach consensus in the face of two types of obstacles:
  - Imperfections in the network
    - latency (global time?)
    - nodes crashing
  - Deliberate attempts by some nodes

# Impossibility results

- The lack of global time heavily constrains
- Byzantine Generals Problem:
  - Army divisions, commanded by generals
  - Generals communicate by messenger on joint plan
  - Some generals may be traitors, prevent joint plan
  - Loyals try for consensus
  - Proven:  $1/3$  traitor threshold!
- Bad news!

# Byzantium Empire



# Good news?

- Impossibility proofs are for specific models
- Bitcoin is beyond traditional assumptions
- It runs better in practice than in theory!
  - Incentives: currency → natural to act honestly
    - Consensus for currency, not general
  - Randomness
    - Consensus in long time (practically 1 hour)

# Consensus without identity

- Sybil attack:
  - Sybils are just copies of nodes that a malicious adversary can create
  - to look like there are a lot of different participants,
  - in fact all those controlled by the same adversary
- No identity
  - Lottery, ticket etc.
  - Random node selection
  - All Sybils may be treated as one node!

# Bitcoin consensus algorithm (simplified)

- Simplified: assumes selecting a random node is not vulnerable to Sybil attacks
  1. New transactions are broadcast to all nodes
  2. Each node collects new transactions into a block
  3. In each round a random node gets to broadcast its block
  4. Other nodes accept the block only if all transactions in it are valid (unspent, valid signatures)
  5. Nodes express their acceptance of the block by including its hash in the next block they create

Why does this work?

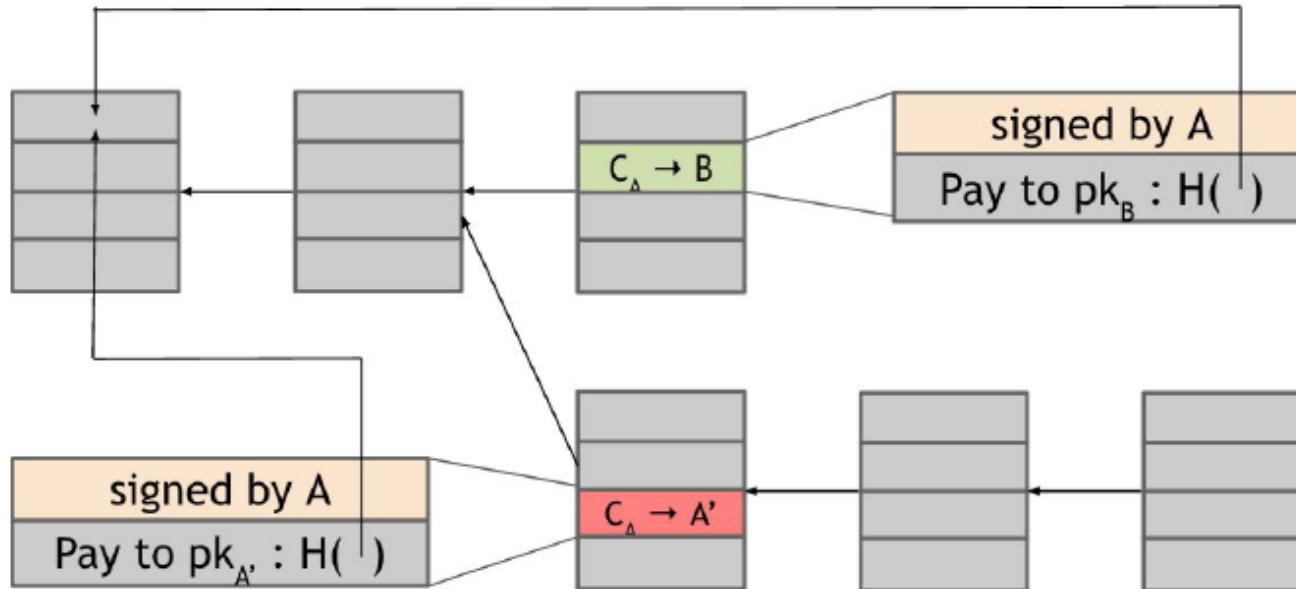
# Consider malicious adversary: Alice

- **Stealing Bitcoins:**
  - Can Alice simply steal bitcoins belonging to another user at an address she doesn't control?
  - No. Even if it is Alice's turn to propose the next block, she cannot steal!
  - It would require Alice to create a valid transaction that spends that coin.
  - It would require Alice to forge the owners' signatures - → secure digital signature scheme.
  - IF underlying cryptography is solid, she cannot steal

# Consider malicious adversary: Alice

- **Denial of service attack.**
  - Say Alice really dislikes some other user Bob.
  - Alice can decide to include no transactions originating from Bob's address in any block that she proposes
  - While this is a valid attack that Alice can try to mount, luckily it's nothing more than a minor annoyance.
  - If Bob's transaction doesn't make it into the next block by Alice, he will just wait until an honest node
  - So that's not really a good attack either.

# Alice: Double Spend Attempt

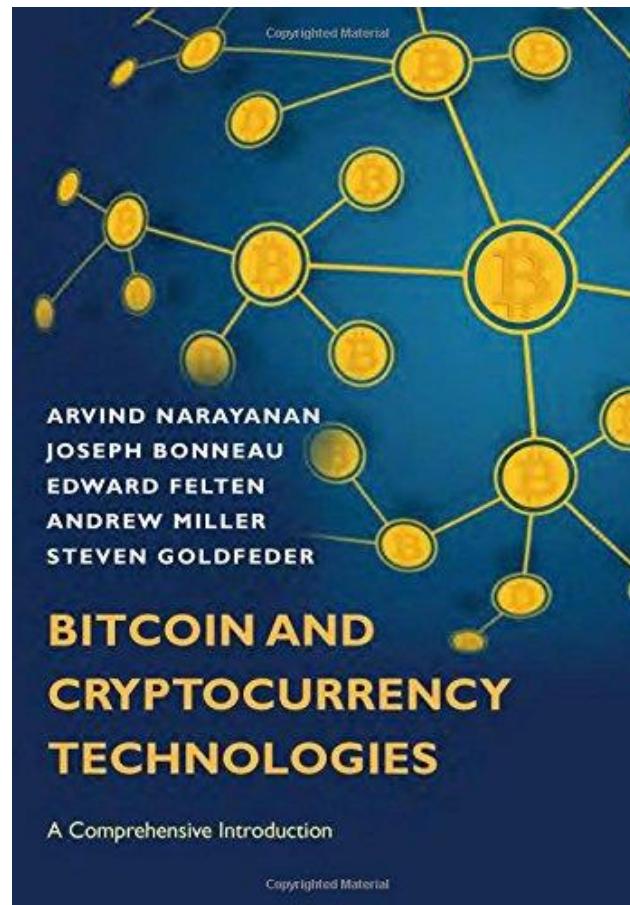


**Figure 2.2 A double spend attempt.** Alice creates two transactions: one in which she sends Bob Bitcoins, and a second in which she double spends those Bitcoins by sending them to a different address that she controls. As they spend the same Bitcoins, only one of these transactions can be included in the block chain. The arrows are pointers from one block to the previous block that it extends including a hash of that previous block within its own contents.  $C_A$  is used to denote a coin owned by Alice.

# Blockchain & Business Application

Lecture:  
Mechanics of Bitcoin

# Chapter 3: Mechanics of Bitcoin



- First chapters
  - High level
- This chapter:
  - Real data structures
  - Real scripts
  - Language of Bitcoin
  - Challenging!

# Bitcoin Transactions

- Simplified model of a ledger.
- Instead of blocks, let's suppose:
  - Individual transactions are added to the ledger one at a time.

Transfer 17 coins from Alice to Bob	SIGNED(Alice)
Transfer 8 coins from Bob to Carol	SIGNED(Bob)
Transfer 5 coins from Carol to Alice	SIGNED(Carol)
Transfer 15 coins from Alice to David	SIGNED(Alice)

- Transactions OK;
- Balance?

Create 25 coins and credit to Alice	ASSERTED BY MINERS
Transfer 17 coins from Alice to Bob	SIGNED(Alice)
Transfer 8 coins from Bob to Carol	SIGNED(Bob)
Transfer 5 coins from Carol to Alice	SIGNED(Carol)
Transfer 15 coins from Alice to David	SIGNED(Alice)

**Figure 3.1** an account-based ledger

- Balance?
- Does Alice have 15 coins to transfer to David?

Create 25 coins and credit to Alice	ASSERTED BY MINERS
Transfer 17 coins from Alice to Bob	SIGNED(Alice)
Transfer 8 coins from Bob to Carol	SIGNED(Bob)
Transfer 5 coins from Carol to Alice	SIGNED(Carol)
Transfer 15 coins from Alice to David	SIGNED(Alice)

**Figure 3.1** an account-based ledger

1	Inputs: Ø Outputs: 25.0→Alice
2	Inputs: 1[0] Outputs: 17.0→Bob, 8.0→Alice <small>SIGNED(Alice)</small>
3	Inputs: 2[0] Outputs: 8.0→Carol, 9.0→Bob <small>SIGNED(Bob)</small>
4	Inputs: 2[1] Outputs: 6.0→David, 2.0→Alice <small>SIGNED(Alice)</small>

**Figure 3.2** a transaction-based ledger, which is very close to Bitcoin

# Basic Issues

- Change Address
- Efficient verification
- Consolidating Funds
- Joint Payments
- Transaction Syntax

- Change Address: Why Alice sends to herself?

1	Inputs: Ø Outputs: 25.0→Alice
2	Inputs: 1[0] Outputs: 17.0→Bob, 8.0→Alice SIGNED(Alice)
3	Inputs: 2[0] Outputs: 8.0→Carol, 9.0→Bob SIGNED(Bob)
4	Inputs: 2[1] Outputs: 6.0→David, 2.0→Alice SIGNED(Alice)

*Figure 3.2* a transaction-based ledger, which is very close to Bitcoin

- Change Address
- Efficient verification
  - Hash pointers
  - Sufficient to scan only referenced → latest block
- Consolidating Funds
  - Bob receives 17 and 2, how to sum up to 19?
  - Create new self trans. with 2 inputs, 1 output
- Joint Payments
- Transaction Syntax

- Change Address
- Efficient verification
- Consolidating Funds
- Joint Payments
  - Carol and Bob pays to Alice: 2 Inp (signs), 1 Outp;
- Transaction Syntax
  - Each transaction: Bit strings

```
{  
    "hash": "5a42590fbe0a90ee8e8747244d6c84f0db1a3a24e8f1b95b10c9e050990b8b6b",  
    "ver": 1,  
    "vin_sz": 2,  
    "vout_sz": 1,  
    "lock_time": 0,  
    "size": 404,  
    "in": [  
        {  
            "prev_out": {  
                "hash": "3be4ac9728a0823cf5e2deb2e86fc0bd2aa503a91d307b42ba76117d79280260",  
                "n": 0  
            },  
            "scriptSig": "30440..."  
        },  
        {  
            "prev_out": {  
                "hash": "7508e6ab259b4df0fd5147bab0c949d81473db4518f81afc5c3f52f91ff6b34e",  
                "n": 0  
            },  
            "scriptSig": "3f3a4ce81...."  
        }  
    ],  
    "out": [  
        {  
            "value": "10.12287097",  
            "scriptPubKey": "OP_DUP OP_HASH160 69e02e18b5705a05dd6b28ed517716c894b3d42e  
                           OP_EQUALVERIFY OP_CHECKSIG"  
        }  
    ]  
}
```

- **Metadata:** some housekeeping information
  - the size of the transaction,
  - the number of inputs,
  - the number of outputs.
  - the hash of the entire transaction
    - serves as a unique ID for the transaction.
    - it allows us to use hash pointers to reference trans's
  - Finally there's a “lock\_time”.. later.

- Input(s):
  - form an array, each input has the same form.
  - specifies a previous transaction, so it contains a hash of that transaction, which acts as a hash pointer to it.
  - also contains the index of the previous transaction's outputs that's being claimed.
  - signature: we have to sign to show that we actually have the ability to claim those previous transaction outputs

- Output(s):
  - again an array.
  - each has just two fields.
  - each have a value,
    - the sum of all the output values has to be less than or equal to the sum of all the input values.
    - If the sum of the output values is less than the sum of the input values, the difference is a transaction fee to the miner who publishes this transaction
  - recipient address: pk, hash, commands..

# Outline

- Bitcoin Scripts
  - Technical aspects
  - Business applications
- Managing & Protecting coins
  - Local storage
  - Hot & Cold storage

# Bitcoin Scripts

- Each transaction output specifies
  - A script, not just a “public key”
- We want it to say:

“this can be redeemed by a signature from the owner of address X”.. But!
- Address: Hash of a public key
  - Does not tell the key
  - So, how to check the signature?

# Transaction outputs

- Hence, transaction must say:
  - “this can be redeemed by a public key that hashes to X, along with a signature from the owner of that public key”

```
OP_DUP  
OP_HASH160  
69e02e18...  
OP_EQUALVERIFY  
OP_CHECKSIG
```

an example Pay-to-PubkeyHash script

# Bitcoin Scripts

- Script Language built specifically for Bitcoin, called “Script”
  - Stack-based
  - Simple & Compact
  - Native support for cryptographic operations
    - Compute Hash functions
    - Compute signatures
    - Verify signatures

# Script

- Stack-based
  - Each instruction executed once
  - No loops
  - Number of instructions
    - How long it can take
    - How much memory it can use
  - Not Turing-complete, by design!
    - Arbitrary miners can't submit infinite-loops



```
for i in range(0,10,2):  
    print i
```

```
0  
2  
4  
6  
8
```

- Only two possible outcomes:
  - Execute normally with no errors → Valid,
  - Error → Invalid transaction
- Each instruction represented by 1 Byte
  - 256 in total ( $2^8=256$ )
  - 15 disabled, 75 reserved (to be added later)

- Instructions
  - Basic arithmetic
  - Basic logic (if-then)
    - (Not) throwing errors
  - Crypto instructions
    - Hash functions
    - Signature operations
    - CHECKMULTISIG: Check multiple with one instruction

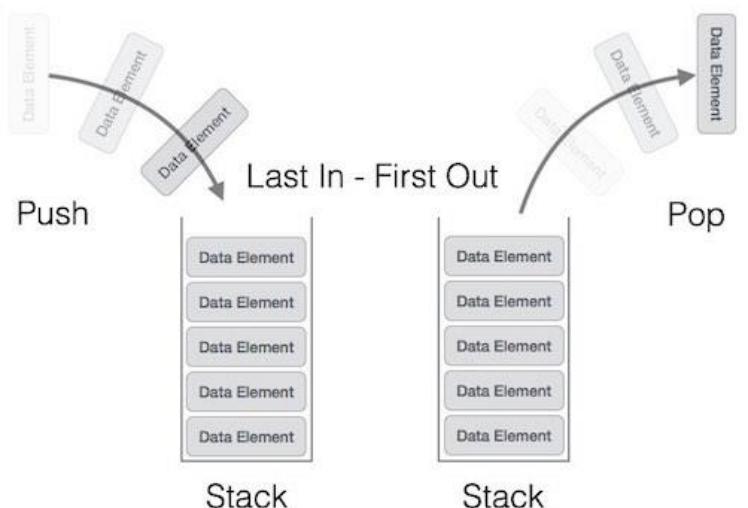
- CHECKMULTISIG requires specifying
  - n public keys,
  - and a parameter t , for a threshold.
- For this instruction to execute validly,
  - at least t signatures from t out of n of those valid pk
  - t out of n specified entities must sign in order for the transaction to be valid.

<b>OP_DUP</b>	Duplicates the top item on the stack
<b>OP_HASH160</b>	Hashes twice: first using SHA-256 and then RIPEMD-160
<b>OP_EQUALVERIFY</b>	Returns true if the inputs are equal. Returns false and marks the transaction as invalid if they are unequal
<b>OP_CHECKSIG</b>	Checks that the input signature is a valid signature using the input public key for the hash of the current transaction
<b>OP_CHECKMULTISIG</b>	Checks that the $k$ signatures on the transaction are valid signatures from $k$ of the specified public keys.

**Figure 3.6** a list of common Script instructions and their functionality.

# Executing a Script

- Stack that we can push data to & pop data from.
- We won't need any other memory or variables.
- Two types of instructions:
  - data instructions (data pushed onto the top)
  - OpCodes (take input data from top)

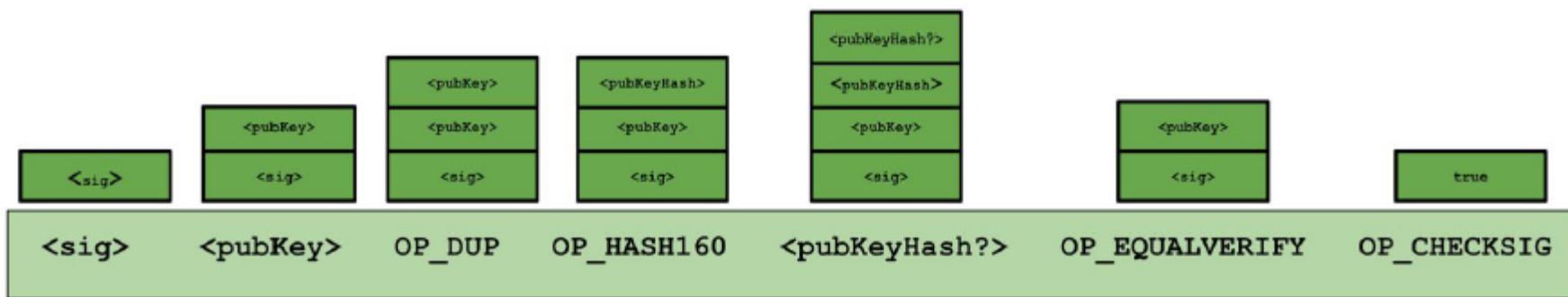


```

<sig>
<pubKey>

-----
OP_DUP
OP_HASH160
<pubKeyHash?>
OP_EQUALVERIFY
OP_CHECKSIG

```



**Figure 3.7 Execution of a Bitcoin script.** On the bottom, we show the instruction in the script. Data instructions are denoted with surrounding angle brackets, whereas opcodes begin with “OP\_”. On the top, we show the stack just after that instruction has been executed.

# Theory vs. Practice

- Theory: Specify arbitrary conditions
- Practice: 99.9% of Bitcoin transaction history
  - The same!
  - The above script!
- Nodes have a “whitelist” of transactions

# Business Applications of Bitcoin Scripts

- Escrow Transactions
  - Alice to pay Bob in Bitcoin (after receiving goods)
  - Bob to send physical item (after receiving Bitcoins)
  - Alice doesn't send directly to Bob, rather
    - Alice, Bob & arbitrator Judy
    - This transaction added to blockchain
    - Coins held in escrow between Alice, Bob, Judy
    - 2-of-3 can specify where coins should go

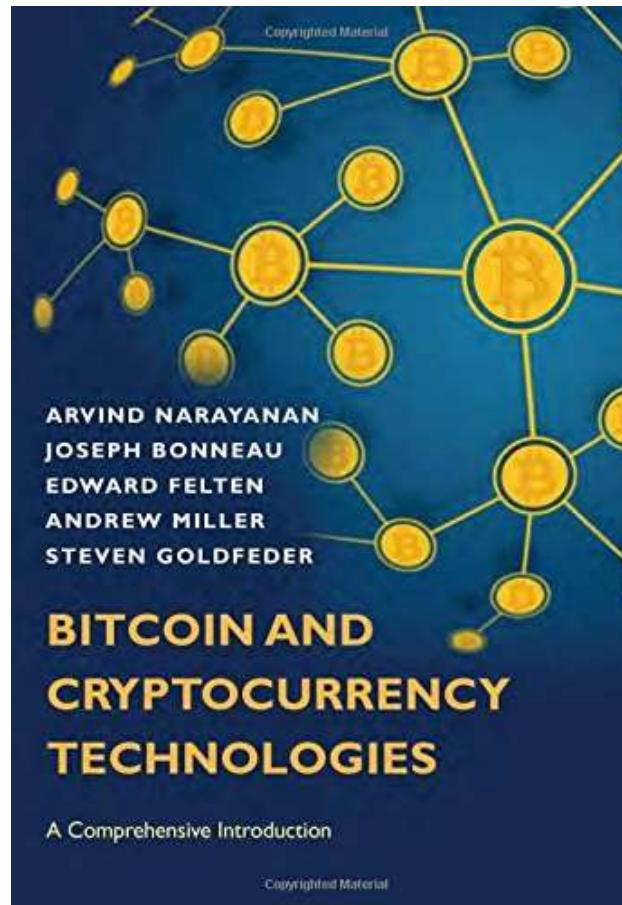
- Bob is convinced to deliver the item
- Normal case
  - Both are honest
  - Both sign a transaction
  - Redeeming coins from escrow to Bob
  - No dispute
  - No need to Judy to involve
  - Not much less efficient than direct transaction
  - Requires just one more transaction on the blockchain

- ***Green addresses.***
  - *Bob is offline/can't wait for 1 hour*
  - *Bob is a street food seller*
  - Introduction of “*Bank*”
  - Not a blockchain but a real life guarantee
- **Micro-payments**
- **Smart Contracts**

# Blockchain & Business Application

Lecture:  
Managing and Protecting  
Bitcoin Assets

# Chapter 4: How to Store and Use Bitcoins



# Simple Local Storage

- To spend a bitcoin, you need to know
  - some public info
    - Identity of the coin
    - How much it's worth, etc
  - some secret info
    - Secret key (sk) of the owner of bitcoin (you)
  - You can always get the public info
  - It's all about storing and managing your keys

- Three goals:
  - Availability
    - Capability of spending anytime
  - Security
    - Nobody else can spend your coins
  - Convenience
    - Key management should be easy
- Challenging, trade-offs

- Simplest: Carry on your computer/phone
  - Convenient!
  - Not always available
  - Not always secure
- Wallet software
  - Keeps track of your coins
  - Manages keys

- Encoding keys: base58 and QR codes
  - To spend or receive bitcoins, you also need a way to exchange an address with the other party
  - Two main ways addresses are encoded can be communicated from receiver to spender:
    - as a text string or
    - as a QR code.

- base58:
  - Upper & lower case letters & numbers
  - Excluding some confusing: {O, 0}

1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa

- Barcode



- Vanity addresses:
  - Satoshi Bones (gambling website)
  - 1bonesEeTcABPjLzAb1VkFgySY6Zqu3sX
  - Starts with 1: pay-to-pubkey-hash
  - How to “find” this address?

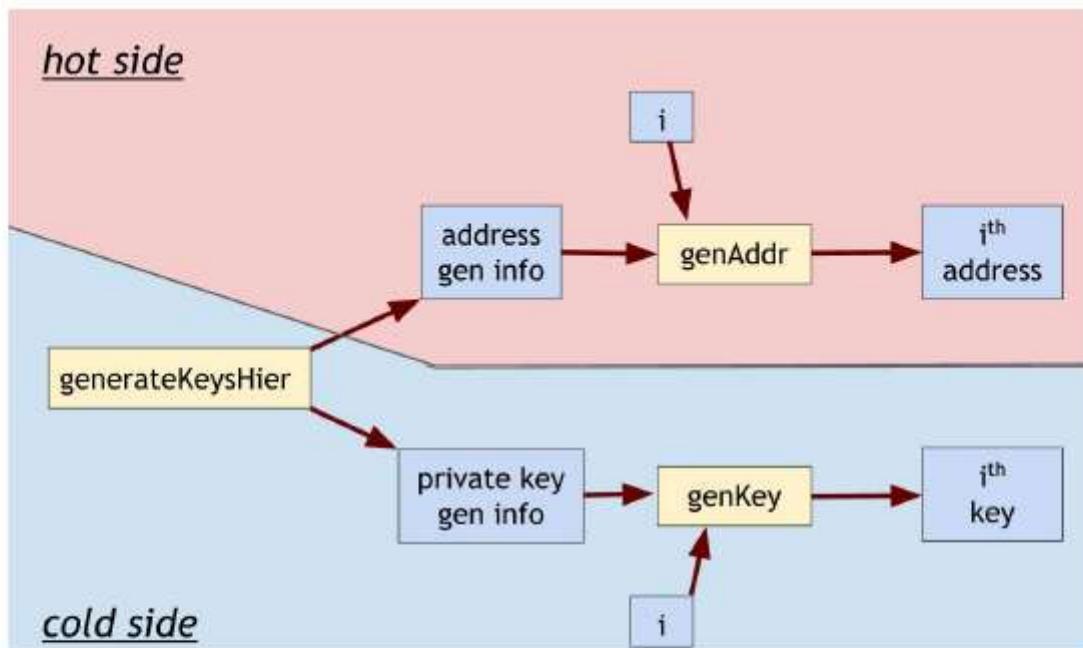
# Hot/Cold Storage

- Pocket: **Hot** (convenient but risky)
- Locked somewhere offline: **Cold**
  - Safer
  - More Secure
  - Not convenient
- Separate keys for each, why?

- Cold: Offline
  - Can't connect hot&cold
  - But Cold can receive coins
  - Can transfer Hot → Cold anytime

- Problem managing Cold addresses:
  - Privacy: Receive each coin at a fresh address
  - Cold is offline: How to manage
  - Solution:
    - Create a bunch of addresses for Cold
    - Use one each
    - Drawback: Connect to repeat periodically

- More effective Solution:
  - Hierarchical wallets
    - Cold: Use unbounded addresses
    - Hot learns these via short&one-time comm.
    - Requiring more cryptography
    - generateKeys that generates a public key, different
  - Some DS don't support hierarchical key generation
  - ECDSA (Bitcoin) supports.



**Figure 4.2: Schema of a hierarchical wallet.** The cold side creates and saves private key generation info and address generation info. It does a one-time transfer of the latter to the hot side. The hot side generates a new address sequentially every time it wants to send coins to the cold side. When the cold side reconnects, it generates addresses sequentially and checks the block chain for transfers to those addresses until it reaches an address that hasn't received any coins. It can also generate private keys sequentially if it wants to send some coins back to the hot side or spend them some other way.

- Different approaches for Cold
  - Store in a device, keep it safe (lock it up)
  - Brain Wallet
    - Control access of coins via passphrase (no device)
    - Poor physical security (traveling, etc)
  - Have an algorithm to turn passphrase to keys
  - Hash functions ☺ passphrase → private key
- Security: Can an adversary guess passphrase?

- Generating memorable passphrases.
  - One procedure that gives 80 bits of entropy:
    - Pick a random sequence of 6 words from among the 10,000 most common English words
    - $6 \times \log_2(10000)$  is roughly 80.  
*worn till alloy focusing okay reducing  
earth dutch fake tired dot occasions*
  - Easier than whole random letters, but security?

- Key- stretching :
  - Deliberately slow the function
  - e.g. use SHA-256 but compute  $2^{20}$  iterations
    - pros, cons?
- If brain-wallet gone, coins gone forever!

- Paper Wallet



**Figure 4.3:** A Bitcoin paper wallet with the public key encoded both as a 2D barcode and in base 58 notation. Observe that the private key is behind a tamper-evident seal.

# Splitting and Sharing Keys

- Too technical, only for who is interested
- Research opportunity
  - Threshold cryptography
  - Multi-signatures

PHYSICAL REVIEW A 71, 012314 (2005)

## Threshold quantum cryptography

Yuuki Tokunaga,<sup>1,2,\*</sup> Tatsuaki Okamoto,<sup>1</sup> and Nobuyuki Imoto<sup>2</sup>

<sup>1</sup>NTT Information Sharing Platform Laboratories, NTT Corporation, 1-1 Hikari-no-oka, Yokosuka, Kanagawa 239-0847, Japan

<sup>2</sup>Division of Materials Physics, Graduate School of Engineering Science, Osaka University, Toyonaka, Osaka 560-8531, Japan

(Received 9 August 2004; published 10 January 2005)

We present the concept of *threshold collaborative unitary transformation* or *threshold quantum cryptography*, which is a kind of quantum version of threshold cryptography. Threshold quantum cryptography states that *classical* shared secrets are distributed to several parties and a subset of them, whose number is greater than a threshold, collaborates to compute a quantum cryptographic function, while keeping each share secretly inside each party. The shared secrets are reusable if no cheating is detected. As a concrete example of this concept, we show a distributed protocol (with threshold) of conjugate coding.

DOI: 10.1103/PhysRevA.71.012314

PACS number(s): 03.67.Dd

# Online Wallets & Exchanges

- Online wallets:
  - like local, you might manage yourself,
  - except the information is stored in the cloud,
  - access it using a web interface / an app.
  - Popular: Coinbase and blockchain.info.
- Security:
  - Password protected
  - You need to trust

- Bitcoin Exchanges
  - Similar to traditional banking
  - Accepts fiat currency & bitcoins
  - Promise to give back on demand
  - Exchange between fiat&crypto currencies
    - Match customers

- Exchange example:
  - I have 5000\$ and 3 bitcoins
  - I put in an order to buy 2 bitcoins for 580\$ each
  - If the exchange finds someone, transaction OK
  - Now I have 3840\$ and 5 bitcoins.
- No real transaction occurred, no new block
- Only promises changed

- Pros: Connect the Bitcoin economy
- Cons: Risks (bank risks)
  - Bank run: Too many people show up
  - Ponzi scheme
  - Hack

- Statistics:
  - (by 2013) 18 of 40 exchanges closed!
  - “Mt. Gox”: Japanese, largest exchange bankrupted
    - Still in US&Jp courts
- Regulations for traditional banks
  - Minimum reserve requirement; %3-10 @USA
- For Bitcoin exchanges
  - None!

- Proof of Reserve
  - To give some comfort to customers
  - Makes a self-transaction
    - Not a proof: Just that there's someone willing to coop
    - Under-claim possible (transaction: 10K but actual 15K)

# Payment Services

- Store OK
- Manage OK
- Spend?
- Merchants
  - Better accept ☺
  - Probably immediately exchange to \$
  - Should not worry much about the technology
  - Low risk

- Risks
  - Website may go down → \$
  - Security of handling coins
  - Exchange rate fluctuate
- Payment services aim to address these risks
  - Provide interface for generating a pay-with-Bitcoin button → generate a HTML snippet to embed

Choose A Way To Accept Bitcoin or [see examples](#) of each payment method.

Type  Button  Hosted Page  iFrame  Email invoice

Payment  Buy now  Donation  Subscription

Button Style

Pay with Bitcoin 

Pay with Bitcoin 

Pay With Bitcoin

Pay With Bitcoin

Item Name

Alpaca Socks

Amount

BTC  0.00

Item Description

The ultimate in lightweight footwear

Send Funds To

My Wallet (0.00 BTC) 

[Show Advanced Options](#)

[Generate Button Code](#)

- Process of receiving Bitcoin payments through a payment service might look like this to the merchant:
  - 1. The merchant goes to payment service website and fills out a form describing the item, price, and presentation of the payment widget, etc. (example: Coinbase.)
  - 2. The payment service generates HTML code that the merchant can drop into their website.
  - 3. The customer clicks the payment button, various things happen in the background and eventually the merchant gets a confirmation saying,  
*“a payment was made by customer ID [customer-id] for item [item-id] in amount [value].”*

- Manual process OK for donations, few items.
- CTRL C+V HTML code for thousands of items?
- Payment services also provide  
programmatic interfaces  
for dynamically generated webpages.

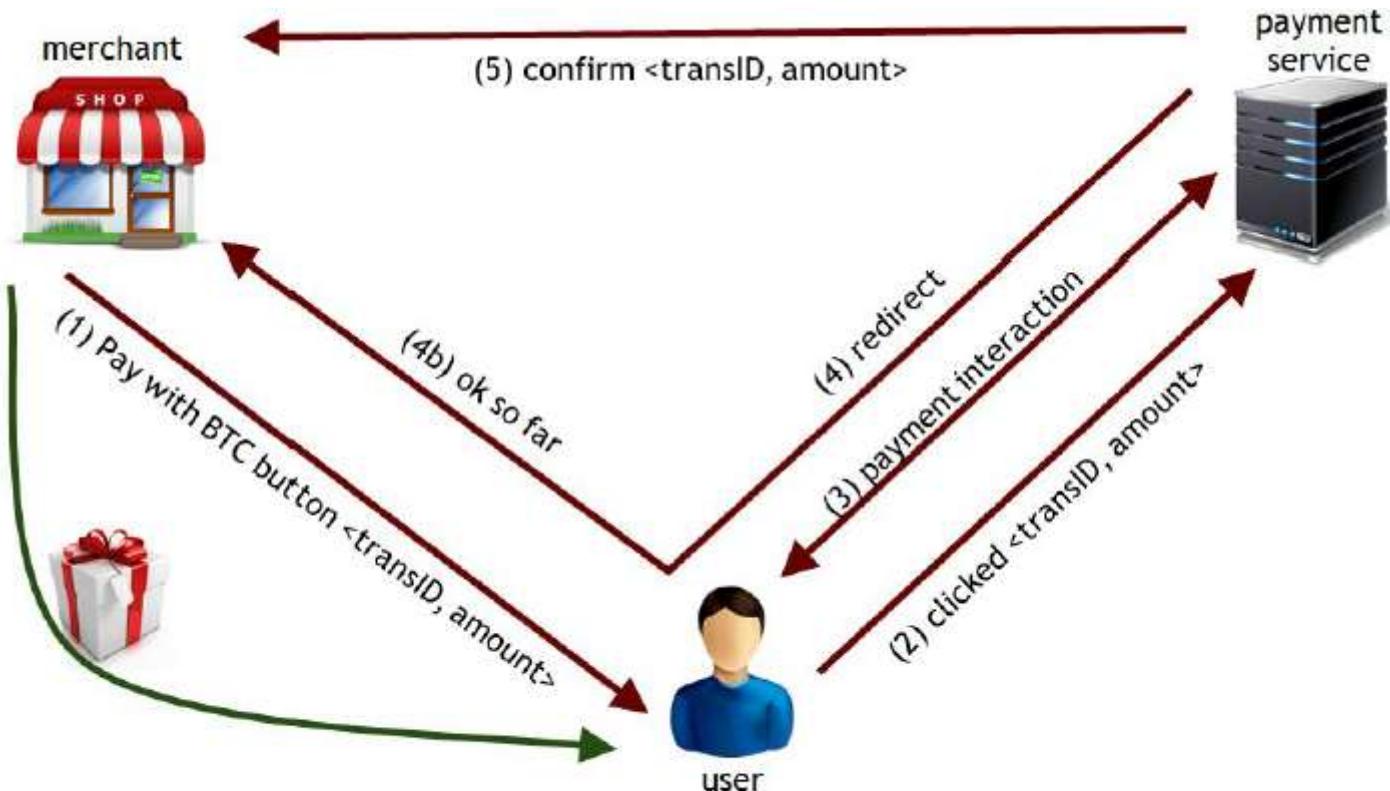
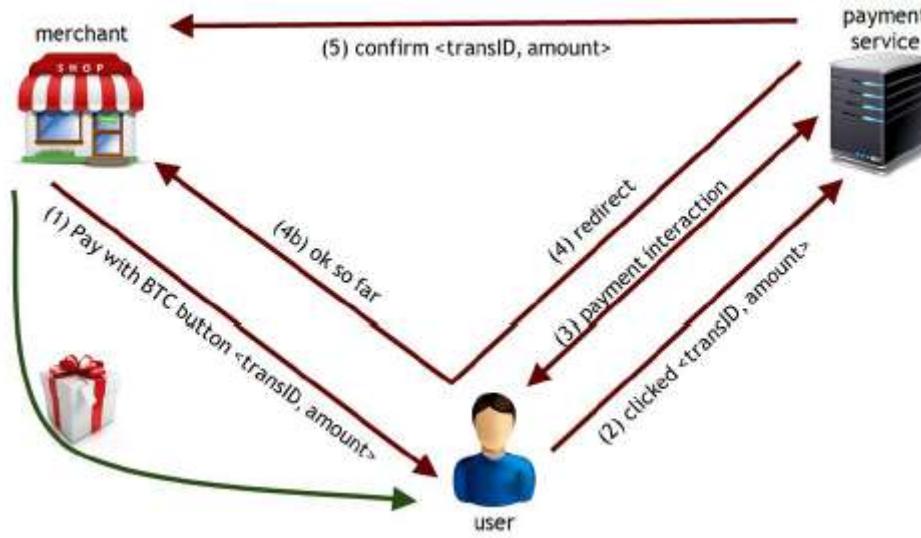
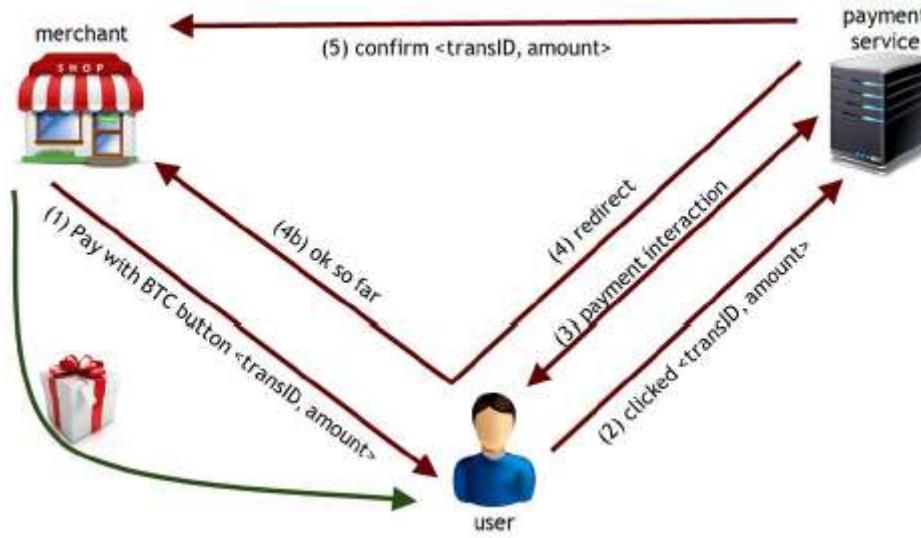


Figure 4.8: Payment process involving a user, merchant, and payment service.



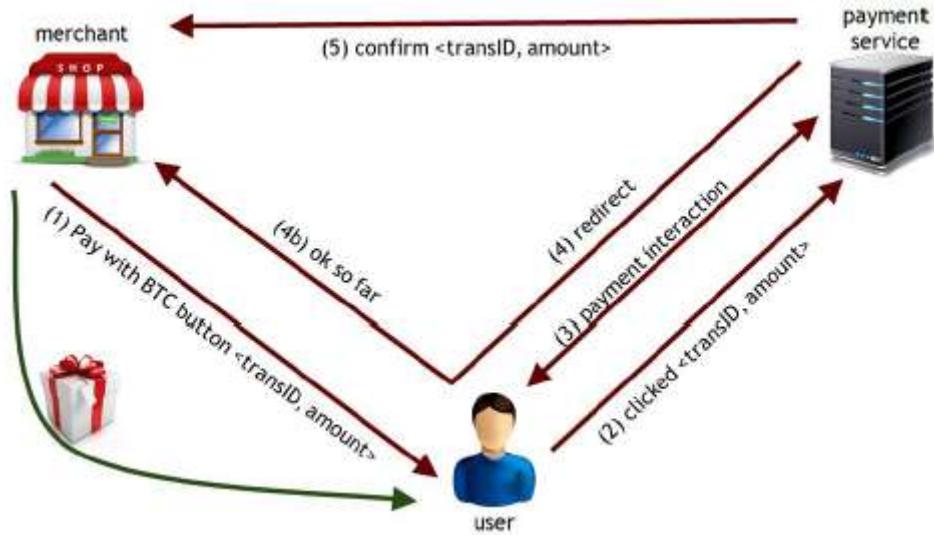
- Step 1

- The user picks out an item to buy on the merchant site,
- The merchant will deliver a webpage which will contain the Pay with Bitcoin button, which is the HTML snippet provided by the payment service.
- The page will also contain a transaction ID — which is an identifier that's meaningful to the merchant and allows them to locate a record in their own accounting system — along with an amount the merchant wants to be paid.



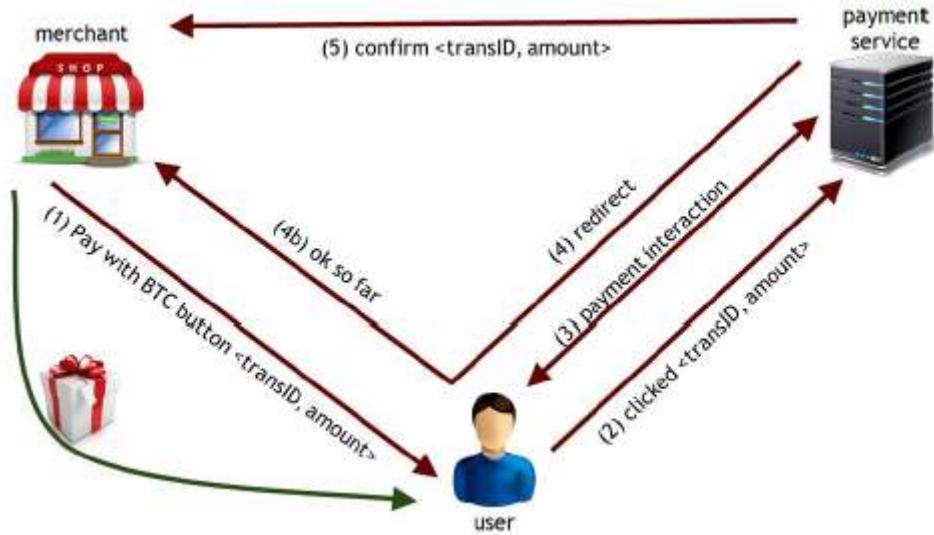
- Step 2

- If user wants to pay with bitcoins, clicks that button.
- That will trigger an HTTPS request to the payment service saying that the button was clicked, passing on the
  - identity of the merchant,
  - the merchant's transaction ID,
  - the amount.



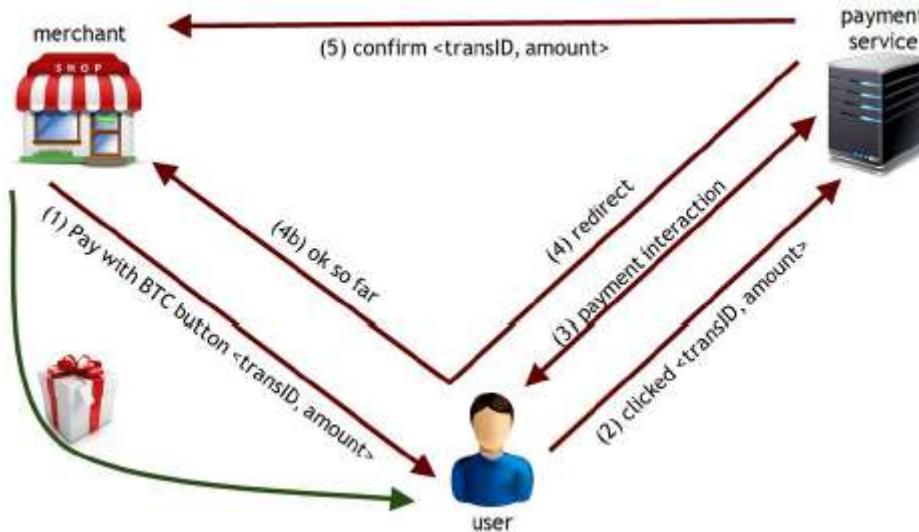
- Step 3

- Now the payment service knows that this customer wants to pay a certain amount of bitcoins,
- and so the payment service will pop up some kind of a box,
- or initiate some kind of an interaction with the user.
- This gives the user information about how to pay,
- the user will then initiate a bitcoin transfer to the payment service through their preferred wallet.



- Step 4

- Payment service will redirect the browser to the merchant, passing on the message from the payment service that it looks okay so far.
- This might mean, e.g. the payment service has observed the transaction broadcast to the peer-to-peer network, but the transaction hasn't received enough confirmations.
- This completes the payment as far as the user is concerned, with the merchant's shipment of goods pending a final confirmation from the payment service.



- Step 5
  - The payment service later directly sends a confirmation to the merchant containing the transaction ID and amount.
  - By doing this the payment service tells the merchant that the service owes the merchant money at the end of the day.
  - The merchant then ships the goods to the user.
- The payment service keeps a small % as a fee

- In summary:
  - the customer pays bitcoins
  - the merchant gets dollars, minus a small %.
  - everyone is happy ☺
- Recall that the merchant wants \$.
- The payment service handles everything else:
  - receiving bitcoins from customers
  - making deposits at the end of the day.

- Crucially, the payment service absorbs all of the risk.
  - Security risk, so it needs to have good security procedures to manage its bitcoins.
  - Exchange rate risk: Loss, gain.. Risk!
- Absorbing it is part of the payment service's business.
- Participate in the exchange market!

# Transaction Fees

- Any transaction on the blockchain might include a transaction fee
- Total value of coins IN - Total value of coins OUT
- IN > OUT
- As of 2016
- Why fee?
  - Miner: Who builds your transaction to a block
  - That miner's block is longer → Takes longer to propagate
  - Another block was probably found almost simultaneously
  - Cost to Network & Miners

- Nodes are not paid (currently)
- You can set your fee:
  - None
  - High → Relayed & Recorded quickly & reliably

# Default Transaction Fees

- No fee if all satisfied:
  - Transaction < 1KB
  - All outputs  $\geq 0.01$  BTC
  - Priority is large enough:  
$$(sum\ of\ input\ age * input\ value) / (transaction\ size)$$
Typical transaction: 400 Bytes → Free
- Otherwise: 1mBTC per 1KB

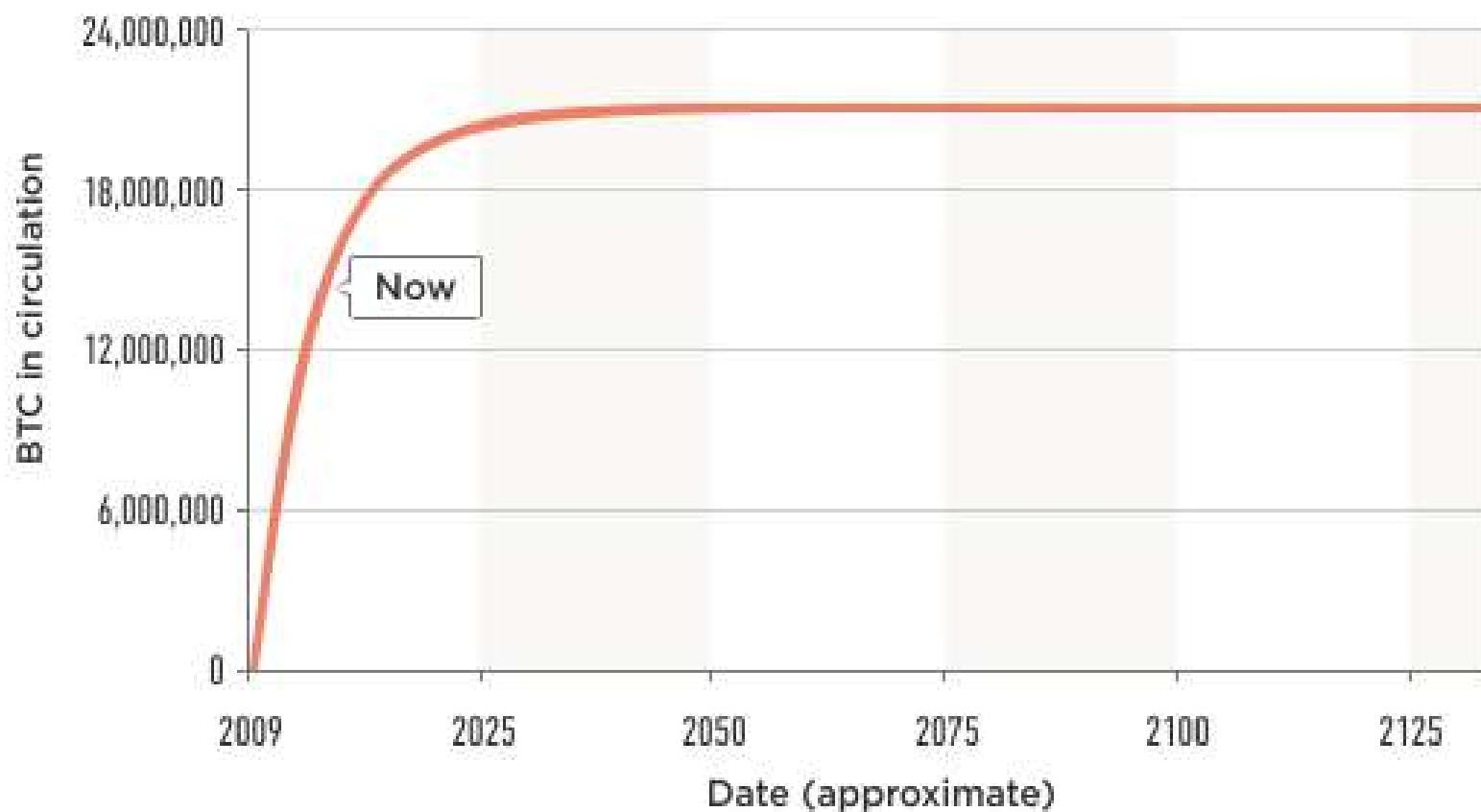
# Currency Exchange Markets

- Similar in many ways to market for \$ & Euro
- Live value, fluctuate over time
- March 2015
  - Bitfinex per 1 day: 70K BTC = 21 M\$
- Local sites: localbitcoins.com
  - Meet at a café/park
- Directly meet at places, pre-scheduled
  - Anonymous!

# Supply & Demand

- Supply
  - October 2015: 13.9 M BTC
  - Eventually: 21 M BTC
- Demand, why?
  - Mediating fiat currency transactions
    - Alice & Bob, use BTC just for transfer
    - Alice buys, sends; Bob receives, sells
    - During transaction, BTCS out of circulation
  - Investment
    - Buy & Hold: Out of circulation

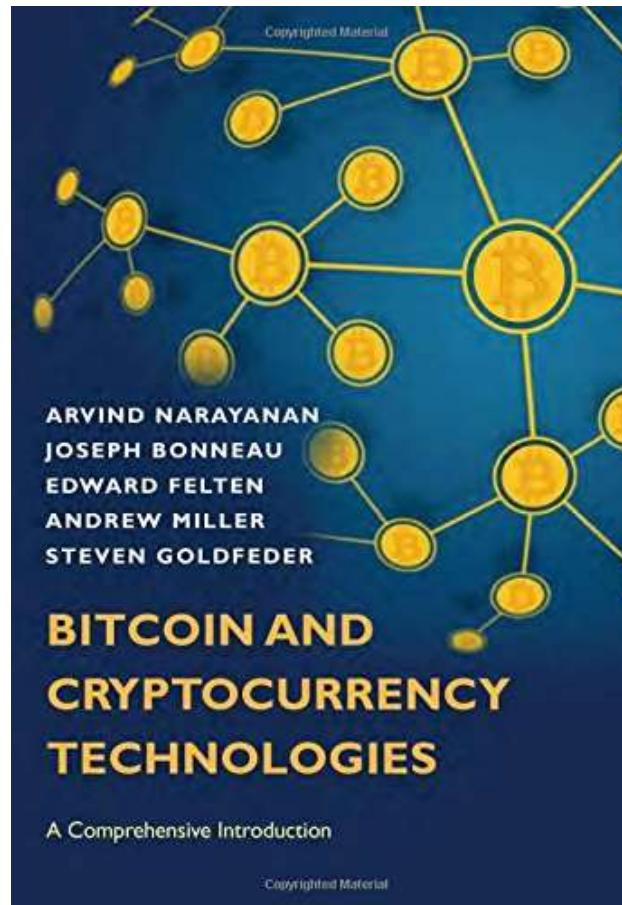
## CUMULATIVE BTC IN CIRCULATION



# Blockchain & Business Application

Lecture :  
Proof of Work  
Mining

# Chapter 5.1-5.4:Bitcoin Mining



# Miners?

- They
  - validate transactions
  - build & store blocks
  - reach consensus on blocks,
  - and.. rewarded \$ BTC \$
- But
  - Who are they?
  - How they get into; how they operate
  - Business model?
  - Impact on the environment?

Please watch this movie before this chapter



# Once connected to network, 6 tasks

1. Listen for transactions
2. Maintain blockchain & listen for new blocks
3. Assemble a candidate block
4. Find a nonce that makes your block valid
5. Hope your block is accepted
6. Profit \$\$\$\$ BTC \$\$\$\$ ☺

# Once connected to network, 6 tasks

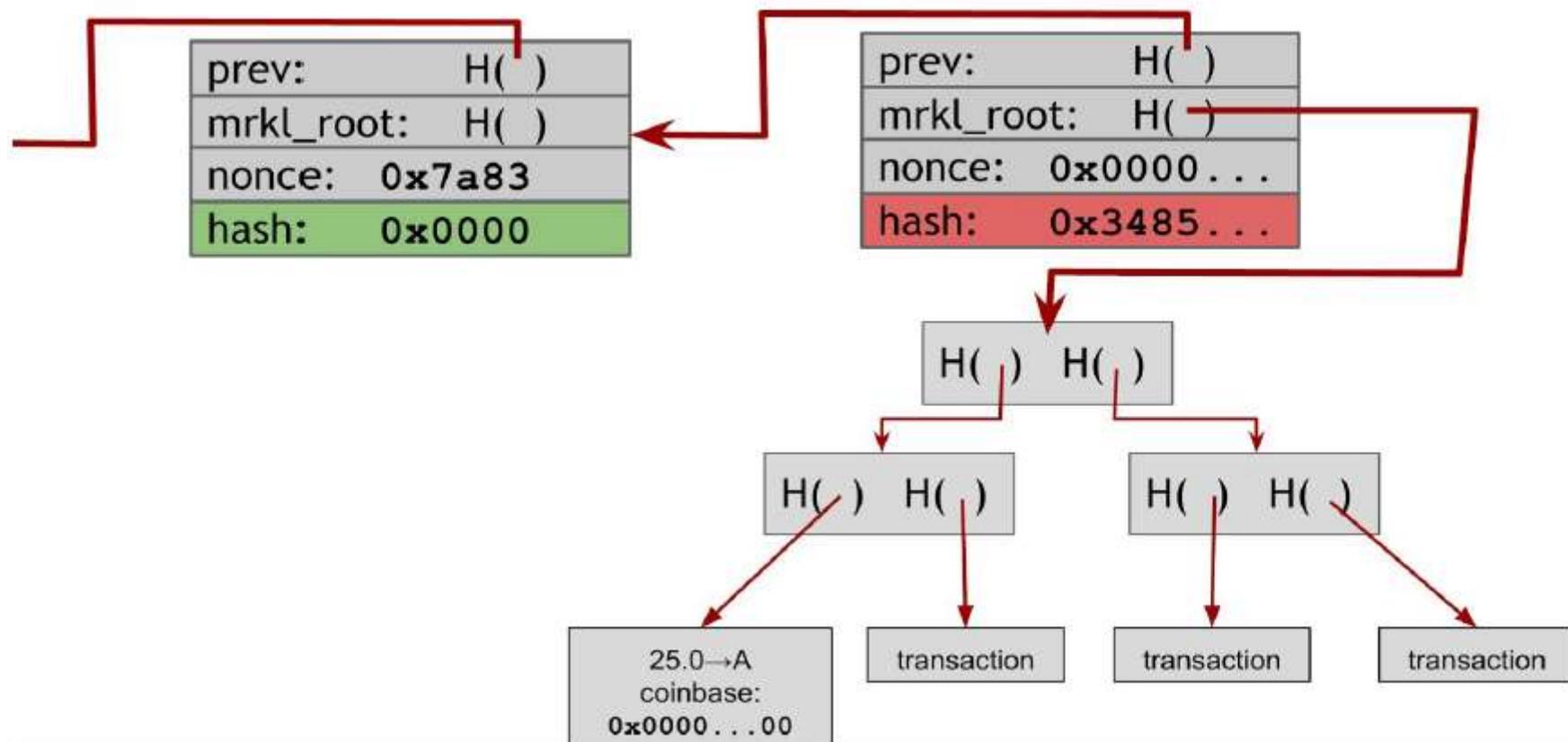
1. Listen for transactions
  - Listen on the network
  - Validate them
    - Check signatures
    - Outputs not spent before
2. Maintain blockchain & listen for new blocks
  - Ask other nodes all the historical blocks
  - Listen for new ones
  - Validate each block you receive

3. Assemble a candidate block
  - Validate each transaction
  - Group transactions to construct a block
4. Find a **nonce** that makes your block valid
  - Most work
  - Real difficulty for miners
5. Hope your block is accepted
6. Profit: Block reward:  
(2015) 25 BTC = 6.000\$ + tr. Fee  
(2019) 12,5 BTC x 9.300\$ per coin = 120K \$ !!!  
(2021) 12,5 BTC x 50.000\$ per coin = 625K \$ !!!

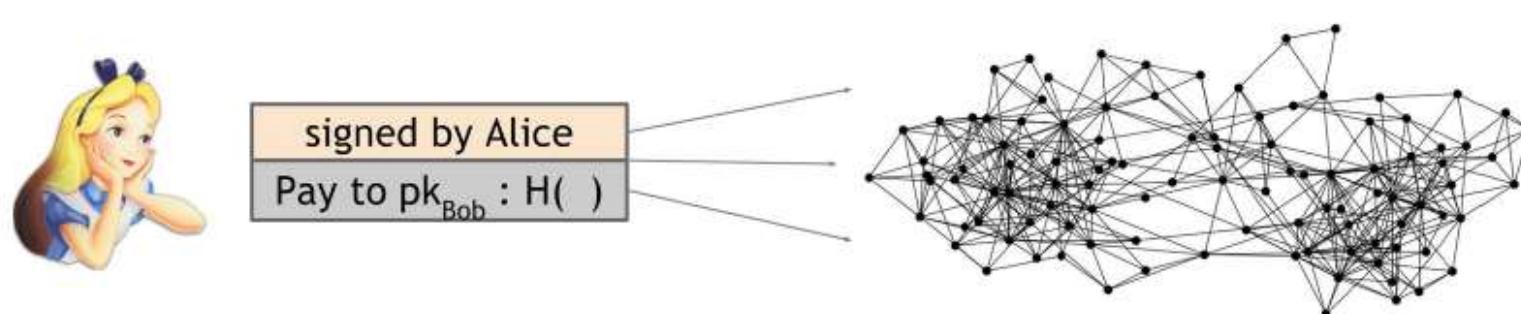
The amount of the reward halves every 210,000 blocks = 4 years The amount is expected to hit zero around 2140.

# Finding a valid nonce

- 32 bit nonce: block's hash to be under target



- Is everyone solving the same puzzle?
  - Faster miner always wins?
- No: Propagation, different transactions



**Figure 2.1 Broadcasting a transaction** In order to pay Bob, Alice broadcasts the transaction to the entire Bitcoin peer-to-peer network.

# Difficulty of finding a valid block

- March 2015, difficulty was:

0000000000000000172EC00000000000000000000000000000000000000  
00000000 (hexadecimal)

- Any valid hash: below this value
- 1 on  $2^{67}$ , huge number!
- Difficulty changes in every 2016 blocks
  - 1 block in 10 mins; 2016 blocks: 2 weeks!

`next_difficulty = (previous_difficulty * 2016 * 10 minutes) / (time to mine last 2016 blocks)`

# 2014: ~150K TH/s

- Network grows, hardware faster, but difficulty increases → Next block always in 10 mins

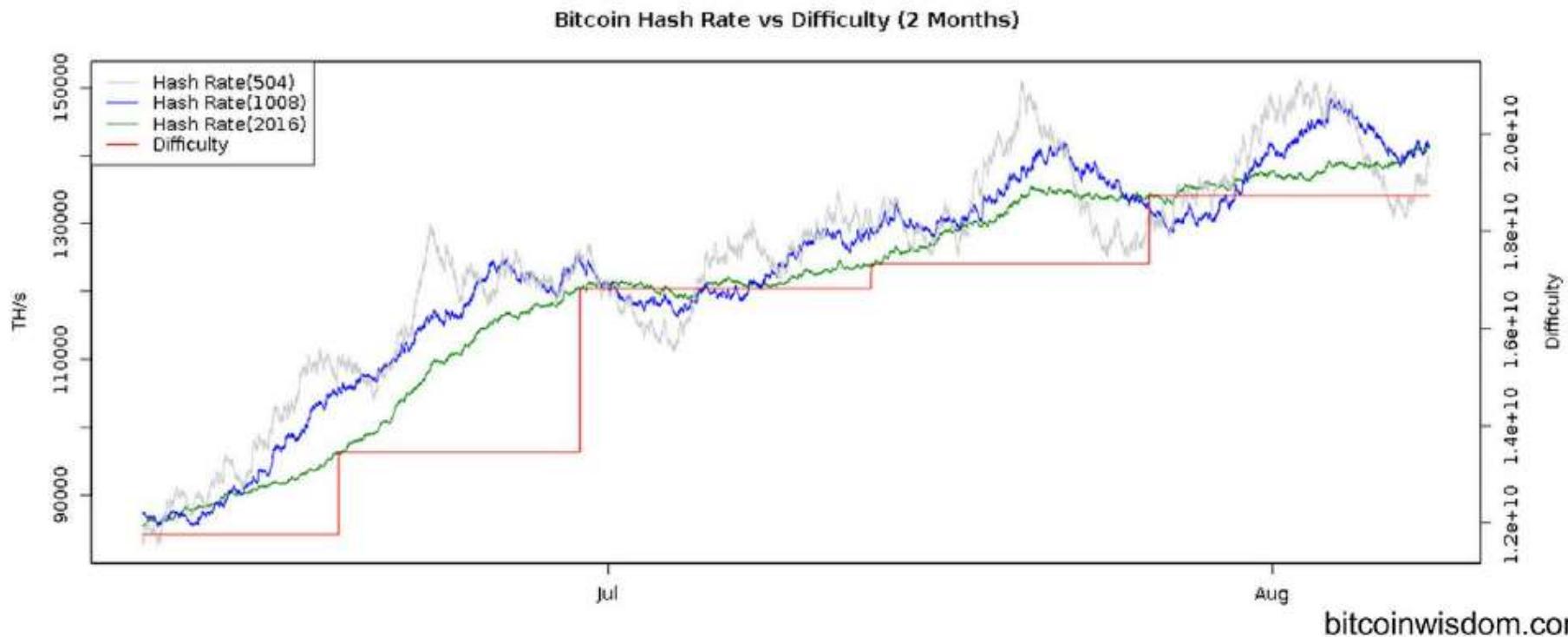
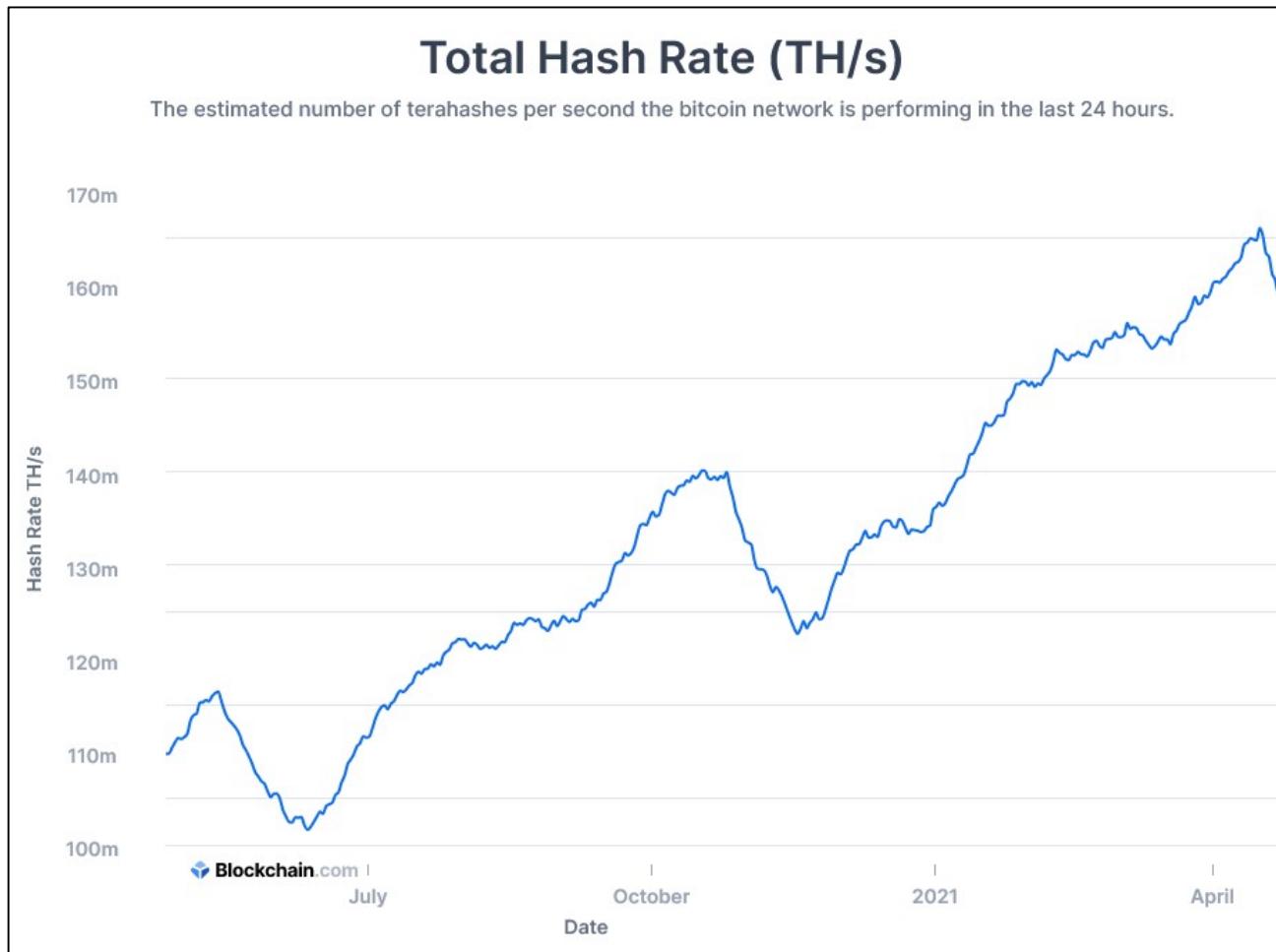


Figure 5.3: Mining difficulty over time (mid-2014). Note that the y-axis begins at 80,000 TH/s.

# 2021: ~150M TH/s



# Mining Hardware

- Difficulty arises from SHA-256
  - SHA-256 comes from US National Security Agency
  - Will eventually become weak
    - Processors becoming faster
    - Quantum computers!
  - Yet still strong enough
- Why difficult → Technical detail

- **Hardware for Mining:**

- CPU
- GPU
- FGPA
- ASIC

# CPU Mining

- First generation; general purpose computers
  - Search nonces linearly;
  - compute SHA-256;
  - Check if block valid

```
TARGET = (65535 << 208) / DIFFICULTY;
coinbase_nonce = 0;
while (1) {
    header = makeBlockHeader(transactions, coinbase_nonce);
    for (header_nonce = 0; header_nonce < (1 << 32); header_nonce++){
        if (SHA256(SHA256(makeBlock(header, header_nonce))) <
TARGET)
            break; //block found!
    }
    coinbase_nonce++;
}
```

Figure 5.6 : CPU mining pseudocode.

# CPU Mining, profitable?

- High-end PC 20 million hashes per sec (MH/s).
- Current difficulty level →  
Several hundred thousand years!!!
- Profitable?

# GPU Mining

- Performance graphics processing
- 2010, OpenCL introduced for other tasks
- Bitcoin Mining easily parallelized:
  - Compute multiple hashes
  - Different nonces
- For amateurs, easily
  - Available
  - Set up
  - Overclocked!

# GPU Mining

- Overclocking GPU
  - Run faster
  - Risk: Malfunction, overheat
  - For BTC Mining, profitable
  - Overclock for 50% faster → 1.5 times
  - Errors: 30% → 0.7 times correct
  - Product:  $1.5 \times 0.7 = 1.05$ ; means 5% gain!

# GPU Mining

- One PC with multiple GPU; Still early days



**Figure 5.7: A home-built rack of GPUs used for Bitcoin mining.** You can also see the fans that they used to build a primitive cooling system. Source: LeonardH, cryptocurrenciestalk.com.

# GPU Mining

- Disadvantages
  - Floating points in videos, not needed in SHA256
  - Not designed for cooling several of them
- Gamers turned to miners
- Electricity & initial cost
- One good GPU: 200 MH/s
- Hundreds → 300 years for a block (625K\$)
- Dead for Bitcoin, still alive for early altcoins

# FPGA Mining

- “Field Programmable Gate Array”



Figure 5.8: A home-built rack of FPGAs. Although you don't see the cooling setup pictured here, a rack like this would need a cooling system.

# FPGA Mining

- Field Programmable Gate Array
- Advantages
  - Customize for SHA-256
  - Easier systematic cool down
- One good FPGA: 1 GH/s (billion hashes / sec)
- Hundreds of FPGA → 50 years for a block ☹

# ASIC Mining

- “Application-Specific Integrated Circuits”
- Designed & Optimized specifically for mining
- Requires expertise & time
- Early times < 2015, race condition (shipping)
  
- You can start mining and earning
- Yet not profitable (price, electricity, cooling)

# Today: Professional Mining

- Obtain new ASIC not available to public

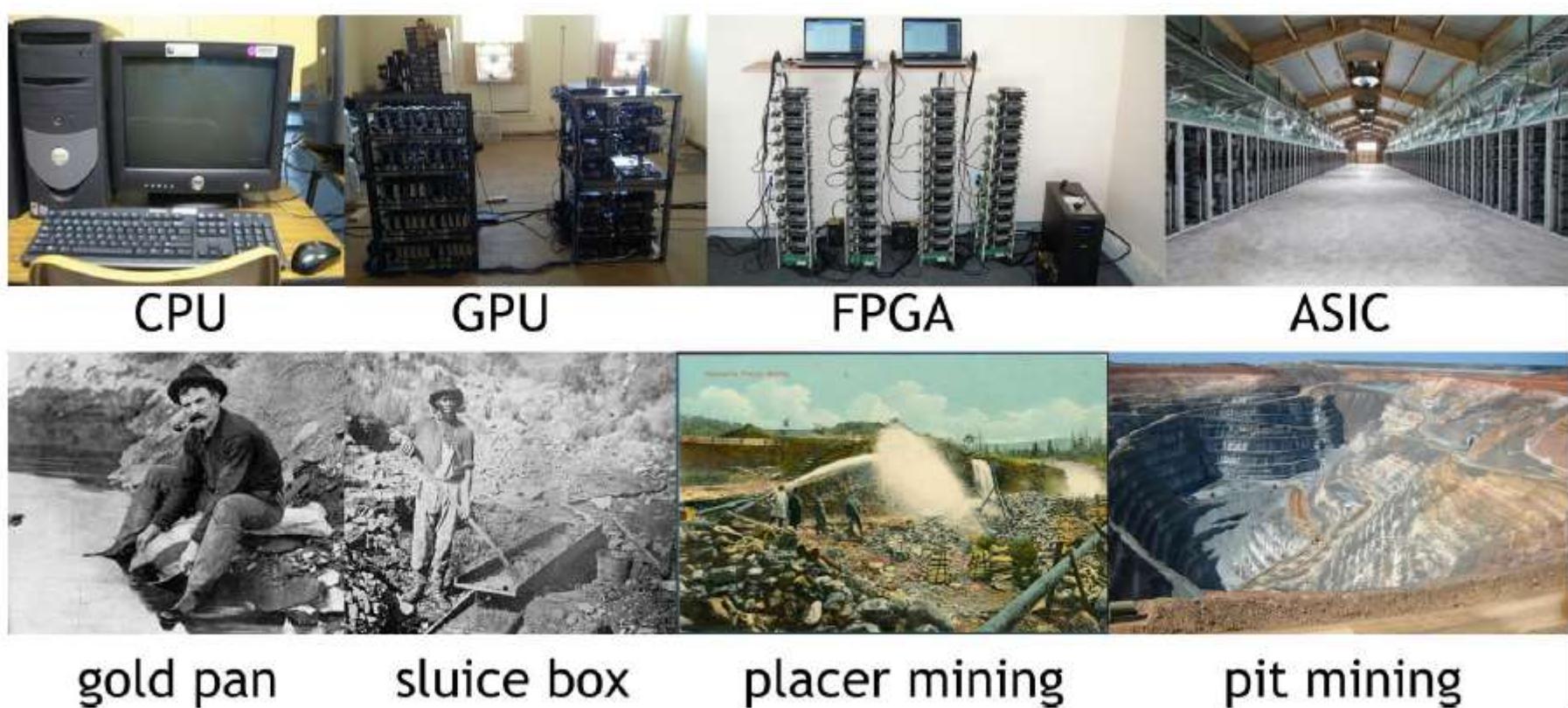


**Figure 5.9: BitFury mining center**, a professional mining center in the republic of Georgia.

# Setting up a mining center

- 3 biggest considerations:
  - Climate
  - Cost of electricity
  - network speed
- Georgia & Iceland are popular

# Similarities & Evolution



**Figure 5.10: Evolution of mining.** We can see a clear parallel between the evolution of Bitcoin mining and the evolution of gold mining. Both were initially friendly to individuals and over time became massive operations controlled by large companies.

# Future?

- Only ASICS & Professional miners?
- What about individuals
  - How to incorporate them?
- Violating decentralization?
- A method for only individuals?
- Altcoins?

# One of my current projects: *Quantum Fuels for Space Applications*



CA COST Action CA15220

Quantum Technologies in Space

THE JOURNAL OF  
**PHYSICAL CHEMISTRY C**

J. Phys. Chem. A B C Letters Pre-1997

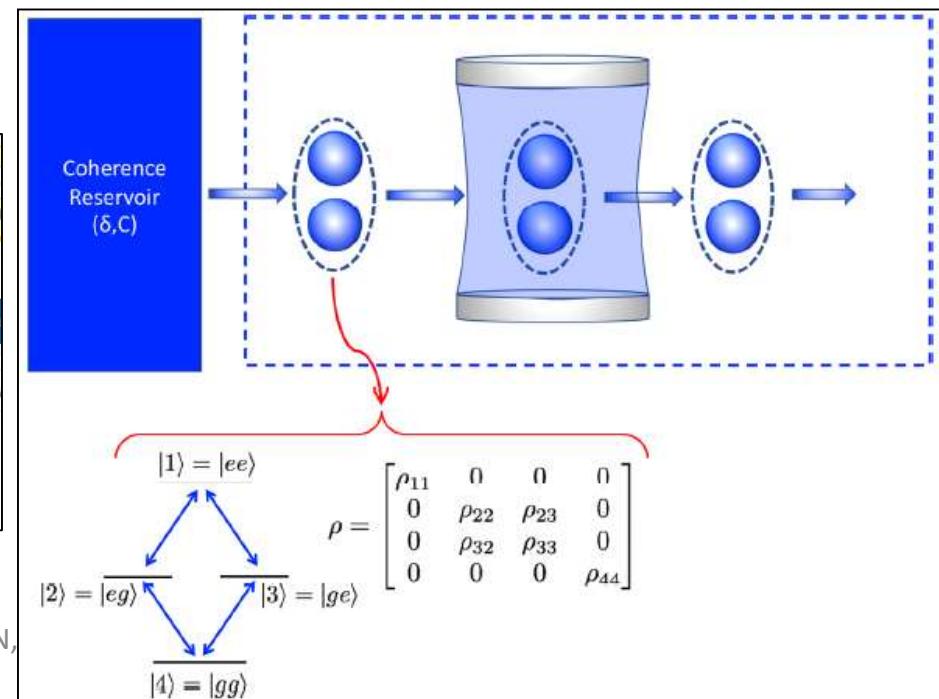
Browse the Journal Articles ASAP Current Issue Submission & Review Open Access About the Journal

Article

Temperature Control in Dissipative Cavities by Entangled Dimers

Ceren B. Dağ<sup>†</sup>, Wolfgang Niedenzu<sup>\*\$</sup>, Fatih Ozaydin<sup>†</sup>, Özgür E. Müstecaplıoğlu<sup>#</sup>, and Gershon Kurizki<sup>\$</sup>

Fatih OZAYDIN



# Energy consumption and ecology

- “Information is physical”
- Thermodynamics limit
- *Landauer’s principle* (1960s)
  - “*any non-reversible computation must use a minimum amount of energy*”
  - Not that optimized yet but a hard lower limit
  - Erasing information.. Logical operations?
  - SHA-256 is irreversible
  - *Reversible computation?*

# How does Bitcoin mining use energy?

- Embodied energy
  - Manufacturing: Raw materials → ASIC; Shipping?
  - May go down in future
- Electricity
  - Circuits: Going down (optimization, efficiency)
  - Landauer's principle, forever there!
- Cooling
  - Gets worse by scale

# How to estimate energy consumption

- Small dams: 10 MW
- Typical Dam: 1000 MW
- Typical Nuclear PP: 4000 MW
- Largest Nuclear PP: 7000 MW (Japan)
- Largest dam 10.000 MW (China)

# How to estimate energy consumption

- Top-down approach
  - Each block found: 25 BTC = 6,500 \$
  - 10 dollars / second
  - If all spent on electricity (upper bound)
  - 350 MW, in 2014
  - Block rate almost constant but difficulty increases

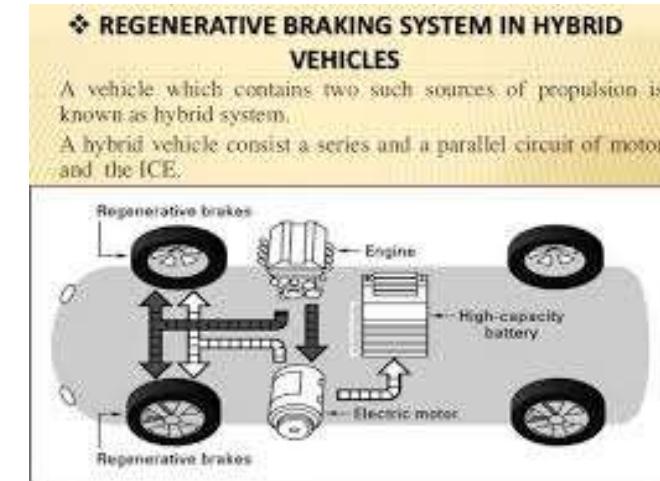
# How to estimate energy consumption

- Bottom-up approach
  - Most efficient ASIC: 3 GHashes / Watt
  - Total Hash Rate 350 Million Giga Hashes / sec
  - $(3 \text{ GH/W}) \times (350 \text{ M G H/s}) = 120 \text{ MW}$
  - Just computation;
  - Add Manufacturing & Cooling

# Bitcoin: Waste of Energy?

- Computing SHA-256, only for Bitcoin
- Unauthorized cost
- Energy Harvesting?
  - Bitcoin Prius ☺
- Opportunity Cost
  - Any method for “currency/payment” consumes
  - ATMs, Banks..

# *Repurposing energy?*



- Data Furnace:
  - Instead of buying a heater, buy a Mining Kit
  - Efficiency similar
  - Complexity similar

# Drawbacks of Repurposing

- Electric Heaters not efficient ( as gas heaters)
- During summer?
- Ownership not clear
  - Who takes coin rewards?
  - Digital Rights Management (DRM) Battles?

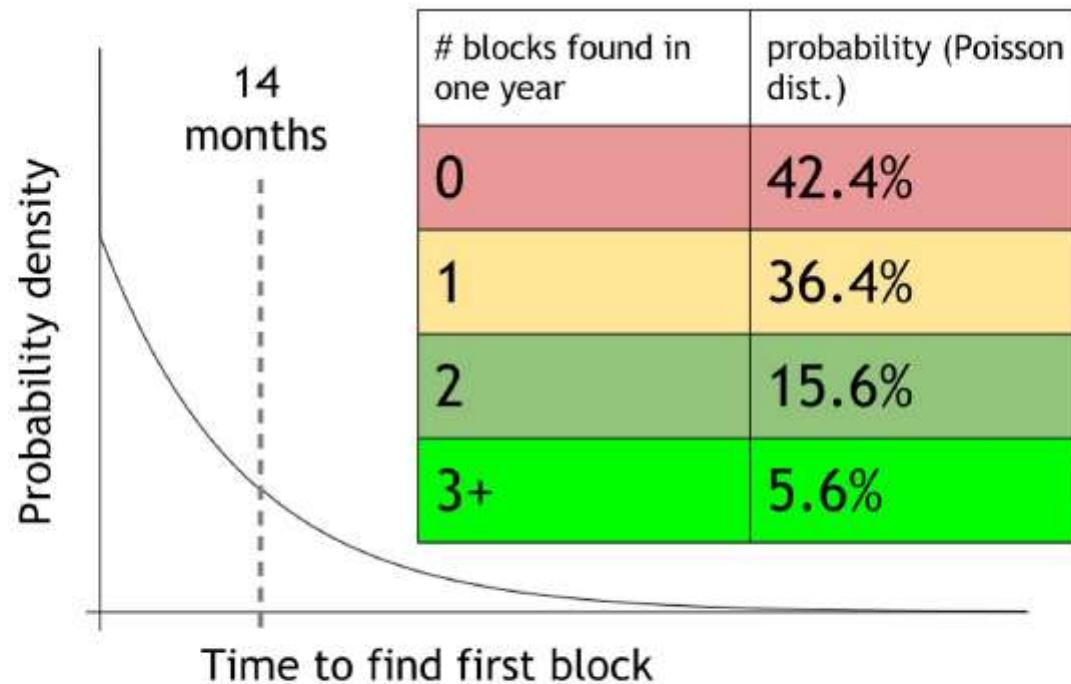
# Turning electricity in cash

- Providing free or low-cost electricity is open to new forms of abuse
- Governments subsidize industrial electricity
- Free electricity outlets everywhere!

# Mining Pools

- Consider the economics
  - Individual attempts
  - Initial Cost vs. Reward in every 14 months?
  - Add electricity & cooling costs;
  - Prefer a check every month?
  - Mining is random

- Blocks found in the first year:
  - Variance: High
  - Expected number: Low
  - Poisson Distribution
- If expectance: 1 Block in 14 months
- → 40% chance you won't find another same year
- Actual chance: 36% in the first year: Too bad



**Figure 5.11: Illustration of uncertainty in mining.** Assuming that the global hash rate is constant and the mean time to find a block is 14 months, the variance for a small miner is quite high.

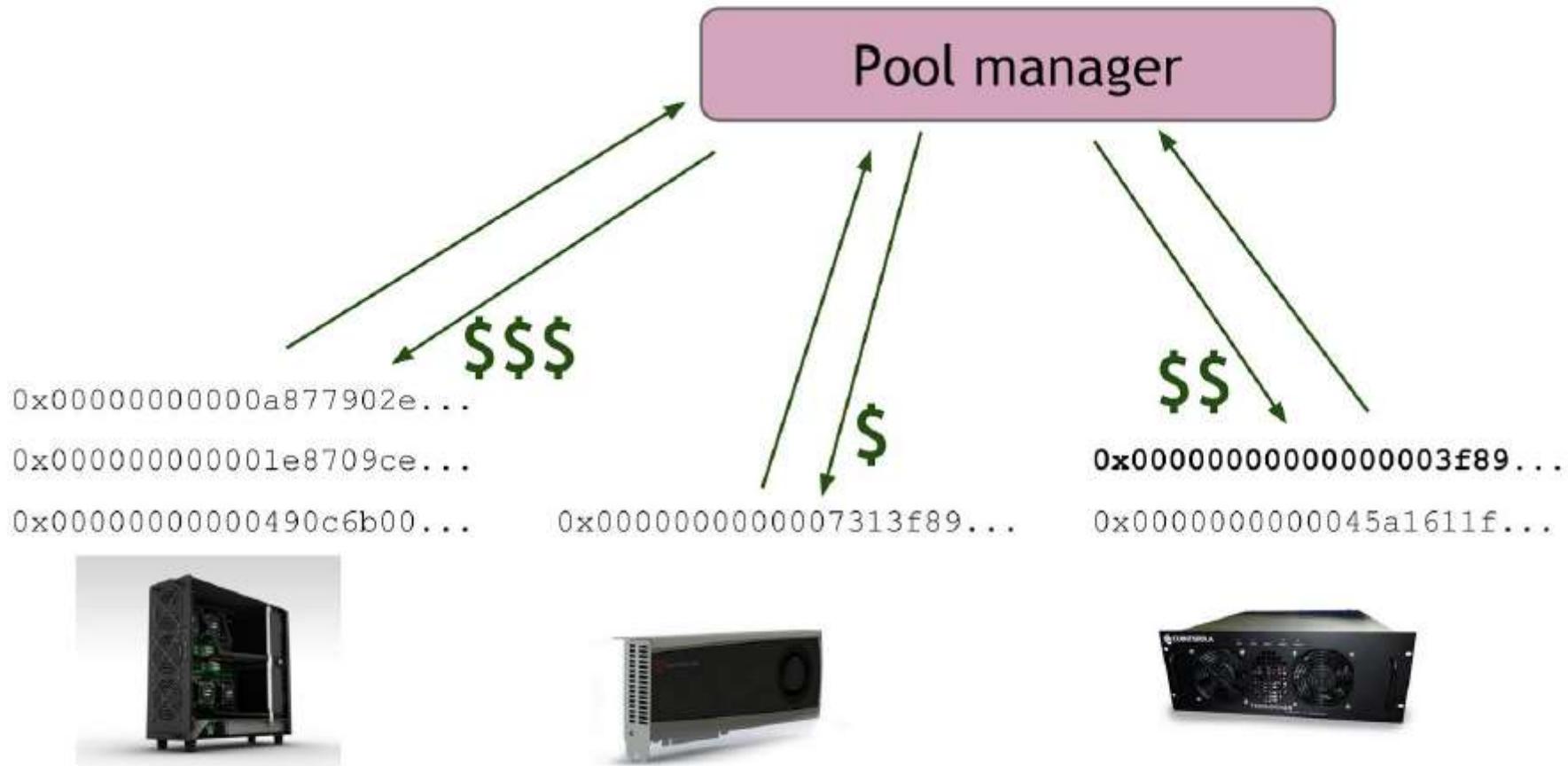
- Mutual Insurance companies to lower the risk
  - e.g. Farmers
- Mining Pools
  - Individuals get together in a pool
  - Pool manager receives all rewards to share among
  - How to share just? Who worked harder?

# Mining Shares

- Elegant solution: Miners can prove probabilistically how much they worked
- By **outputting** shares , or near-valid blocks.
  - e.g. target number beginning with 67 zeros.
  - Miners find hashes with less zeros (but they find)

```
4AA087F0A52ED2093FA816E53B9B6317F9B8C1227A61F9481AFED67301F2E3FB
D3E51477DCAB108750A5BC9093F6510759CC880BB171A5B77FB4A34ACA27DEDD
00000000008534FF68B98935D090DF5669E3403BD16F1CFD41CF17D6B474255
BB34ECA3DBB52EFF4B104EBBC0974841EF2F3A59EBBC4474A12F9F595EB81F4B
00000000002F891C1E232F687E41515637F7699EA0F462C2564233FE082BB0AF
0090488133779E7E98177AF1C765CF02D01AB4848DF555533B6C4CFCA201CBA1
460BEFA43B7083E502D36D9D08D64AFB99A100B3B80D4EA4F7B38E18174A0BFB
000000000000000078FB7E1F7E2E4854B8BC71412197EB1448911FA77BAE808A
652F374601D149AC47E01E7776138456181FA4F9D0EEDD8C4FDE3BEF6B1B7ECE
785526402143A291CFD60DA09CC80DD066BC723FD5FD20F9B50D614313529AF3
00000000041EE593434686000AF77F54CDE839A6CE30957B14EDEC10B15C9E5
9C20B06B01A0136F192BD48E0F372A4B9E6BA6ABC36F02FCED22FD9780026A8F
```

**Figure 5.12: Mining Shares.** Miners continually try to find blocks with a hash below the target.



**Figure 5.13: Mining rewards.** Three participants pictured here are all working on the same block. They are awarded commensurate with the amount of work done. Even though the miner on the right was the one to find the valid block, the miner on the left is paid more since this miner did more work. There is (typically) no bonus paid to the miner who actually finds the block.

# Pay per Share

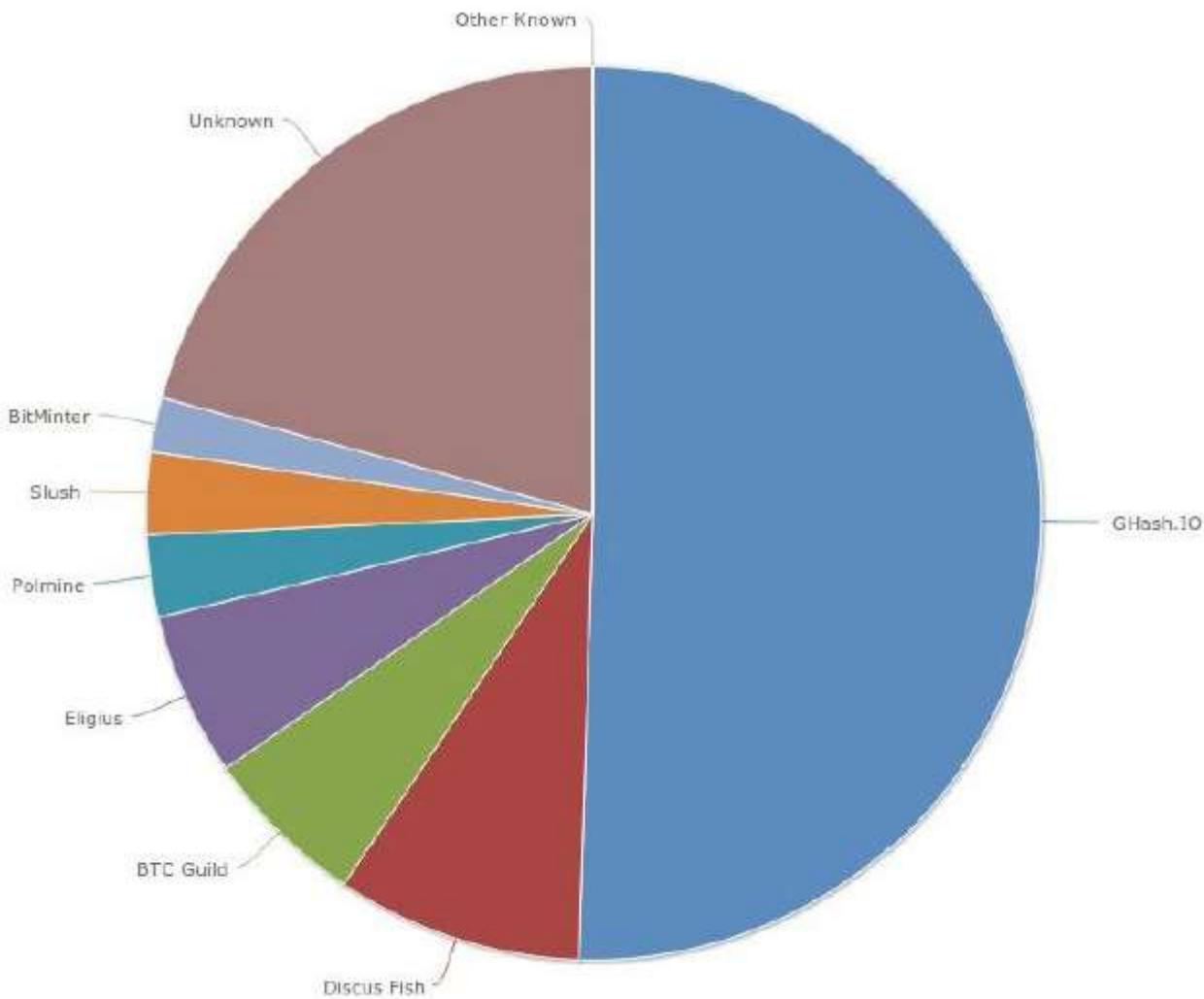
- Manager pays to each miner when they share
  - Does not wait for a valid block
  - Manager absorbs the risk → Takes more
  - May look the best
- Problems
  - Miners may not share valid blocks → Big loss to all
  - What if manager is malicious? Can attack the pool

# Proportional

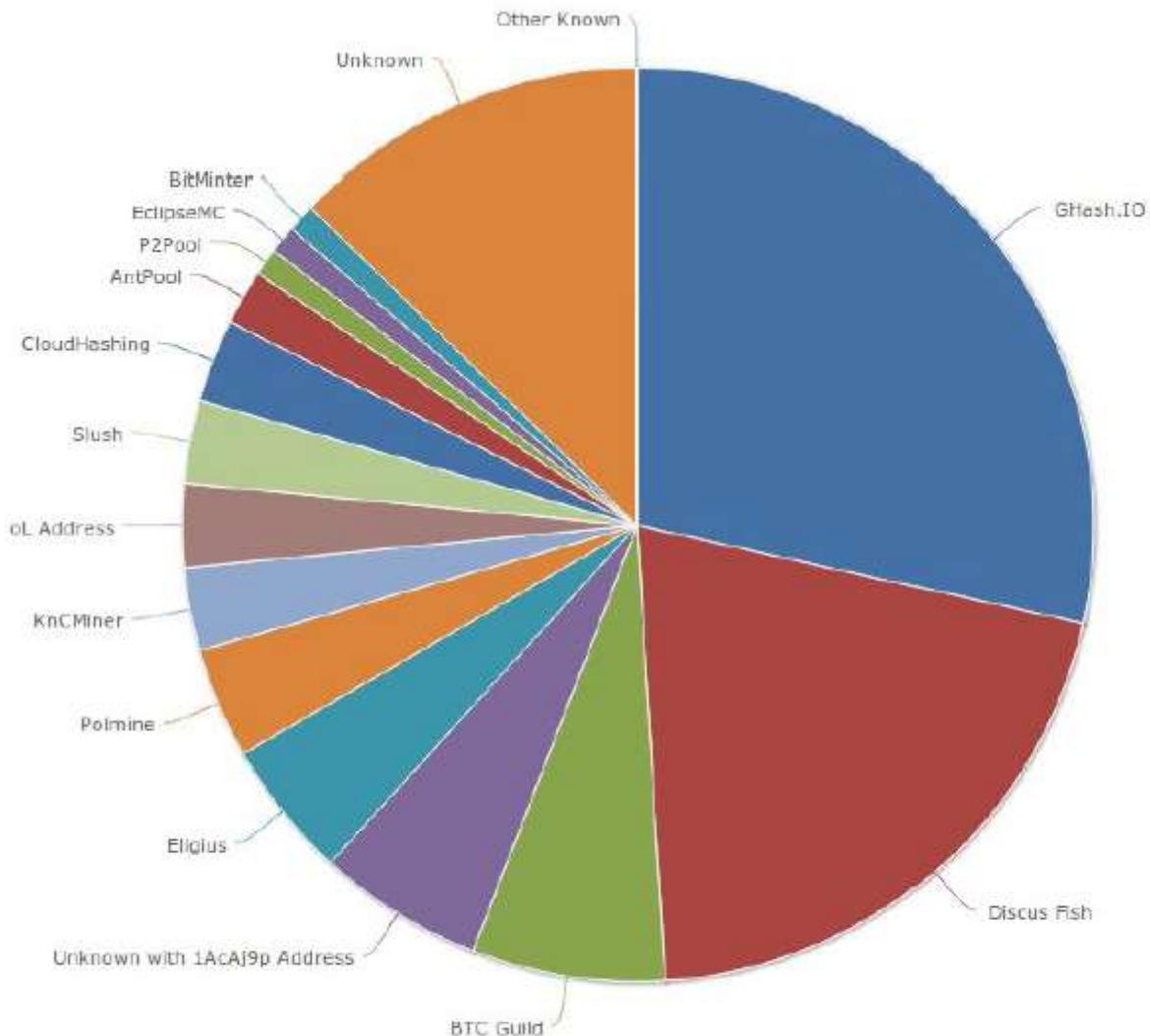
- Whenever a valid block is found
- Reward is shared proportionally wrt work
- To measure each miner's work
  - Advanced techniques
  - More work

# Pool Hopping

- Miners may wish to switch between pools
  - From purely proportional in the early cycles
  - To Pay-per-share in the later cycles
- An open problem
- 51% Concern !



**Figure 5.14 (a) Hash power by mining pool, via blockchain.info (June 2014)**



**Figure 5.14 (b) Hash power by mining pool, via blockchain.info (August 2014)**

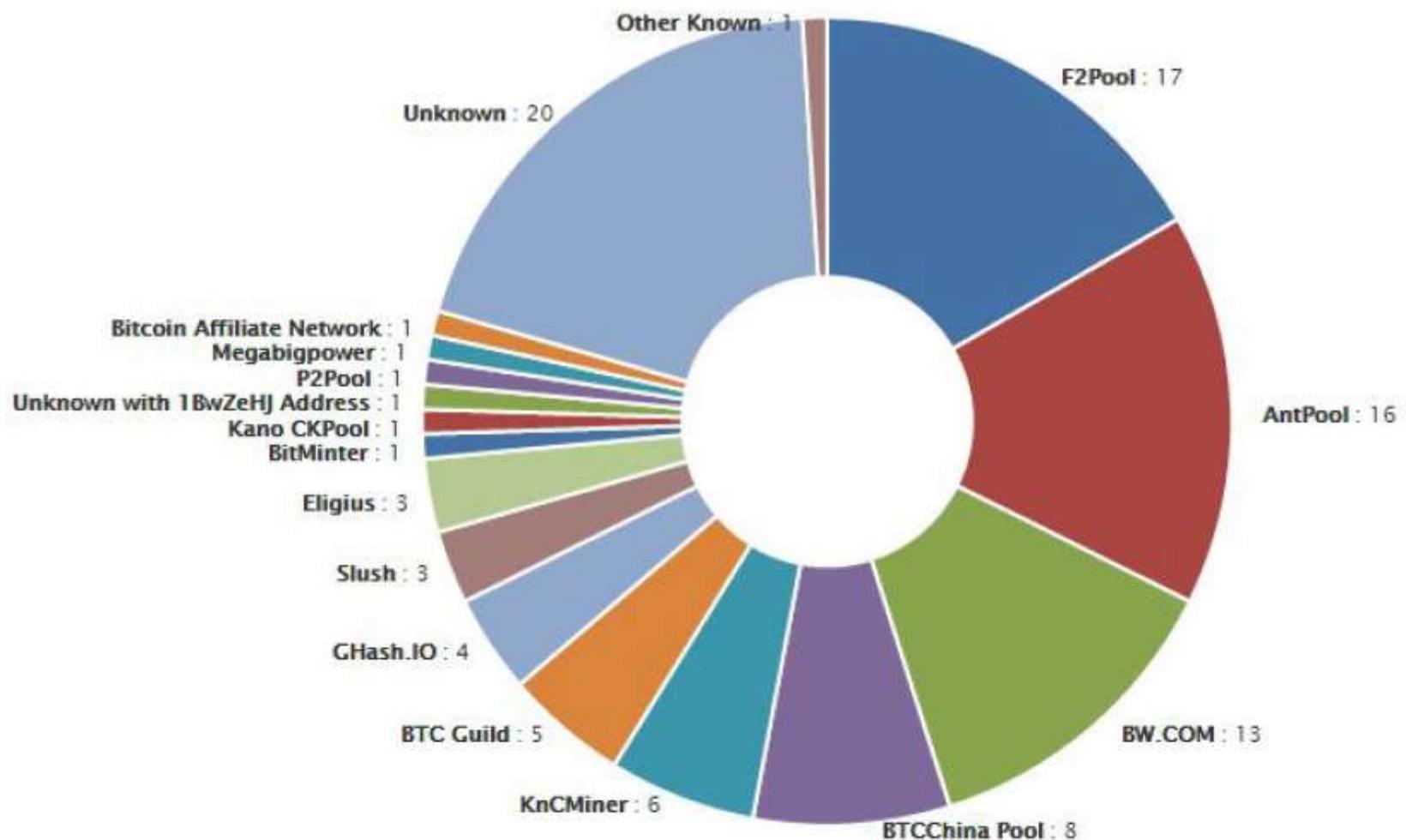


Figure 5.14 (c) Hash power by mining pool, via blockchain.info (April 2015)

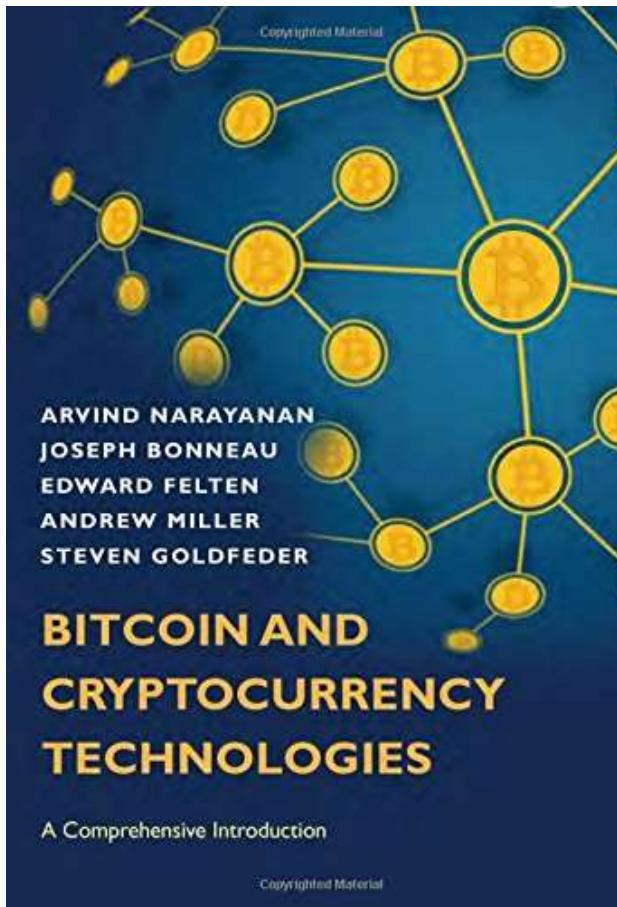
# Are Mining Pools a Good Thing?

- Advantages
  - Lower variance, higher profit predictability
  - Required for small miners
  - One central manager assembling the blocks
  - Easier system upgrade

# Are Mining Pools a Good Thing?

- Disadvantages
  - Centralization!
  - Leaving the pool?
    - In theory OK, in practice?
  - Lowers number of miners running full validation
- Redesigning the Pool Mechanism?

# Chapter 5.5: Bitcoin Incentives & Strategies, Mining Attacks



Blockchain / Smart Contracts

## Once hailed as unhackable, blockchains are now getting hacked

More and more security holes are appearing in cryptocurrency and smart contract platforms, and some are fundamental to the way they were built.

by Mike Orcutt

Feb 19, 2019

**Early last month, the security team at Coinbase noticed something strange** going on in Ethereum Classic, one of the cryptocurrencies people can buy and sell using Coinbase's popular exchange platform. Its blockchain, the history of all its transactions, was under attack.

Blockchain Mar 26

## Nearly all Bitcoin trades are fake, apparently



There have been suspicions for a while that the markets are overinflated. In fact, fears of market manipulation have held up regulatory approval for a number of proposed Bitcoin exchange-traded funds (ETFs), frustrating many enthusiasts who believe that the eventual approval of ETFs will spur broader adoption of the technology by investors.

# Incentives & Strategies

- So far:
  - Mechanisms,
  - Electricity,
  - Pools, etc
- What about strategies?
  - Which blocks to work on?

## Strategies for each miner

- Which transactions to include.
  - The default strategy is to include any transaction which includes a transaction fee higher than some minimum.
- Which block to mine on.
  - The default behavior: extend the longest known valid chain.

- Choosing between blocks at the same height
  - If two different blocks are mined and announced at around the same time, it results in a 1-block fork.
  - The default behavior is to build on top of the block that they heard about first.
- When to announce new blocks.
  - When they find a block, miners have to decide when to announce this to the Bitcoin network.
  - The default behavior is to announce it immediately, but they can choose to wait some time before announcing it.

# Forking Attack

Assumption: A non-default miner with mining power:  $\alpha$

- The miner sends some money to a victim, Bob, in payment for some good or service.
- Bob waits and sees that the transaction paying him has indeed been included in the block chain.
- Perhaps he follows the common heuristic and even waits for six confirmations to be sure.
- Convinced that he has been paid, Bob ships the good or performs the service.

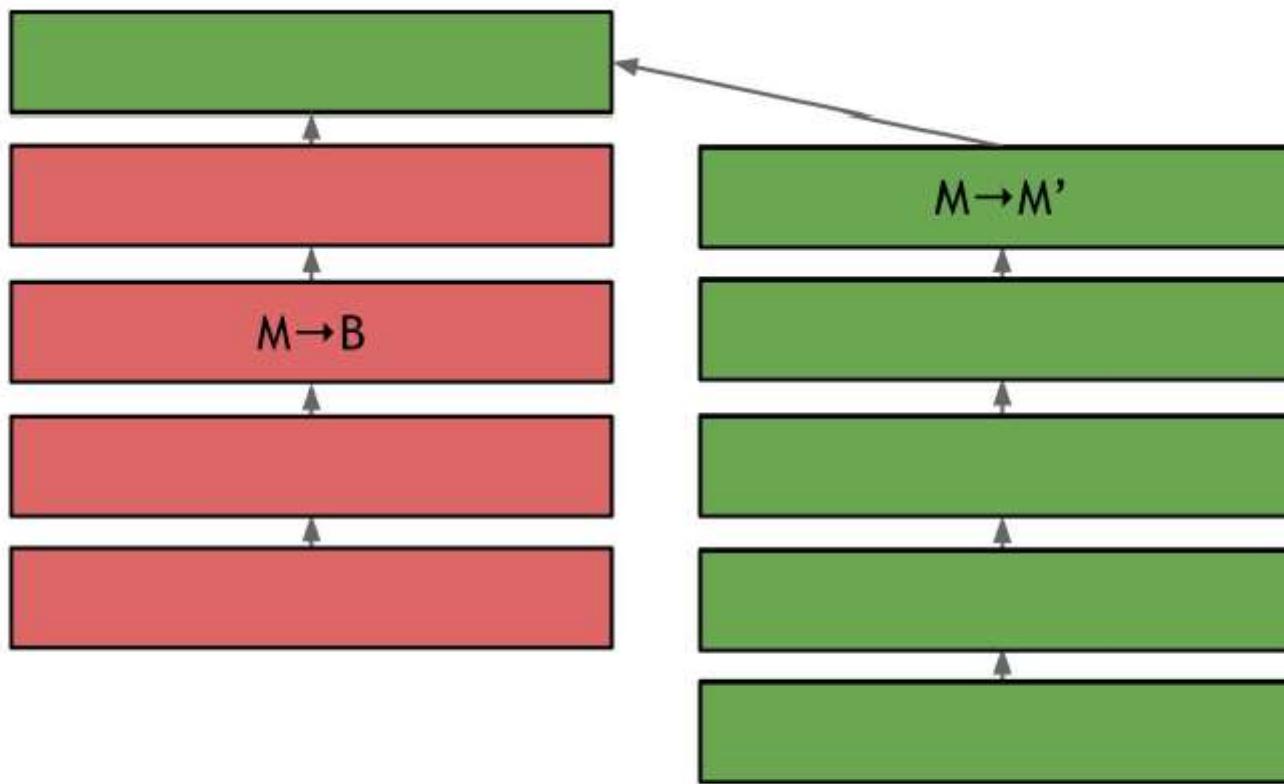
# Forking Attack

- The miner now goes ahead and begins working on an earlier block
  - Before block that contains the transaction to Bob.
- In this forked chain, the miner inserts an alternate transaction — or a double spend
  - which sends the coins paid to Bob on the main chain back to one of the miner's own addresses.

# Forking Attack

- For the attack to succeed, the forked chain must overtake the current longest chain.
- Once this occurs, the transaction paying Bob no longer exists on the consensus block chain.
- This will surely happen eventually if the attacking miner has a majority of the hash power:  $\alpha > 0.5$
- Since the miner's coins have already been spent (on the new consensus chain), the transaction paying Bob can no longer make its way onto the blockchain

# Forking Attack



**Figure 5.15 Forking attack.** A malicious miner sends a transaction to Bob and receives some good or service in exchange for it. The miner then forks the block chain to create a longer branch containing a conflicting transaction. The payment to Bob will be invalid in this new consensus chain.

# *Is 51% necessary?*

- Forking attack is certainly possible if  $\alpha > 0.5$ 
  - Even less power works: Network delays overheads
- Exceeding 50% percent makes it faster
- Historically called 51% attackers

- **Practical countermeasures:** It's not clear whether a forking attack would actually work:
  - The attack is detectable,
  - community would decide to block the attack
  - by refusing to accept the alternate chain even though it is longer

- **Attacks and the exchange rate.**
  - More importantly, it's likely that such an attack would completely crash the Bitcoin exchange rate
- Attacker gains small in short term
- But destroys the system
  - “*Goldfinger Attack*”

# Reading Assignments

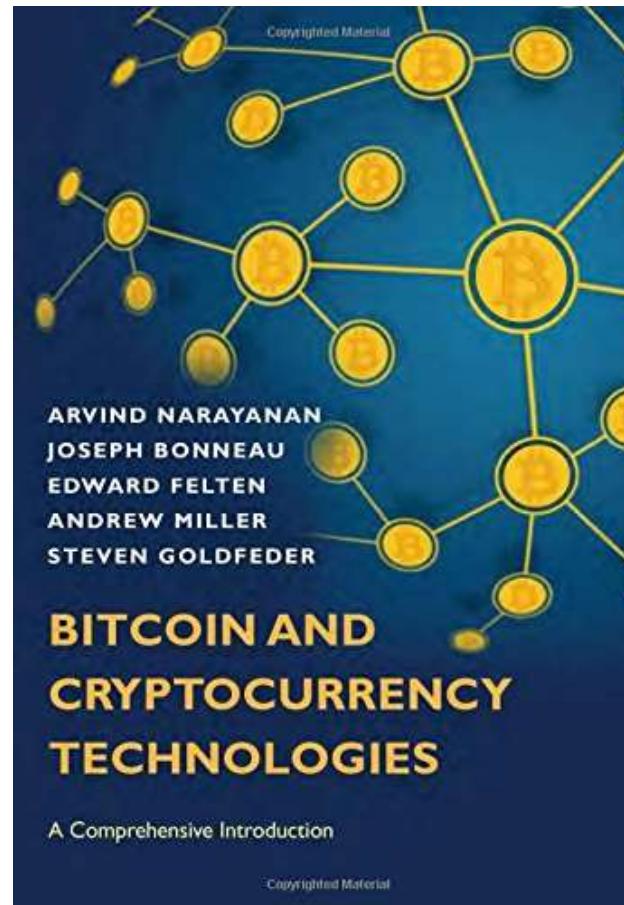
- Bitcoin and the age of bespoke Silicon
- Analysis of Large-Scale Bitcoin Mining Operations
- Research Perspectives and Challenges for Bitcoin and Cryptocurrencies
- Analysis of bitcoin pooled mining reward systems
- Majority is not enough: Bitcoin mining is vulnerable
- The economics of Bitcoin mining, or Bitcoin in the presence of adversaries
- The Miner's Dilemma

# Blockchain & Business Application

Lecture:

Bitcoin & Anonymity

# Chapter 6.1-6.4:Bitcoin & Anonymity



**“Bitcoin is a secure and anonymous digital currency”**

— WikiLeaks donations page

**“Bitcoin won't hide you from the NSA's prying eyes”**

— Wired UK

- Is Bitcoin anonymous?
- Do we want a cryptocurrency truly anonymous?
- Anonymity
  - Pros
  - Cons

# Basic & Hard Questions

- Is having an anonymous cryptocurrency beneficial for the stakeholders?
- Is it good for society?
- Is there a way to isolate the positive aspects of anonymity while doing away with the negative parts?

# Basic & Hard Questions

- Answers depend on one's ethical values.
  - We won't answer them
- We will examine arguments for & against anonymity
- We will study various technologies
  - some already present in Bitcoin
  - others that have been proposed to be added to it

- We'll also look at proposals for altcoins that have different anonymity properties from Bitcoin.
- These technologies raise new questions:
  - How well do they work?
  - How difficult would they be to adopt?
  - What are the tradeoffs to be made in adopting them?

# Anonymity Basics

## Definition

- Literally: “Without a name”
- In Bitcoin, 2 possibilities: Interacting,
  - without using your real name, or
  - without using any name at all
- They lead to very different conclusions!

- Bitcoin addresses are hashes of public keys. You don't need to use your real name in order to interact with the system
  - Bitcoin is anonymous as you do not use your real name
- But you do use your public key hash as your identity
  - it is not; the address that you use is a pseudo-identity.
- Language of Comp. Science (CS), this middle ground:  
***pseudonymity.***

- In CS, anonymity refers to
  - pseudonymity together with unlinkability .
- Unlinkability:
  - Defined wrt the capabilities of a specific adversary.
- Means
  - if a user interacts with the system repeatedly,
  - these different interactions should not be able to be tied to each other from the point of view of the adversary

- Bitcoin is pseudonymous, but
- Pseudonymity is not sufficient for privacy
- Blockchain is public:
  - If your address can be linked to your real ID
  - All your (ppf) transactions will be linked
- Linking is easy
  - Business apps usually ask your real ID
  - Credit card, shipping address.. At a Café?

- Side channels → Deanonymizing
  - Observing Bitcoin activities
  - Observing social media activities, linking them
- **Unlinkability:** It should be hard to link
  - Together different addresses of the same user
  - Together different transactions by the same user
  - The sender of a payment to its recipient (tricky!)

- **Anonymity set**
  - Sending bitcoins not directly but through a long route?
  - No: Each appears on the chain and can be traced
- Don't try full anonymity, try a limited set
- Given a particular adversary,
  - the anonymity set of your transaction:
  - the set of transactions which the adversary cannot distinguish from your transaction.
- Even if the adversary knows you made a transaction,
- they can only tell that it's one of the transactions in the set,
- but not which one it is. → maximize size of the set

- Calculating the set is tricky
- Taint Analysis (popular)
  - How related two addresses are
  - If transactions always  $S \rightarrow T$ , then a high taint score
- Not a good measure
  - Assumes adversary always uses same calculation
  - A more clever adversary; AI?

- Cryptocurrency anonymity problem worse than traditional banking
  - Banking: Revealing a single or few transactions
  - Bitcoin: Revealing all (ppf) transactions!
- Developing cryptocurrencies technically infeasible to reveal?

# Ethical Issues

- We usually hide our salaries
  - What if we receive in Bitcoins?
- Business secrets
  - R&D
  - subcontractors
  - new products
- Money laundering; illegal business?
  - Transactions may be anonymous
  - Not the trade between coins&cash

# Good, Bad?

- Can't we design a technology
  - Taking the good
  - Leaving the bad?
- No!
  - Depends on moral viewpoint, what is good/bad
  - Can't ask miners to include good/bad transactions
- Solution? Separate the technical anonymity from the legal principles

# Anonymization vs. decentralization

- Perfect anonymity requires controls
  - blind-signature protocols with central authority  
→ Destroying decentralization
- Perfect decentralization requires controls
  - Preventing double-spend, public traceability  
→ Destroying anonymity
- Anonymous decentralization?
  - Zerocoin, Zerocash (later)

# How to De-anonymize Bitcoin

- Bitcoin is only pseudonymous,
  - all of your transactions or addresses
  - could ***potentially*** be linked together

Bitcoin is a secure and anonymous digital currency. Bitcoins cannot be easily tracked back to you, and are safer and faster alternative to other donation methods. You can send BTC to the following address:

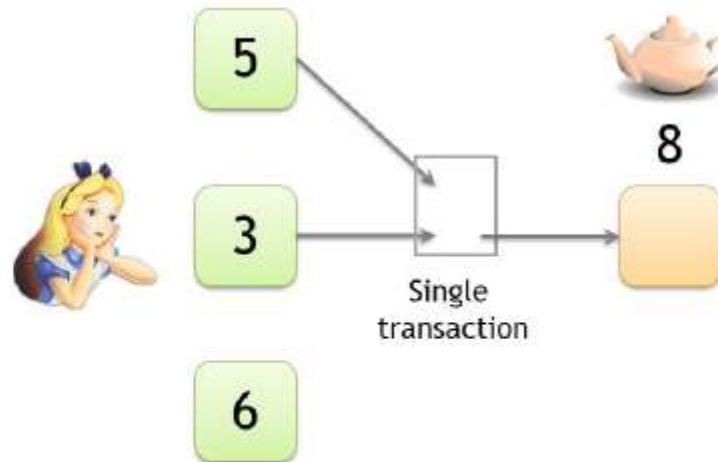
13DFamCvSxG8EG16VyXzdpfqxyooifswYx 

*Figure 6.1: Snippet from WikiLeaks donation page.* Notice the refresh icon next to the Bitcoin address. WikiLeaks follows the Bitcoin best practice of generating a new receiving address for every donation.

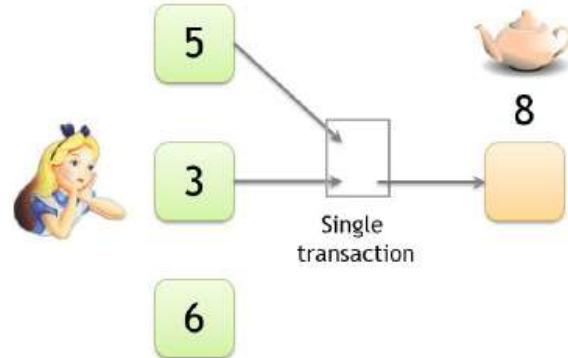
- At the first glance, transactions look unlinkable
- Each donation received (spent?) separately

# shared spending is evidence of joint control

- Suppose Alice
  - has 5, 3 and 6 coins in different addresses
  - Wants to buy a teapot of 8 coins



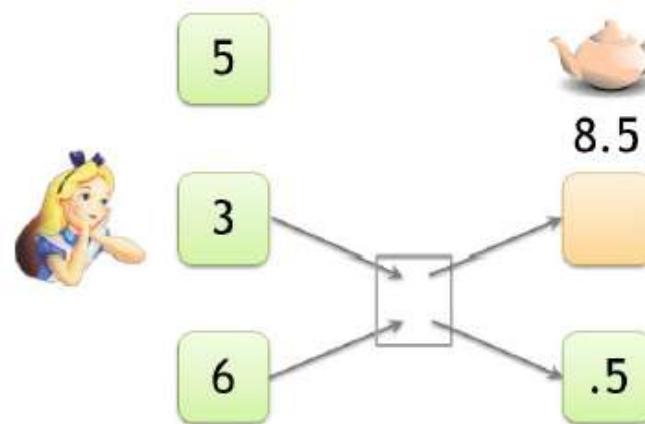
**Figure 6.2 :** To pay for the teapot, Alice has to create a single transaction having inputs that are at two different address. In doing so, Alice reveals that these two addresses are controlled by a single entity.



**Figure 6.2 :** To pay for the teapot, Alice has to create a single transaction having inputs that are at two different address. In doing so, Alice reveals that these two addresses are controlled by a single entity.

- Adversary can reveal that these two addresses are control by same person
- Adversary can continue observing transactions to link more
- Wallet software play a vital role in anonymity

- Suppose the pot was 8.5 coins



*Figure 6.3: Change address.* To pay for the teapot, Alice has to create a transaction with one output that goes to the merchant and another output that sends change back to herself.

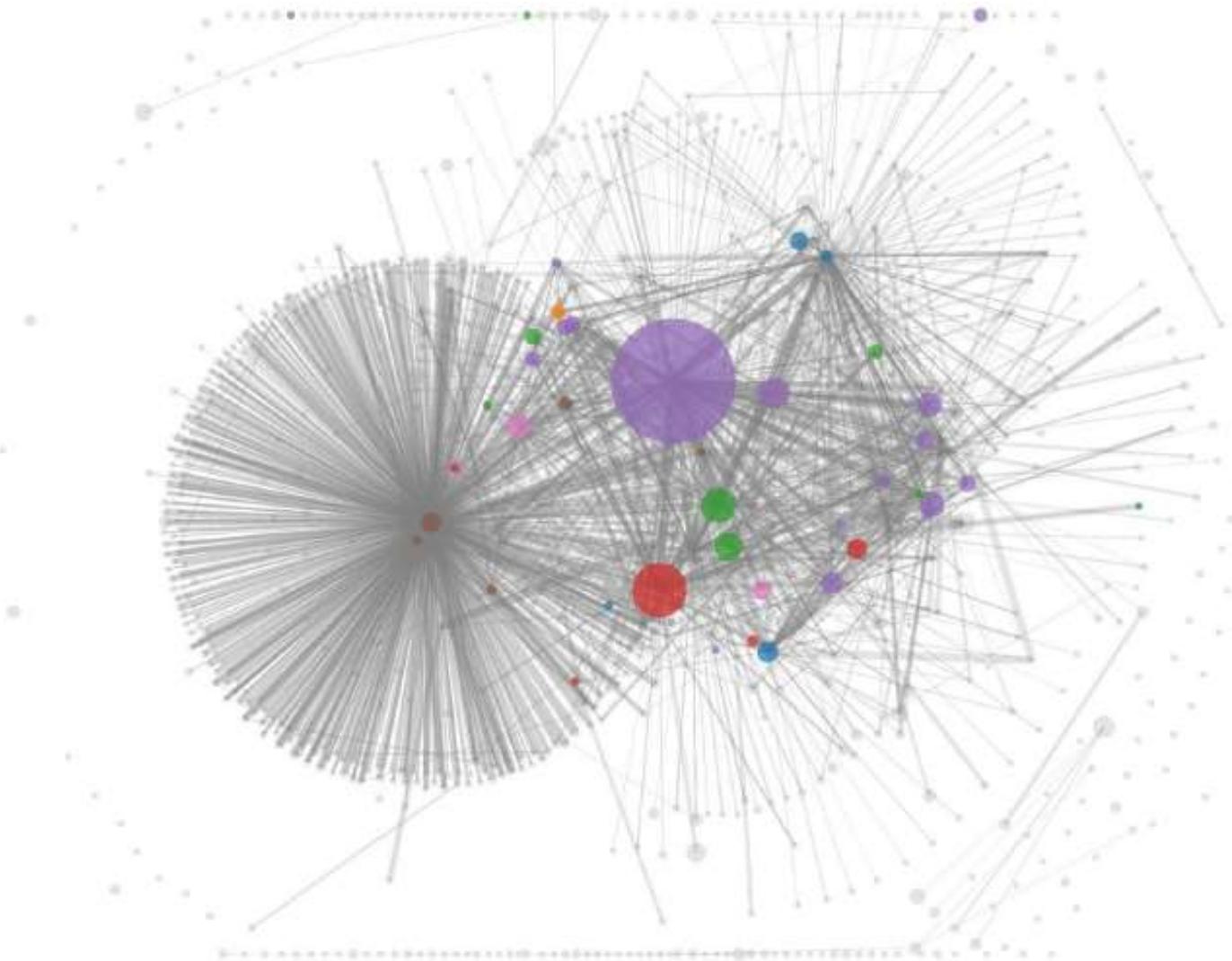
- One output: store's payment address
- The other: “change” address owned by herself
  - Heuristic!

# Idioms of Use

- In 2013, researchers found in most wallets,
  - Change addresses are brand new
  - No change output addresses existing

→ Providing a powerful heuristic on addresses

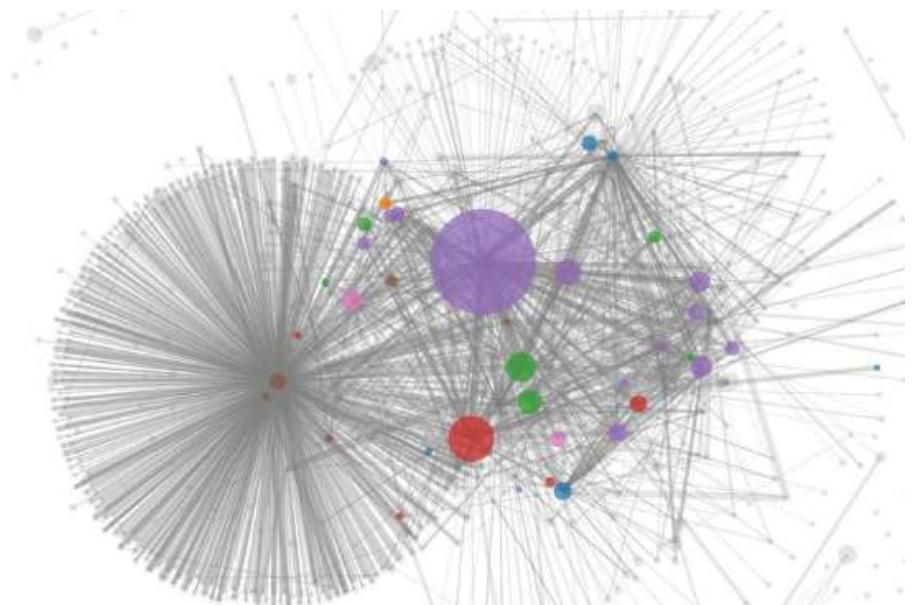
→ Transaction Graph Analysis (TGA)



**Figure 6.4: Clustering of addresses.** In the 2013 paper *A Fistful of Bitcoins: Characterizing Payments Among Men with No Names*, researchers combined the shared-spending heuristic and the fresh-change-address heuristic to cluster Bitcoin addresses. The sizes of these circles represent the quantity of money flowing into those clusters, and each edge represents a transaction.

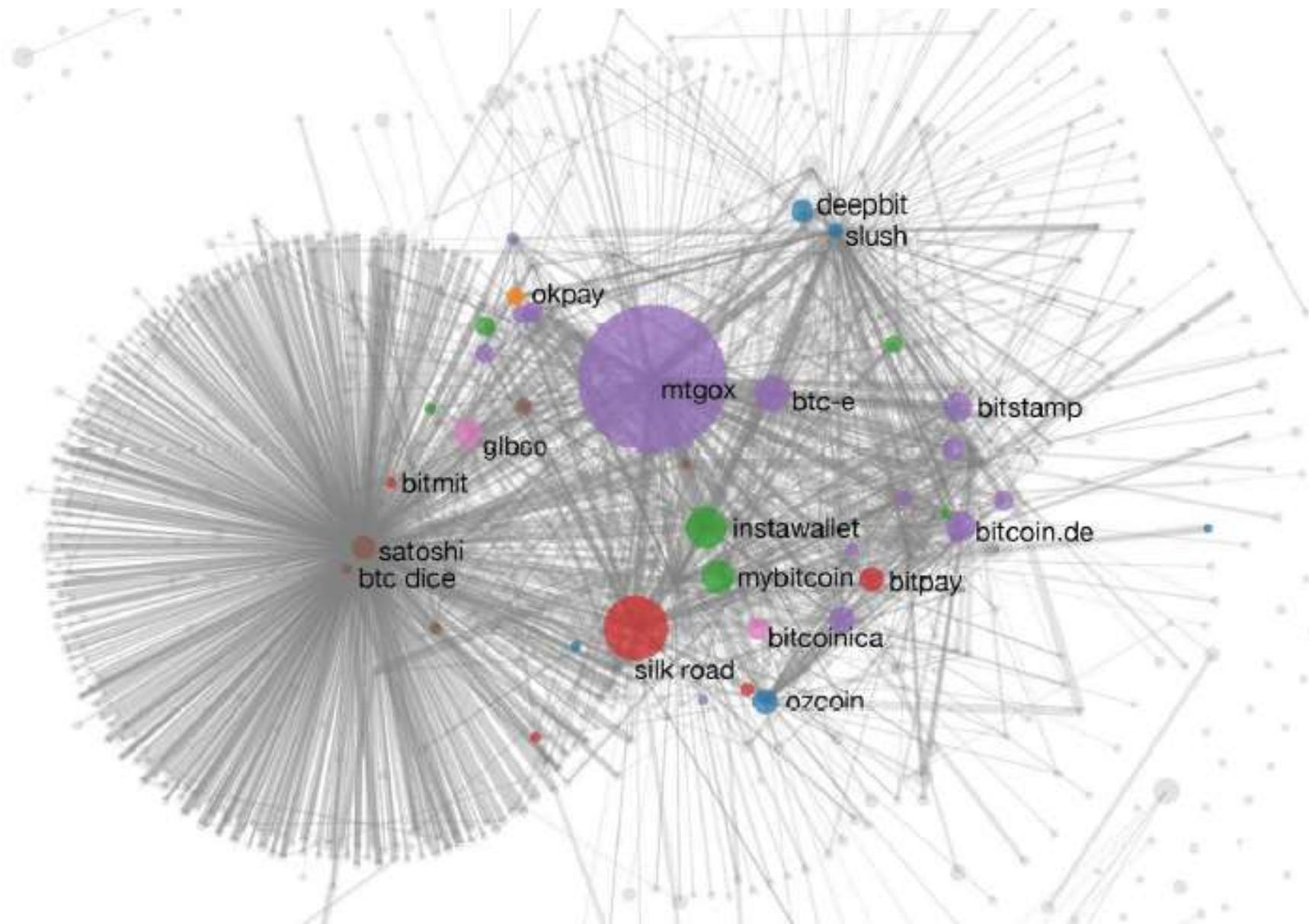
# Attaching Real World IDs

- Mt. Gox was the largest exchange
  - Purple?
- Satoshi Dice (gambling)
  - Brown small?
  - This method works only for a few dominant



# Tagging by transacting?

- Go to their websites, learn the addresses?
- Does not work:
  - Each is a new address (remember wikileaks)
  - That address will never be added to the chain
- Make a real transaction: Send them coins ☺



# Identifying individuals

- Can we do the same thing for individuals
- Can we link addresses to real life IDs?

## **Directly transacting:**

Anyone who transacts with an individual

- an online or offline merchant,
- an exchange,
- a friend who splits a dinner bill using Bitcoin
- knows at least one address belonging to them

## **Via service providers:**

- Almost everyone interacts with exchanges
  - They have to ask the Real IDs
- Law enforcement can reveal Real IDs

## **Carelessness:**

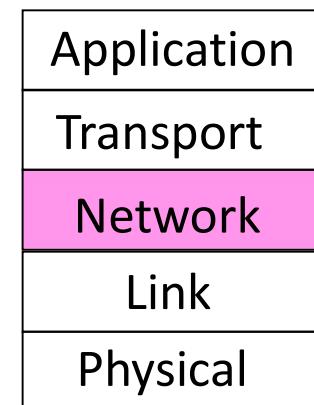
- People post their public addresses on forums

Things get worse over time

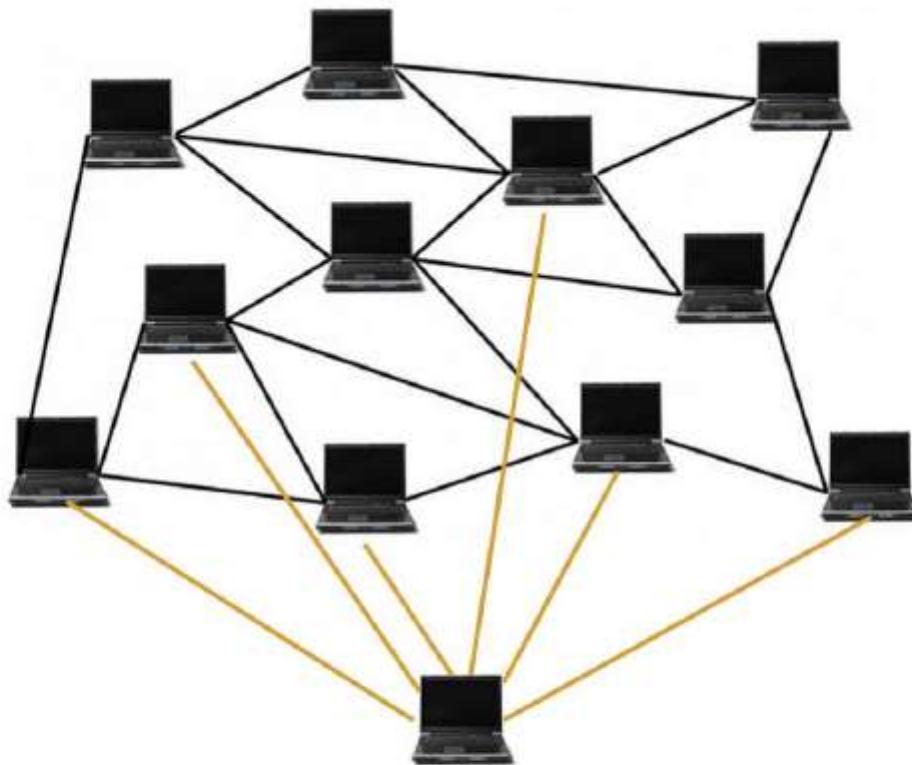
Transaction graph analysis improves

# Network-layer deanonymization

- Alternative to transaction graph analysis
- Each transaction posted/announced
- Not necessarily added to the chain
- Network terminology
  - Blockchain: Application layer
  - P2P Network: Network layer



- When a node creates a transaction, it connects to many nodes at once and broadcasts the transaction.
- If sufficiently many nodes on the network collude with each other (or are run by the same adversary),
- They could figure out the first node to broadcast any transaction.
- Presumably, that would be a node that's run by the user who created the transaction → IP Address!



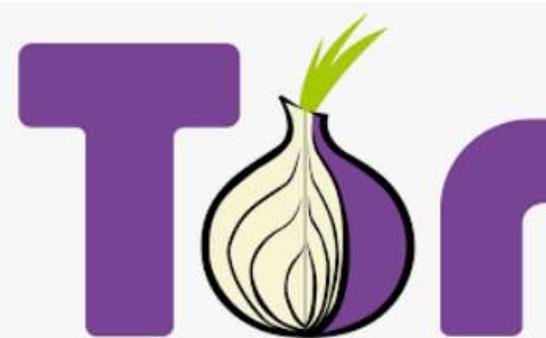
**Figure 6.6. Network level deanonymization.** As Dan Kaminsky pointed out in his 2011 Black Hat talk, “the first node to inform you of a transaction is probably the source of it.” This heuristic is amplified when multiple nodes cooperate and identify the same source.

- But that is a problem of
  - “communication anonymity”
- There is ongoing research and solutions
- Such as Tor

Tor Project | Anonymity Online

<https://www.torproject.org/> ▾

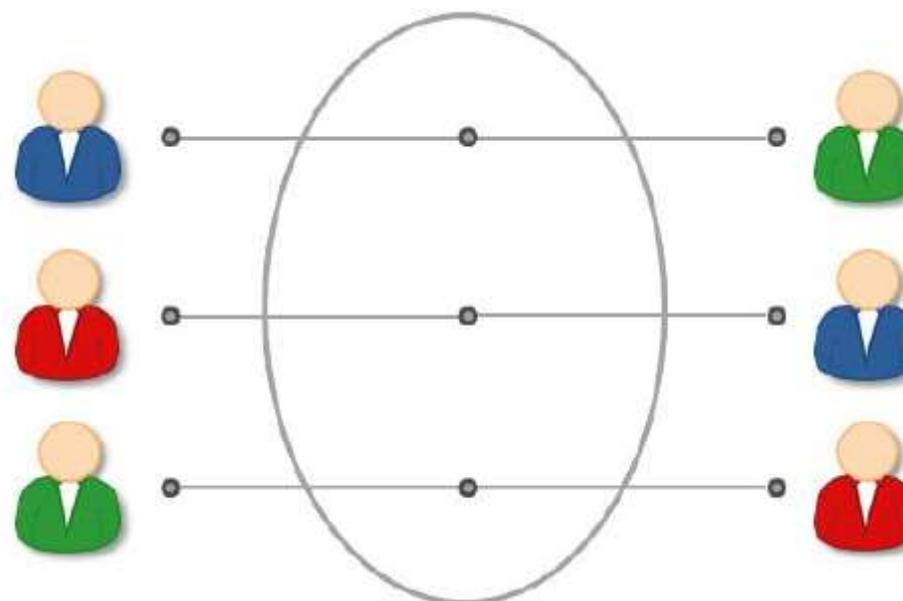
Defend yourself against tracking and surveillance. Circumvent censorship.



- Transaction Graph Analysis is more serious!

# Mixing

- Attempts to make TGA less effective
- Not specific to cryptocurrencies (anonymity)
- If you want anonymity, use an *intermediary*



## **Online wallets** as mixes; as intermediaries.

- Services where you can
  - store your bitcoins online,
  - withdraw later.
  - Typically the coins that you withdraw won't be the same as the coins you deposited.
- Support anonymity up to some degree
  - Anyways keep records, some ID info..

## Dedicated mixing services.

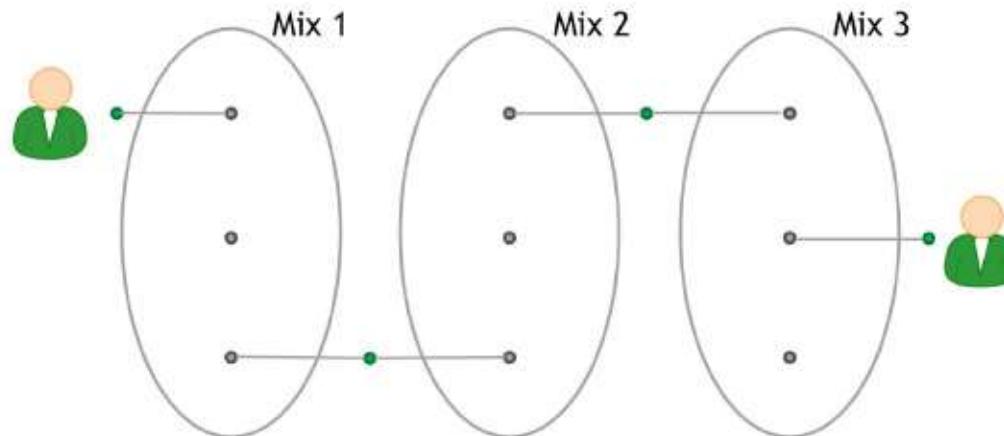
- Promise not to keep records, nor require your ID
- No even need a username or other pseudonym
- You send your bitcoins to an mix's address
- You tell the mix a destination address to send bitcoins to.
- Mix will soon send you (other) bitcoins at
- It's essentially a swap

## Dedicated mixing services.

- They “promise”
- You need to “trust”
- Mixing terminology
  - Mix vs. mixer? Mix ☺
  - Laundry? Nope!

# Use a series of mixes

- Make it even more mixed
  - Graph algorithms are demanding!
- More difficult to trace on the graph
- At least one mix supposedly destroy records
- e.g. Tor uses a series of 3 routers



# Uniform Transactions

- If transactions of
  - different users
  - different quantities
- Then it would be straightfw to trace
- Determine a ***chunk size***
- Huge size? Small transactions cannot
- Small size? Big transactions need to split a lot
- Multiple standard chunk sizes
- Tradeoff: Privacy vs. Efficiency

- ***Client side should be automated***
- Adversary observing attack
  - Not only amounts
  - But also timing
- Timing: Too complex by humans
- Interaction with mixes should be automatic

- ***Mixing Fees should be all-or-nothing.***
  - Not transaction fees but mixing fees
- Cutting a % from each transaction
  - destroy sizes of chunks
  - leads to vulnerabilities
- Instead of cutting **0.1%** of each transaction,
- Swallow **1** whole transactions among all **1000**
- Mixer should convince that 0.1%, not 1%

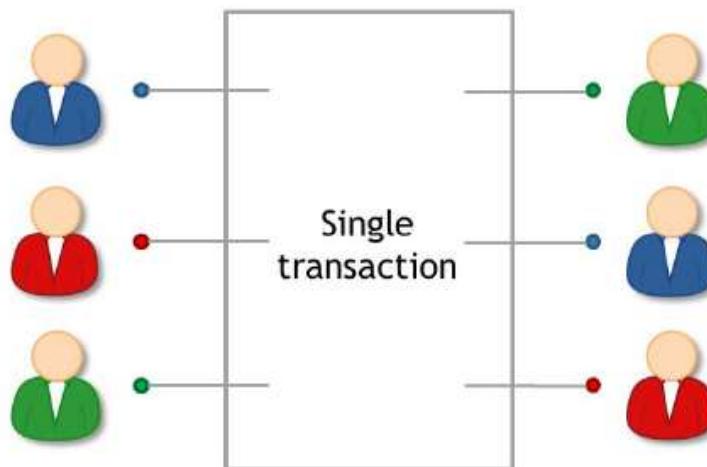
- ***Mixing in practice***
  - No truly functioning mixing service
  - Many reported to steal coins
  - Users do not prefer mixing services
  - ***Anonymity loves company***
  - No strong anonymity with few users
- Mixers are all independent
- Not following the model

# Decentralized Mixing

- Idea:
  - Replace mixers with a P2P network of users
  - Fits better to blockchain's philosophy
- Practical advantages
  - No need to wait for trusted service (bootstrapping)
  - Theft is impossible
- Several proposals, main: ***Coinjoin***

# Coinjoin

- Different users jointly create single transaction
- Order of inputs & outputs randomized
  - Participants check output addresses & amounts



*Figure 6.9. A Coinjoin transaction.*

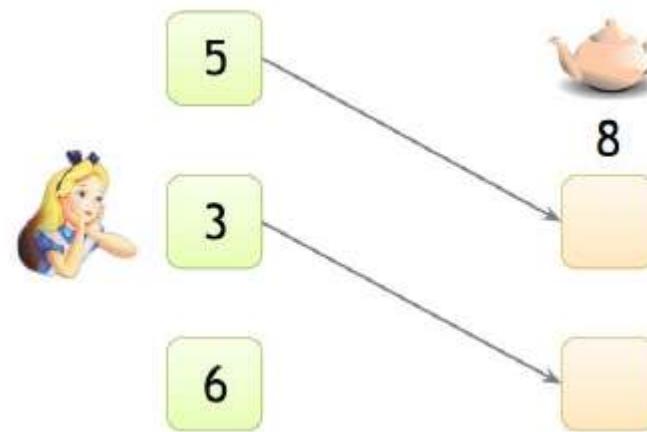
- Five steps of Coinjoin
  - Find peers who want to mix
  - Exchange input/output addresses
  - Construct transaction
  - Send the transaction around. Each peer signs after verifying their output is present. !!!
  - Broadcast the transaction

# High Level Flows

- Side channels can be very tricky
  - Suppose Alice receives 43.12312 BTC, each month
  - She immediately transfers 5% of that amount to her retirement account, which is another Bitcoin address.
- No mixing strategy can effectively hide the relationship between these two addresses
- The specific amounts and timing are extraordinarily unlikely to occur by chance.

# Merge Avoidance

- Normally: Combine inputs for a single output
- Why not receiver provides many addresses?



- Alice makes payments at arbitrary times

# Zerocoins and (its successor) Zerocash

- “Ingenious Cryptography”
- Others try to add anonymity on the protocol
- Zerocoins adds mixing at the protocol level
- Compatibility:
  - Zerocoins not directly compatible with Bitcoin
  - Can become by a soft fork
  - Zerocash cannot. It requires an altcoin
- Cryptographic guarantees
  - Protocol level mixing
  - No need to trust to anyone (mixes, intermediaries)

# Zerocoins (extension of Basecoin)

- Basecoin: A Bitcoin like altcoin
- Convert Basecoins → Zerocoins → Basecoins
  - Link between original and new Basecoin is broken
- Basecoin acts like the currency
- Zerocoins provides a way for anonymity
- Your Zerocoins prove:
  - You owned Basecoins (not which)
  - You made them unspendable (Casino example)

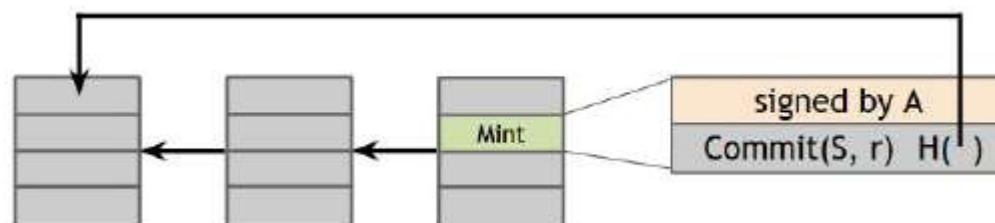
- **Zero-knowledge proofs.**
  - The key cryptographic tool
  - A way for somebody to prove a (mathematical) statement
  - Without revealing any other information that leads to that statement being true.
  - We'll treat ZKP as a black-box

***“I know  $x$  such that  $H(x)$  belongs to the following set: {...}”.***

- Minting Zerocoins
  - Anyone can mint at any time, for any value
  - Suppose 1 Basecoin is worth 1 Zerocoins
  - You can mint 1 Zerocoins (free money? ☺ )
  - To put in blockchain, have to give up 1 Basecoin
- Cryptographic commitments



- Three steps
    - Generate serial number  $S$  and a random secret  $r$
    - Compute  $\text{Commit}(S, r)$ ,
    - Publish the commitment onto the block chain
- This burns a basecoin, making it unspendable, and creates a Zerocoins. Keep  $S$  and  $r$  secret for now.

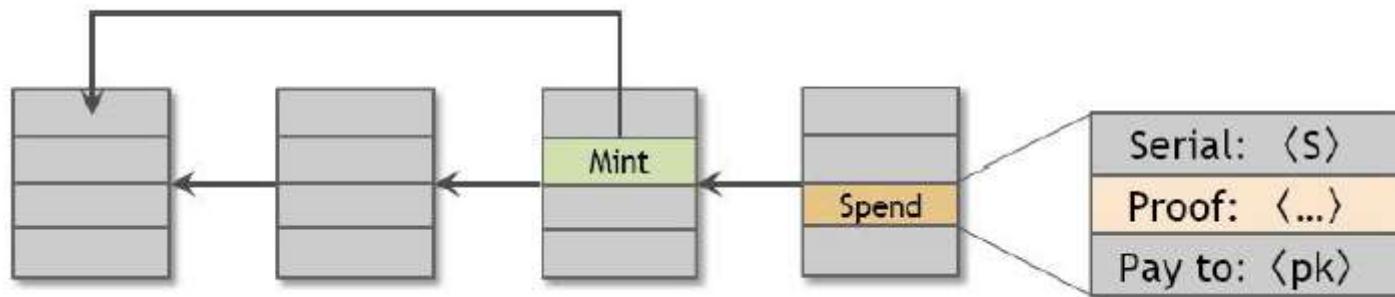


# To spend a Zerocoins

- To redeem a new basecoin,
  - need to prove that previously minted a zerocoins.
  - You could do this by opening your previous commitment, that is, revealing  $S$  and  $r$ .
  - But this makes the *link* between your old basecoin and your new basecoin apparent.
  - How can we break the link?
  - Zero-knowledge-proof!

- Create a special “spend” transaction that contains  $S$ , along with a zero-knowledge proof of the statement:  
“I know  $r$  such that  $\text{Commit}(S, r)$  is in the set  $\{c_1, c_2, \dots, c_n\}$ ”.
- Miners will verify your proof which establishes your ability
  - To open one of the zerocoins commitments on the block chain,
  - without actually opening it.
- Miners will also check that  $S$  has never been used
  - (double-spend)
- The output of your spend transaction → a new basecoin.  
For the output address, you should use an address that you own.

- Spending a Zerocoins



- **Anonymity.**
  - Observe that  $r$  is kept secret throughout;
  - Neither mint nor spend transaction reveals it.
  - Nobody knows which s.number  $S$  corresponds to which zerocoins.
  - This is the key concept behind Zerocoins’ anonymity.

# Zerocash

- A different anonymous cryptocurrency
- Builds on the concept of Zerocoins
- Takes the cryptography to the next level.
- It uses a zero-knowledge SNARKs (zk-SNARKS)
  - which are a way of making zero-knowledge proofs much more **compact** and **efficient** to verify.
- No need to a Basecoin

System	Type	Anonymity attacks	Deployability
Bitcoin	pseudonymous	transaction graph analysis	default
Manual mixing	mix	transaction graph analysis, bad mixes/peers	usable today
Chain of mixes or coinjoins	mix	side channels, bad mixes/peers	bitcoin-compatible
Zerocoin	cryptographic mix	side channels (possibly)	altcoin, trusted setup
Zerocash	untraceable	none known	altcoin, trusted setup

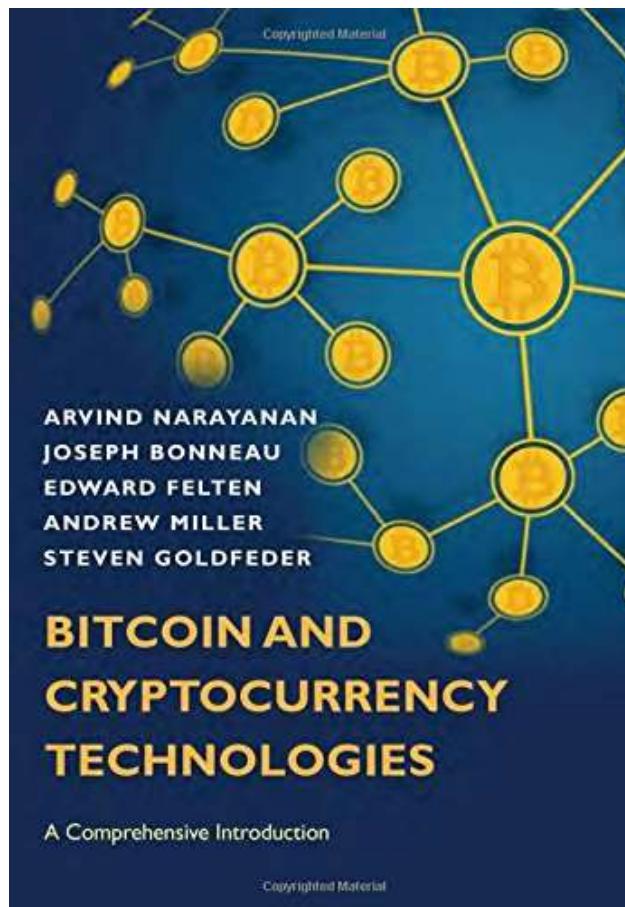
*Table 6.14: A comparison of the anonymity technologies presented in this chapter*

# Blockchain & Business Application

Lecture:

## **Alternative Mining Puzzles**

# Chapter 8



- Mining Puzzles
- Control the Consensus Process
- Profit
  - Help solving, not only network maintenance
- Modifying/Designing puzzles!

# Puzzle Requirements

- Secure?
- Difficult to solve, easy to verify
- Adjustable difficulty

# *(From Mining Chapter)*

- Network grows, hardware faster, but difficulty increases → Next block always in 10 mins

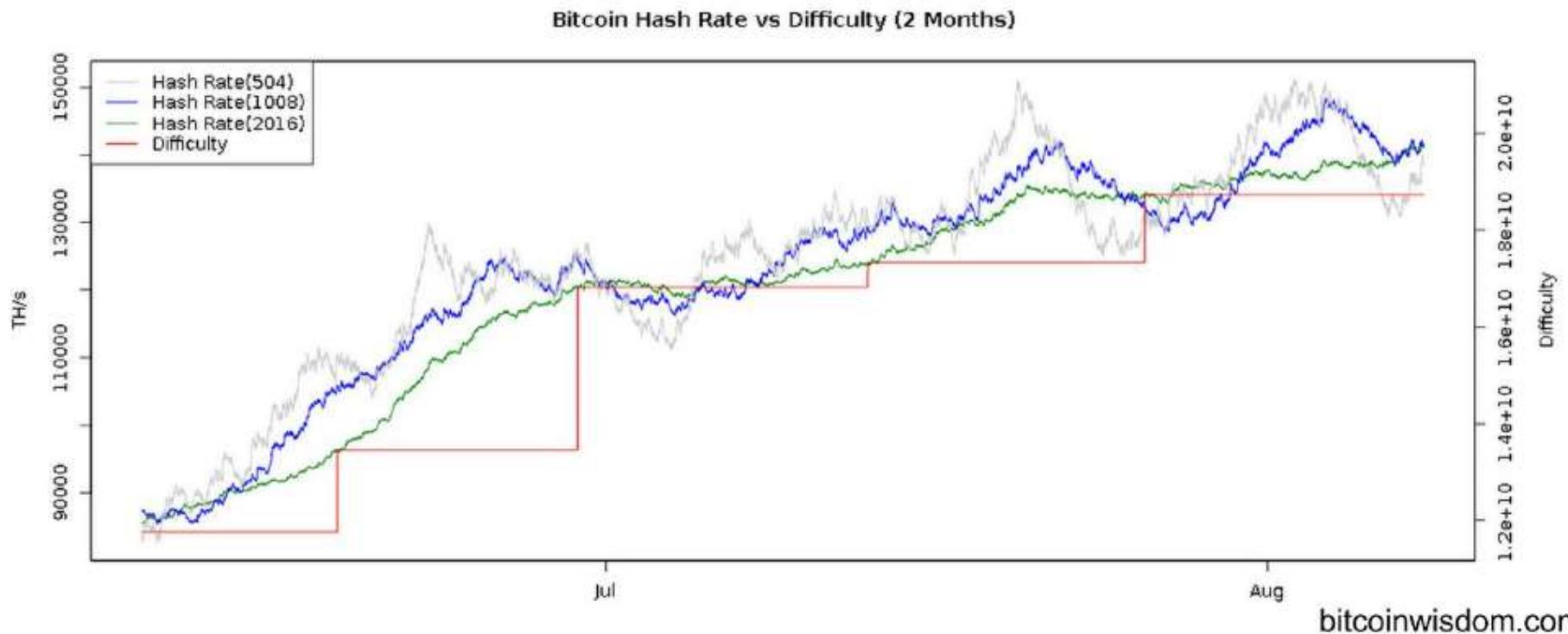


Figure 5.3: Mining difficulty over time (mid-2014). Note that the y-axis begins at 80,000 TH/s.

# What is Bitcoin Puzzle?

- “Partial hash-preimage puzzle”
- Goal:
  - to find preimags for partially specified hash output
  - namely, an output below a certain target value
- SHA-256 hash based puzzle satisfies both
  - Adjustable difficulty
  - Verification (checking solutions) trivial

# Additional Requirement

- Progress-freeness:
  - Chance of winning: Only roughly proportional to hash-power
  - Small miners also should have some chance (r.p.)
  - Otherwise, no small/starters would join
  - Trial&Error
  - No reward for past efforts (*don't confuse pools*)
  - Mathematically: Memoryless

- SHA-256 hash based puzzle satisfies all three
  - Adjustable difficulty
  - Fast verification
  - Progress-free
- Transition
  - From “one-CPU-one-vote”
  - To powerful minority

# ASIC Resistance

- Ideal Case
  - Each “computer” shall have equal power
  - Even recent CPU’s are optimized for cryptography
  - Can require to be a traditional computer? NO.
- Modest Case
  - Reduce the ASIC-CPU gap
  - Allow ASICs be more efficient only one order
  - Let everyone join the game

# *“But the Memory Remains”*

- Processor technology
  - Improving fast
  - Costly
- Memory: slow progress, cheap
- Memory-bound
  - Memory access time dominates the total time
- Memory-hard
  - Require large memories
- We would require both memory-bound&hard

- SHA-256
  - Requires only 256 bits (fits to CPU registers)
- Scrypt
  - The most popular memory-hard puzzle
  - Used in Litecoin and some other altcoins
  - Already used for password-hashing; *why?*

# Memory turns $O(N^2)$ to $O(N)$

Figure 8.1: Scrypt pseudocode

```
1 def scrypt(N, seed):
2     V = [0] * N // initialize memory buffer of length N

        // Fill up memory buffer with pseudorandom data
3     V[0] = seed
4     for i = 1 to N:
5         V[i] = SHA-256(V[i-1])

        // Access memory buffer in a pseudorandom order
6     X = SHA-256(V[N-1])
7     for i = 1 to N:
8         j = X % N // Choose a random index based on X
9         X = SHA-256(X ^ V[j]) // Update X based on this index

10    return X
```

- Having a memory of  $N/k$
- Compute  $(k+3) N/2$
- Halving memory requirement:
  - $k \rightarrow 2k$
  - Computations:  $4N/2 \rightarrow 5N/2$
- Memory  $\rightarrow$  Zero, if  $k \rightarrow N$ 
  - Computations:  $2N \rightarrow (N+3) N/2 = O(N^2)$

# Additional Limitation of Scrypt

- Verification Cost
  - Requires as much memory to verify as to compute
- Drawback:
  - Verifiers require large memory
  - Takes long time to verify&propogate
  - Forking more likely

- Until recently no puzzle known to be
  - Solving: Memory hard/bound
  - Verifying: Memory easy
- Based on Cuckoo Hash Table (2001)
- A new puzzle Cuckoo Cycle proposed (2014)
  - No known way to compute without a large table
  - Easy to verify
- However no proof yet if solved without memory
- It takes time to be trusted & become common

# *Scrypt* in Practice

- ASICs already appeared for efficient Scrypt
  - Revealed that it is not ASIC-resistant (LiteCoin)
  - No (A.R.) advantage over Bitcoin anymore
- 
- Reason: Scrypt required only 128kB
  - Exploit the tradeoff between memory & PU
  - ASICs designed/optimized

# Other Approaches for ASIC-R

- In addition to memory hard/bound?
- Design puzzle that makes hard to design ASIC
- X11. Used in DarkCoin (DASH)
  - Uses 11 different hash functions
  - So that it is inconvenient for ASIC designers
  - But once designed.. Oops!

# 11 Hashes?

- US National Inst. of Standards ran competition
- Design and submit
  - Design document
  - Source code
- Many candidates
- 24-winners
  - No known cryptographic attack

# Another Approach

- “Moving Target” (not implemented yet)
- Not only the difficulty level but also
- Change the puzzle!
- e.g. **Pick** one among 24-winners
- How?
  - Centralized authority?
  - Periodically change with a schedule?

# *ASIC Honeymoon*

- Despite potential market for X11
- No ASIC for X11 yet
- A new ASIC:
  - High cost & long time to design
  - Low cost to produce
- Eventually there will be ASICs for each C.C.
- A honeymoon for each ASIC

# Arguments against ASIC-R

- It may be impossible
- It may be risky (already proven SHA256)
- Security vs. value
  - If attackers hack the CC, its value drops ☺
- ASIC-friendly puzzles
  - ASICs efficient only for mining

# Proof-of-*Useful*-Work

- Current CCs terrible for the environment
- Any puzzle allows recycling, appreciated
- Use idle computers (spare-cycles) older idea
- Volunteering for society
- Design new such puzzles

Project	Founded	Goal	Impact
Great Internet Mersenne Prime Search	1996	Finding large Mersenne primes	Found the new “largest prime number” twelve straight times, including $2^{57885161} - 1$
distributed.net	1997	Cryptographic brute-force demos	First successful public brute-force of a 64-bit cryptographic key
SETI@home	1999	Identifying signs of extraterrestrial life	Largest project to date with over 5 million participants
Folding@home	2000	Atomic-level simulations of protein folding	Greatest computing capacity of any volunteer computing project. More than 118 scientific papers.

**Table 8.3: Popular “Volunteer computing” projects**

# SETI@home, candidate?

- Huge computational power by people
- Statistical anomalies, difficult to find
- Drawbacks:
  - SETI has a fixed raw data (by radio telescopes)
  - Some segments may be more likely
    - Not “progress-free”
  - “Central” & “trusted” administration
- Prime numbers?

# Great Internet Mersenne Prime Search

- Infinite numbers, primes, Mersenne numbers
  - Puzzle space is inexhaustable
- Drawbacks
  - Rare
  - Loong time to find
  - GIMPS found only 14 ***Mn*** in 18 years!

# Primecoin

- Challenge: Find a Cunningham chain
- A sequence of  $k$  prime numbers  $p_1, p_2 \dots p_k$
- $p_i = 2p_{i-1} + 1$
- 2, ... ?
  - Length:  $k=5$
  - Next number: 95, not a prime, end of the chain
- Longest known:  $k=19$ , starts at
  - 79910197721667870187016101

- Not proven but believed that infinite chains
- Been used in Primecoin since 2014
- Most Cunningham chains found since then
- Variations emerged:

$$p_i = 2p_{i-1} - 1$$

- Maybe used widely in future
- !! No known practical applications

# Permacoin and proof-of-storage

- What if we could design a puzzle that required storing a large amount of data to compute?
- A large file  $F$ 
  - Public
  - LHC's PB-large experimental data?
- Difficult to store  $F$ 
  - Store  $H(F)$
  - Or even store  $F$  as a Merkle tree & store root
- And some technical cryptographic details..

# Public Good (PG)

*Any proof-of-useful-work should be pure PG*

- Non-excludable
  - Nobody can be prevented from using it
- Non-rivalrous
  - Good's use by others does not affect its value
- Lighthouse ☺
- Is “protein folding” a pure PG?

# Long Term Challenges & Economics

- Proof-of-useful-work
  - Natural goal
  - But challenging as different requirements
- Primecoin & Permacoin, candidates
  - Technical drawbacks (primes, rare)
  - Too minor public benefits (where is PG?)

# Nonoutsourceable Puzzles

- Preventing the formation of mining pools
  - Most miners tend to join pools
  - Dangerous trend; threat to Bitcoin's philosophy
- A large pool
  - Can attack the network (implementing strategies)
  - Pool operators may cheat
  - Target for hackers
  - Selling your vote?
- How to prevent the pools?

# (Revisiting Mining Pools)

- A pool operator
- Members mine
- Send their partial solutions (proof)
- When one participant finds valid block
- Revenue distributed among members

# Existence of Pools

- Two technical properties of Bitcoin
- Members can easily prove
  - their efforts
  - that the efforts are for the blocks of the pool
- Sabotage?
  - Member always send the shares, never valid block
  - Loss of the whole pool

- Sabotage, vandalism?
  - Loss to the pool, loss to himself/herself
  - Really?
- Surprisingly, it can be profitable!
  - Consider two pools, **A** & **B**, each with 50% power
  - **A** dedicates its 25% power to **B** (discarding blocks)
  - **A** makes profit: 5/9
  - More than half: 4.5/9

- A new puzzle design
  - Members mine in pool but not submit valid blocks
  - Manager knows the secret key, and distributes
  - Puzzle requires/lets members know the secret key
- Change the puzzle from
  - “find a block with hash is below a certain target”
- To
  - “find a block for which the hash of *a signature on the block* is below a certain target.”

- Manager can
  - a) Distribute the key:
    - Members can steal coins!
  - b) Perform signature calc's for members
    - Too much effort, better to mine solely
- This “non-outsourceable” prevent pools with untrusted members

- In current situation, this may cause opposite:
  - Individual miners don't join pools, don't mine
  - Only few & large pools can survive
  - Centralization!
- We do not know the solution yet ☹

# Proof-of-Stake and Virtual Mining



Figure 8.5: The cycle of Bitcoin mining

- Why not simply allocate mining “power” directly to all currency holders in proportion to how much currency they actually hold?



Figure 8.6: The virtual mining cycle

# Advantages

- Remove wasteful right half → Environment
- No ASIC, No AR → Centralization
- CC's value → Miners tend to behave good

# Implementing Virtual Mining

- Not
  - researched scientifically
  - analyzed practically
  - (Bitcoin is too dominant)
- Peercoin (2012)
  - Hybrid: proof-of-work & proof-of-stake
  - Coin-age, coin-stake: solving & adjusting difficulty

# Alternative forms of stake

- Proof-of-Stake
  - Similar to Peercoin but no coin-age
  - The richer, the easier to solve (become richer)
- Proof-of-deposit
  - When coins are used for a block, become frozen for some time
  - Reward miners who are willing to keep coins unmoved

# Drawbacks of Virtual Mining

- Nothing-at-stake
  - Always attempt to fork
  - Low probability to gain
  - Nothing to lose
  - (no opt.cost as in traditional mining)
- Save-up to burst power
- Once 51%, keep it forever

# Can V.M. work?

- Some believe real resources pay for security
  - Not proved ☺

# Blockchain & Business Application

Lecture:

## **Community, Politics, and Regulation**

# Consensus in Bitcoin

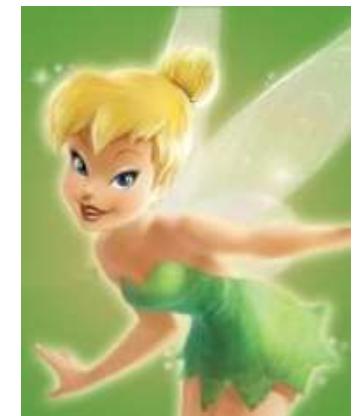
- **Consensus about rules.**
  - *Rules*: what makes a transaction or a block valid,
  - the core protocols and data formats
- Required among all different participants
- Talk to each other and agree on things.

# Consensus in Bitcoin

- **Consensus about history.**
  - What is and isn't in the block chain,
  - → which transactions have occurred.
  - → which coins — which unspent outputs — exist
  - who owns them.

# Consensus in Bitcoin

- **Consensus that coins are valuable**
  - When you give bitcoins, you take sth
  - Now & tomorrow?
- Fiat money: Not by consensus but by fiat
  - What is dollar, what is not?
  - Determined by law, not history
- Cryptocurrency:
  - You believe that tomorrow too people believe that you believe that... Circular!



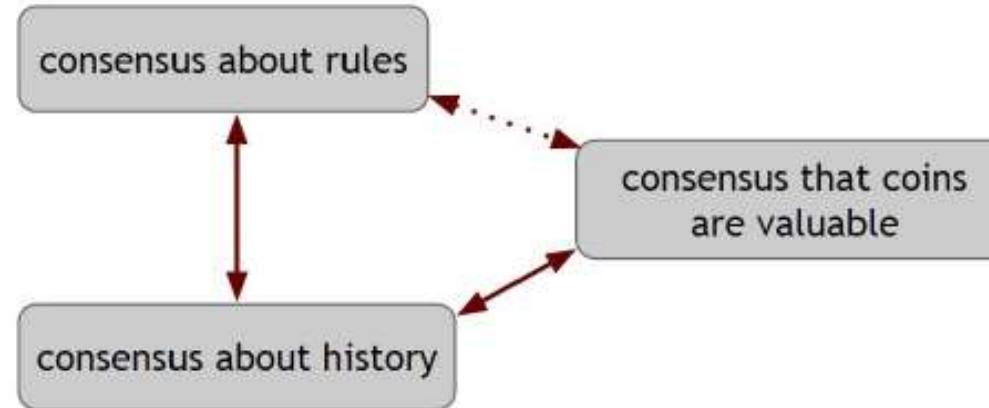


Figure 7.1: Relationships between the three forms of consensus in Bitcoin

- Without knowing which blocks are valid you can't have consensus about the block chain.
- Without consensus about which blocks are in the block chain, you can't know if a transaction is valid or double-spend attempt.

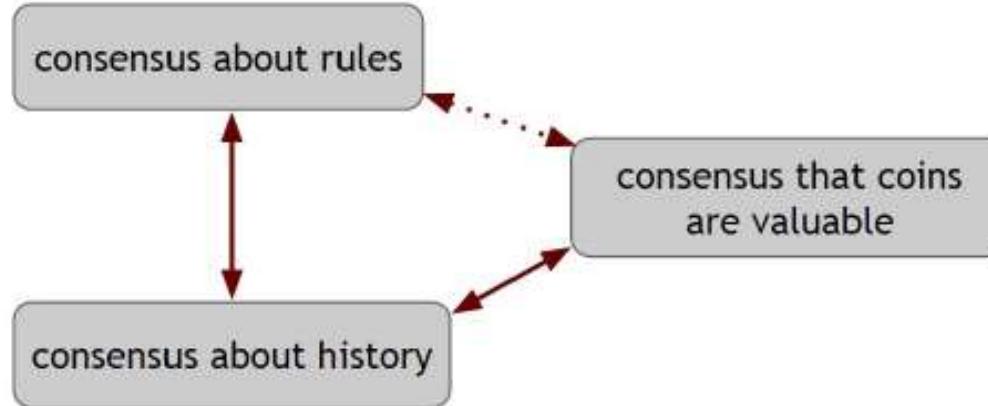


Figure 7.1: Relationships between the three forms of consensus in Bitcoin

- Prerequisite for believing that the coins have value: We agree on who owns which coins
- Consensus about value: motivation for miners  
→ maintain block chain: consensus about history

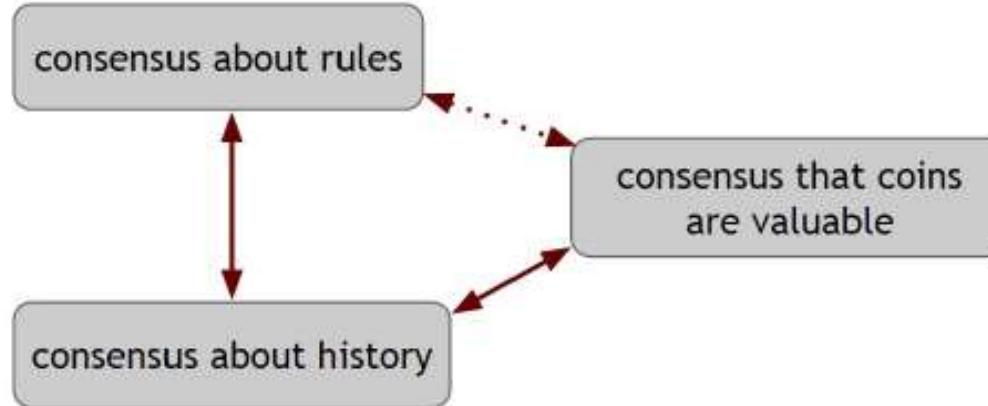


Figure 7.1: Relationships between the three forms of consensus in Bitcoin

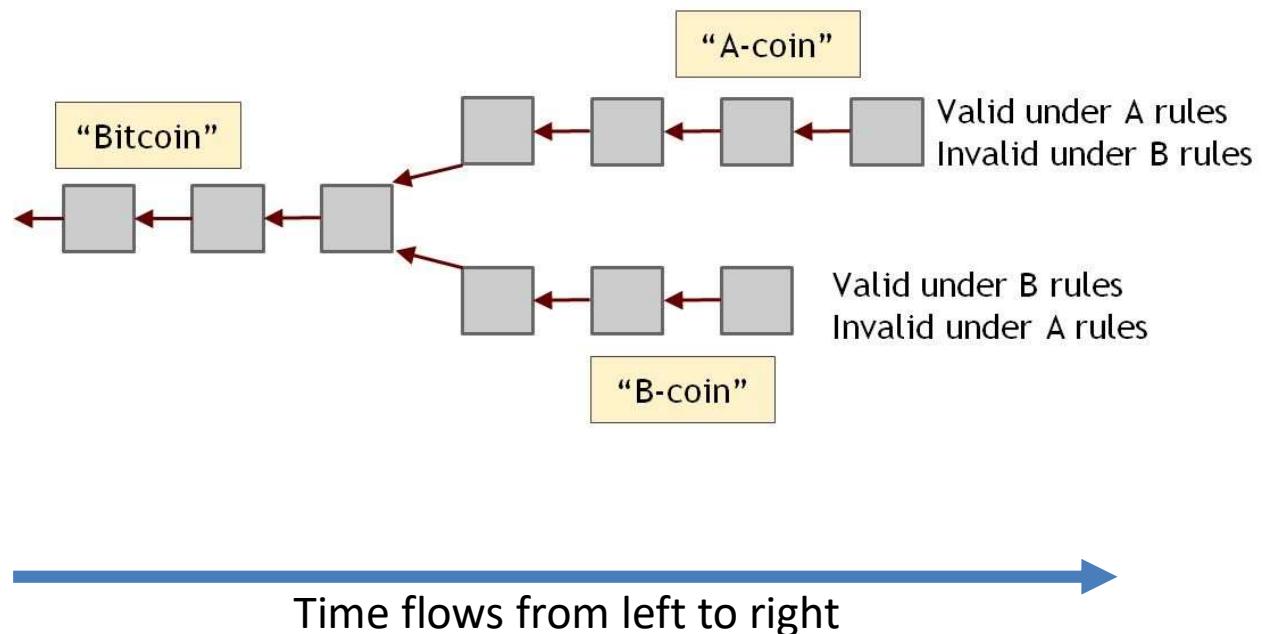
- None can be obtained alone.
- C. about rules & C. about value are loosely tied

# Bitcoin Core Software

- MIT's open-source license
- Core is the de-facto rule book
- Reference for Altcoin developers
- Improvements
  - Pull Requests (as all open-licensed software)
  - Bitcoin Improvement Proposals for big changes

- Bitcoin Core Developers
  - Nakamoto is inactive
  - 5 core developers, powerful?
- Modifying the currency?
  - Fiat money: Nope
  - Cryptocurrency: Possibly

- Disagreement on rules → Can “fork”



# Stakeholders: Who's in Charge?

- Core developers (rules)
- Miners (history)
- Investors (value)
- Merchants & customers (demand)
- Payment services

# Bitcoin advocacy groups

- Bitcoin Foundation (2012, nonprofit)
  - Funding some core developers full-time
  - Talking to governments
- Coin Center (2014, nonprofit)
  - Talking to government
- Should Bitcoin talk to governments?

# Roots of Bitcoin

- Libertarian (even anarchist)
  - “Little or no government possible with crypto”
- Satoshi Nakamoto
  - Claimed to be born in 1972; Japanese, male
  - No real information
  - Acquired lots of coins (early mining), never spent
  - Addresses public
  - Cannot exchange to dollar without revealing ID

# Governments Notice Bitcoin

- Capital control
  - Transfers between countries
  - Disconnecting with fiat money?
- Crime
  - Ransoms
  - Silk Road “the eBay for illegal drugs”
  - <http://silkroad....onion>
    - Escrow
    - 2011-2013, owner arrested

Welcome | Silk Road State of the Road Address | + | [silkroadvb5piz3r.onion](#) | [star](#) | [C](#) | [G](#)

# Silk Road

anonymous marketplace

Welcome [redacted]  
[messages\(0\)](#) | [orders\(0\)](#) | [account\(\\$0.00\)](#) | [settings](#) | [log out](#)

search | [\(0\)](#)

---

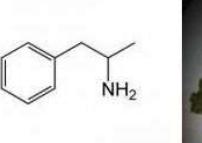
Shop by category:

- Drugs(1582)
  - Cannabis(271)
  - Dissociatives(33)
  - Ecstasy(217)
  - Opioids(106)
  - Other(65)
  - Prescription(274)
  - Psychedelics(306)
  - Stimulants(190)
- Apparel(37)
- Art(1)
- Books(300)
- Computer equipment(9)
- Digital goods(218)
- Drug paraphernalia(33)
- Electronics(13)
- Erotica(165)
- Fireworks(1)
- Food(1)
- Forgeries(34)
- Hardware(1)
- Home & Garden(5)
- Lab Supplies(5)
- Medical(3)
- Money(89)
- Musical instruments(2)
- Packaging(1)



10 Grams high grade  
MDMA 80+%

**\$61.17**



Amphetamines sulfate /  
Speed freebase...

**\$28.59**



2g Jack Frost (weed) \*420  
SALE\*\*\*\*

**\$8.54**



5 Grams of pure MDMA  
crystals

**\$42.04**



100 red Y tablets 111mg  
(lab tested)...

**\$97.77**



Michael Jackson  
Discography 1971-2009...

**\$2.52**



3.5g Albino Rhino (weed)

**\$12.37**



10mg Flexeril (muscle  
relaxant)...

**\$3.22**



\*\*\*10gr. Amphetamine  
Sulphate...

**\$33.19**

News:

- The gift that keeps on **giving**
- Who's your **favorite?**
- Acknowledging **Heroes**
- A new anonymous market **The Armory!**
- **State of the Road Address**

# Anti-Money Laundering

- Try to make organized crime more difficult
- Know your customer laws
  - Identify and authenticate clients
  - Evaluate risk of client
  - Watch for anomalous behavior
- Mandatory reporting
  - Transactions over \$10K
  - Structuring clients requested

# Regulation

- “Some dumb bureaucrat who doesn't know my business or what I'm trying to do, coming in and messing things up”
- Some can be justified
  - Market failures
  - Pareto improvements: Allocation of goods for all

- Lemons market
  - e.g. Only high&low quality cars
- How to fix?
  - Seller reputation
  - Warranties
- Regulatory fixes
  - Quality labels
  - Quality standards
  - Force the warranties

# Reading Assignments

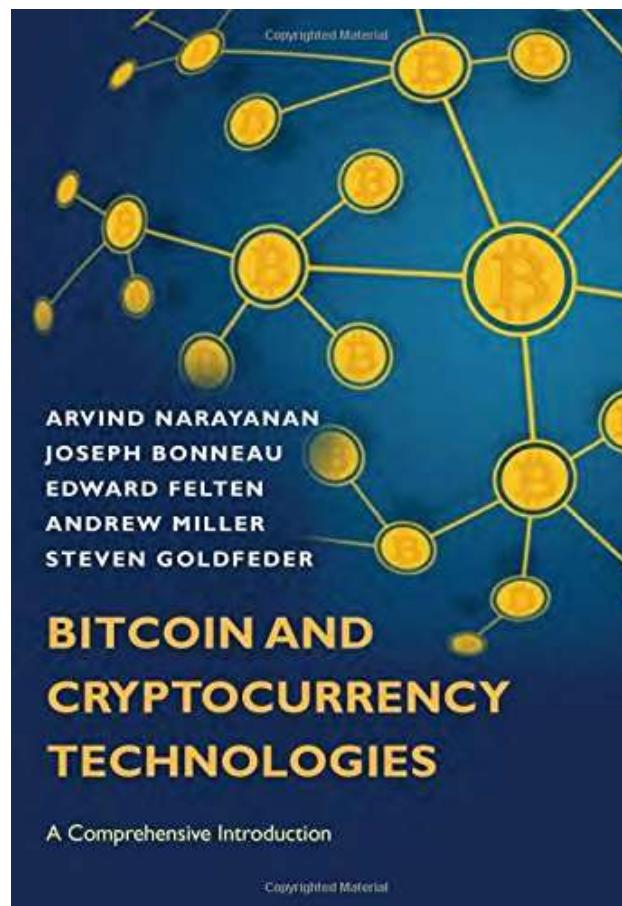
- The economics of information security
- Traveling the Silk Road: A measurement analysis of a large anonymous online marketplace
- Measuring the Longitudinal Evolution of the Online Anonymous Marketplace Ecosystem
- (Guide) Bitcoin: A primer for policymakers
- (Book) Crypto: How the Code Rebels Beat the Government—Saving Privacy in the Digital Age
- Bitcoin: Economics, Technology, and Governance
- How governments can leverage policy and blockchain technology to stunt public corruption

# Blockchain & Business Application

Lecture:  
**Bitcoin as a Platform for  
General Business Applications**

# Chapter 9

## Bitcoin as a Platform



- So far, Bitcoin as a currency (CC)
  - What other possible applications?
- 
- Some already implementable
  - Some needs modifications

- Consider Bitcoin as an *Append-Only-Log*
- Once data written
  - Tamper-proof
  - Forever available
- Everything in order
  - Hash pointers,
  - Not timestamps
    - miners can lie about timestamps,
    - miners' clocks may not be synchronized,
    - latency on the network.
  - If timestamp too weird (1 hour difference) rejected

- We want to be able to prove that:  
**We know some value  $x @ T$ .**
- Might not want to *reveal*  $x @ T$ .
- Instead, we only want to reveal  $x$  when we actually make the proof  $@T+t$
- However, once proof is OK, we want the evidence to be permanent

# Recall committing data

- Instead of publishing  $x$ , we can publish  $H(x)$
- Later, no  $y$  such that  $x \neq y$  but  $H(x) = H(y)$
- $H(x)$  reveals no info about  $x$ .

# Applications of timestamping 1

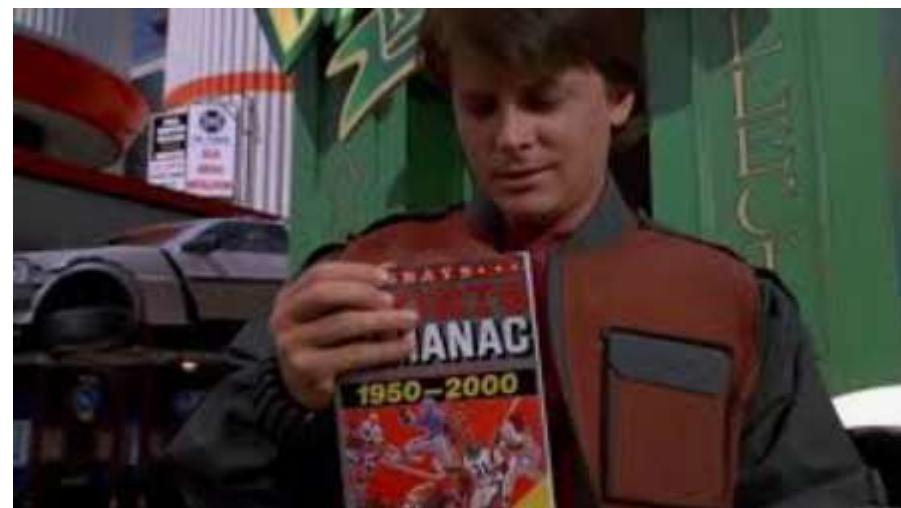
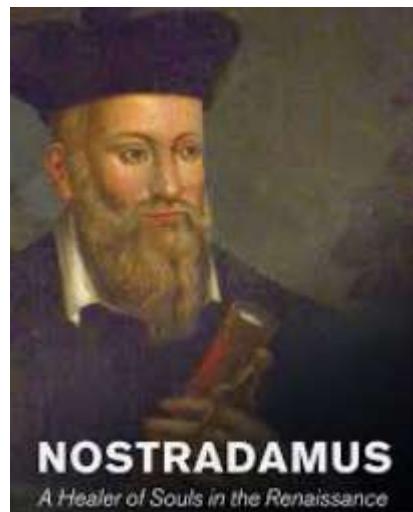
- Suppose we wanted to prove that some invention we filed a patent on was actually in our heads much earlier
- Publish the hash of the design document
- Later anytime, publish the design document
- Anyone can compute  $H(d. \text{ doc})$  and compare

# Applications of timestamping 2

- Suppose Alice hires Bob to perform a programming job;
- Their contract requires Bob to submit his work to Alice by a specific time.
- A&B published the hash of Bob's submitted work signed by both
- If any dispute later, straightfw to check

# Attacks on Proof-of-Clairvoyance

- Clairvoyance: Ability of predicting the future!
- Publish a commitment about a future event
- After it occurred, publish your words



- A Twitter account attempted to “prove” 2014 FIFA Men’s World Cup Final was rigged by “predicting” the outcome of the match



**FIFA Corruption** @fifndhs

Germany will win at ET

17 hours ago · Reply · Retweet · Favorite · 12K more



**FIFA Corruption** @fifndhs

Gotze will score

17 hours ago · Reply · Retweet · Favorite · 14K more



**FIFA Corruption** @fifndhs

There will be a goal in the second half of ET

17 hours ago · Reply · Retweet · Favorite · 12K more

- How come? Is it Possible?

- Tweeted all possibilities
- Kept ones ended true
- Deleted all others

 **FIFA Corruption** @fifndhs  
Germany will win at ET  
17 hours ago · Reply · Retweet · Favorite · 12K more

 **FIFA Corruption** @fifndhs  
Argentina will win in penalties  
17 hours ago · Reply · Retweet · Favorite

 **FIFA Corruption** @fifndhs  
Gotze will score  
17 hours ago · Reply · Retweet · Favorite · 14K more

 **FIFA Corruption** @fifndhs  
There will be a goal in the second half of ET  
17 hours ago · Reply · Retweet · Favorite · 12K more

 **FIFA Corruption** @fifndhs  
Kroos will score  
17 hours ago · Reply · Retweet · Favorite

 **FIFA Corruption** @fifndhs  
Lahm will score  
17 hours ago · Reply · Retweet · Favorite

 **FIFA Corruption** @fifndhs  
Palacio will score  
17 hours ago · Reply · Retweet · Favorite

# Bitcoin

- Secure timestamping system does not tie commitments to any individual's public identity.
- If you don't reveal them, it is easy to publish a large number of commitments
- The ones you never reveal cannot easily be traced back to you.

*Anonymity vs. decentralization*

*How to achieve a secure timestamping in Bitcoin?*

# *Secure timestamping the old-fashioned way*



Figure 9.2: A timestamping service (GuardTime) that publishes hashes in a daily newspaper rather than the Bitcoin block chain.

# Bitcoin: Unspendable outputs.

- *OP\_RETURN* instruction → unspendable output
  - returns immediately with an error
  - so that this script can never be run successfully,
  - the data you include is ignored
- Cost: One transaction fee, 1 penny
- Drawback: Arbitrary data? Child pornography?
- Would you become a miner?
- U.S. Code 2252:
  - “*knowingly* possesses, or *knowingly* accesses with intent to view”

# *Overlay Currencies*

- We *can* write any data we want into Bitcoin
  - Use Bitcoin as an append-only-log
  - Build a new CC *on top of Bitcoin*
  - Without need to develop a consensus mech.
- 
- Bitcoin Miners won't validate the data (ignore)
  - Complicated logic required
  - Double spend?

# Counterparty

- All of its transactions written on Bitcoin
- In 2014, 1% of Bitcoin transactions carried it
- Counterparty's developers don't need to deal with consensus, etc. → Can develop sophisticated applications, smartcontracts, etc
- Ability to create a new CC without consensus mechanism and new miners? Looks good!
- Drawbacks?

# Bitcoins as “Smart Property”

- Use bitcoins to represent sth other than a unit of currency in the Bitcoin system
- Following transaction graph, you can trace ownership of value in the Bitcoin system over time (bad for anonymity)
- No actual “coins” just unspent transactions
- Bitcoins are not fungible (not like gold)
- Histories are different; it “may” matter

# Give a *meaning* to history of..

- Paper currency, first?
- Not jokes but some meaning
- Even a db to match s.n. → no need to stamp
- Inherit the anti-counterfeiting of paper money



Figure 9.4: An example of adding useful metadata to ordinary bank notes

# Bitcoin

- Colored coins: Color as an extra metadata
- “issuing” transaction, we’ll insert some extra metadata

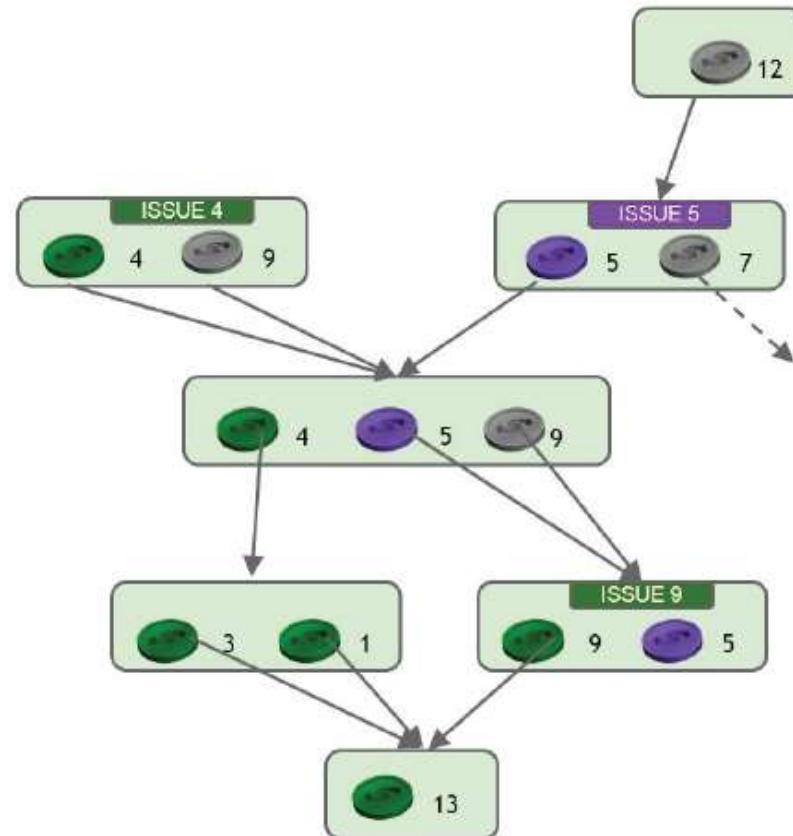


Figure 9.5: Colored coins. The transaction graph shown illustrates issuance and propagation of color

# OppenAssets

- Most popular application using “colors”
- Assets issued using special Pay-to-Script-Hash (P2SH) address
- A transaction to that address adds color
- More than one “color” is OK
- Compatible with Bitcoin (only if color, extra value)
- When using “colored coin”, use a marker
  - To prevent double payment (e.g. ticket)
- Miners don’t check colors, trust a 3<sup>rd</sup> party

# Secure Multi-Party Lotteries in Bitcoin

- Traditional betting
  - Alice and Bob want to bet five dollars.
  - They both agree to the bet in future, and the method
  - Bob will flip a coin in the air
  - while it's rotating Alice calls out “Heads” or “Tails”.
  - When the coin lands,
  - both immediately see who won
  - Both know the outcome was random (no influence)
  - Both “trust” that looser is going to pay

# Cryptographic coin flipping

- If designed, build various applications
- “Secure multiparty computation”
  - two or more mutually untrusting parties
  - each have some data
  - want to compute a result depends on all data
  - but without revealing the data to each other
- Sealed-bid auction, without a trusted auctioneer

- We might want
  - the result of the *computation* to determine a *monetary* outcome in an irrevocable way.
  - to ensure that the winning bidder in the auction pays the seller;
  - we even want to ensure that the seller's (smart) property being auctioned is automatically transferred to the winning bidder
  - to penalize parties if they deviate from the protocol.

# Coin Flipping Online

- Online game
  - Alice, Bob, and Carol
  - All want to select 0,1 or 2 with equal probability
  - They can pick large random numbers  $x, y, z$
  - Compute  $(x+y+z) \% 3$
- If each sends number simultaneously, OK.
- If not, problem!
- Solution?

- Commitment

**Round 1:**

Each party picks a large random string — Alice picks  $x$ , Bob picks  $y$ , and Carol picks  $z$ .

The parties publish  $H(x)$ ,  $H(y)$ ,  $H(z)$  respectively.

Each party checks that  $H(x)$ ,  $H(y)$ ,  $H(z)$  are all distinct values (otherwise aborts the protocol).

**Round 2:**

The three parties reveal their values,  $x$ ,  $y$ , and  $z$ .

Each party checks that the revealed values agree with the hashes published in Round 1.

The outcome is  $(x + y + z) \% 3$ .

**Figure 9.6:** Using hash commitments to implement a fair random number generator.

This protocol can be easily extended to support any number of parties.

- What if Carol gives up declaring  $z$ ?
- How to force each to declare in a limited time
  - Called “fairness” in cryptography
- Bitcoin’s “timed commitment” great for this
  - Some technical details

# Bitcoin as Public Randomness Source

## History

- NBA Draft Lottery
  - 1985 in NY
  - NY Knicks won

## Conspiracy theories

- Envelope's corner bent
- Envelope kept it freezer

- U.S. military draft lottery
  - 1969, which young men to join the Army
  - To make it “look fair”

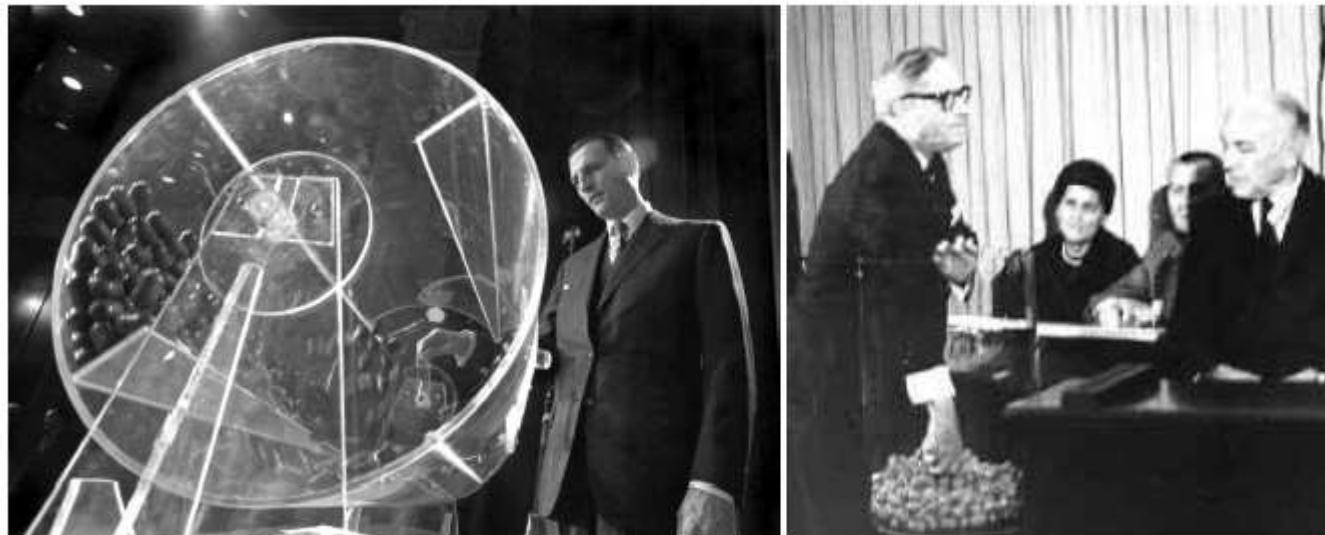
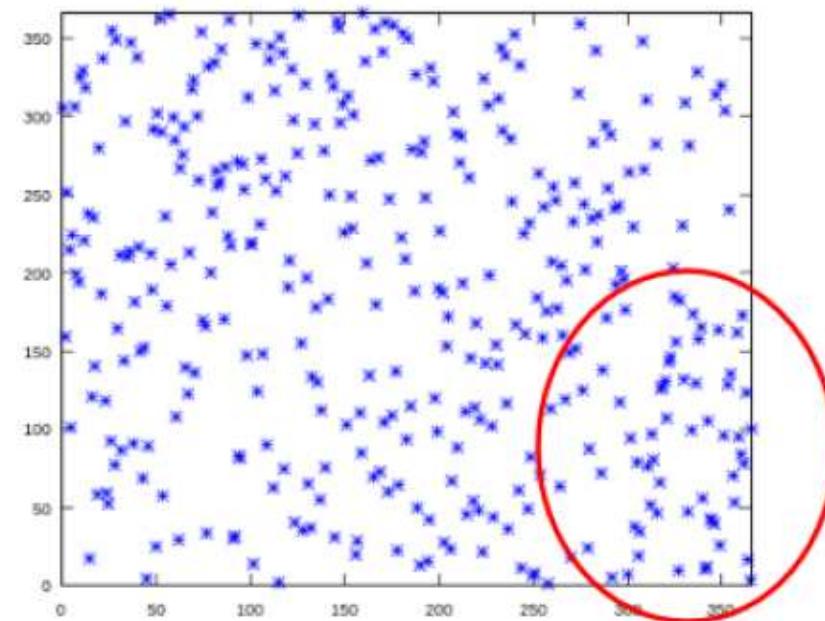


Figure 9.8: Images from the 1969 (Vietnam war) military draft lottery.

- In 1 week, statisticians observed a pattern
- Watched the tapes again and see:
  - Drums rotated exactly even numbers!



**Figure 9.9:** Statistical bias of the 1969 draft lottery. Day of the year (x-axis) versus lottery number (y-axis).

- Very difficult to prepare a “fair” setup
  - Actually, impossible with classical physics
  - *Determinism!*
  - Just ignore some parameters → pseudorandom
  - We need non-determinism
  - A quantum bit provides perfect random bit!
- Even if “fair”, how to convince the public?

# Cryptographic beacon

- If we have a perfect cryptographic beacon
  - Continuously publishing a random string
  - People could trust
  - No need to efforts drums, etc.
  - No need to cryptographic efforts
- We have no perfect (*classical*) beacon yet

# NIST Quantum Beacon '2011

- Preparing two entangled photons,
- Measuring photons' states
- Outcome is QM non-deterministic
- Do you **trust** QM
  - Uncertainty, superposition, entanglement?
- Do you **trust** that
  - Men in the lab actually doing it?
  - Not modifying the results after the measurement?

# Other ways

- Criteria
  - public observability,
  - security against manipulation,
  - an acceptable level of unpredictability
- Possible methods
  - Tomorrow's temperature?
  - Sunspots?
  - Cosmic background radiation?
  - *All actually based on QM?*

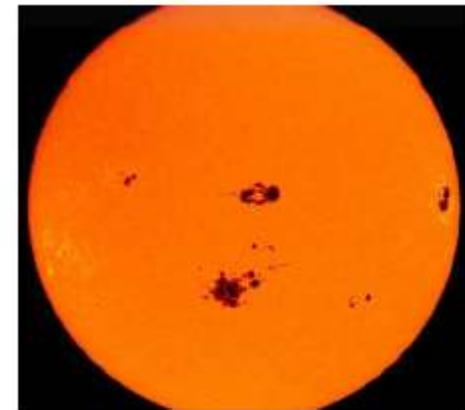


Figure 9.10: NASA image of sunspots.

# Drawbacks of natural phenomena

- Slow
  - Temp once a day?
  - Sunspots change slow
- Expertise
  - Public visible but is it really DIY?
- Measurement imprecision
- Requires **trust** to experts

# Financial Data

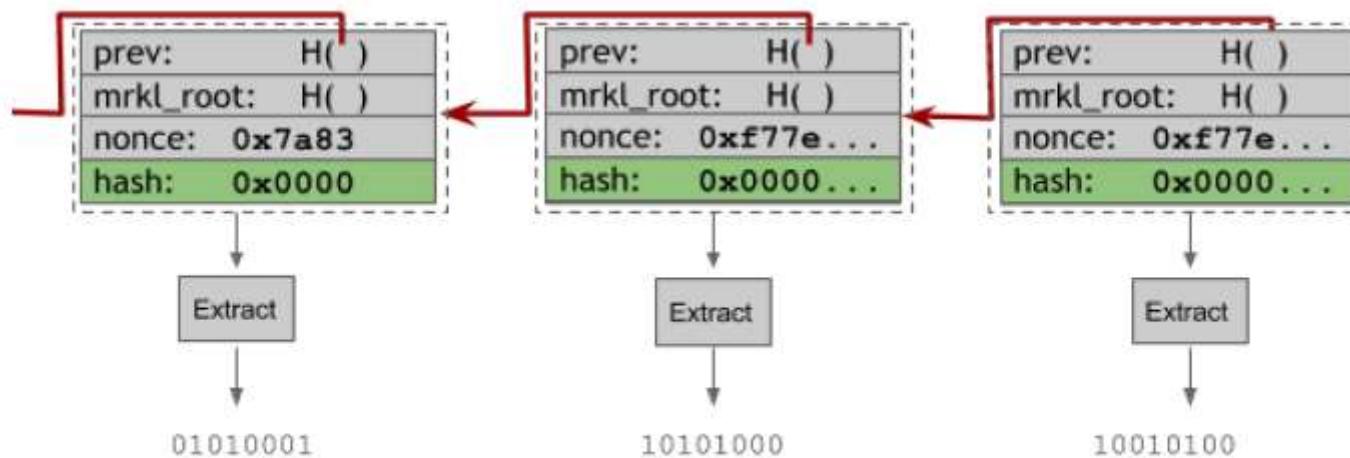
- Stock market prices
- Low-level fluctuations
  - A good level of randomness
  - Difficult to predict
- Drawbacks?

# Financial Data

- Drawbacks of low-level fluctuations
  - If you can predict, why not make direct profit
  - You manipulate stock market, need lots of \$ ☺
  - Need to **trust** the people in charge
- In summary, **trust** is needed!

# Bitcoin as a Beacon?

- Miners compute lots of hash values
- No one “can” predict the hash of next block



**Figure 9.11:** Extracting public randomness from the hashes of blocks in the block chain.

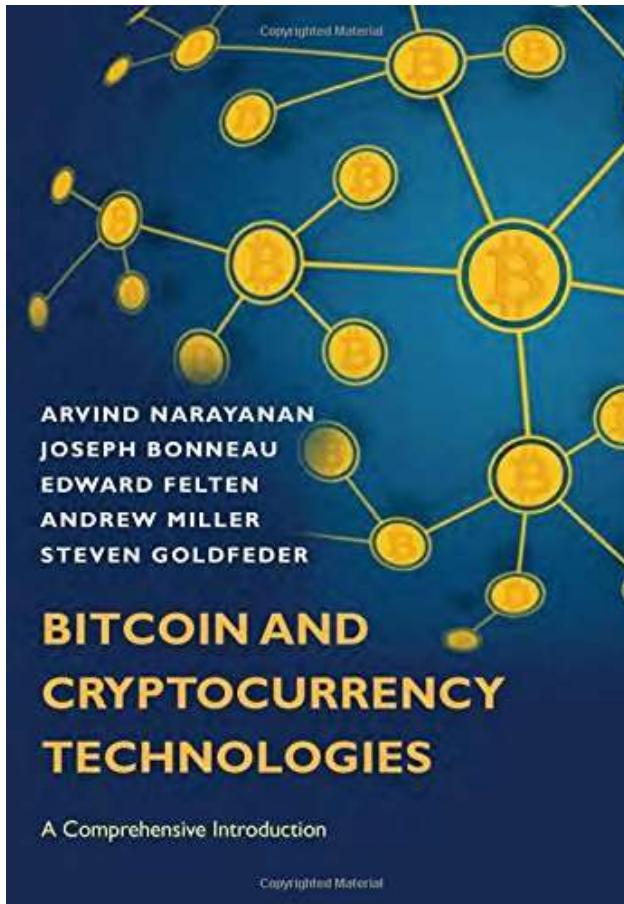
# Bitcoin as a Beacon?

- Security?
  - You find a block, you know its hash
  - Publish it, or play lottery?
- Timing?
  - Next block, exactly when?
- Pool members discarding the blocks?
- Interesting but unproven way

# Blockchain & Business Application

Lecture:  
**Altcoins and the  
Cryptocurrency Ecosystem**

# Chapter 10



- 2009: Bitcoin
- 2011: Namecoin (Bitcoin-like)
- How to count?
- What is “altcoin”? Old CC’s?

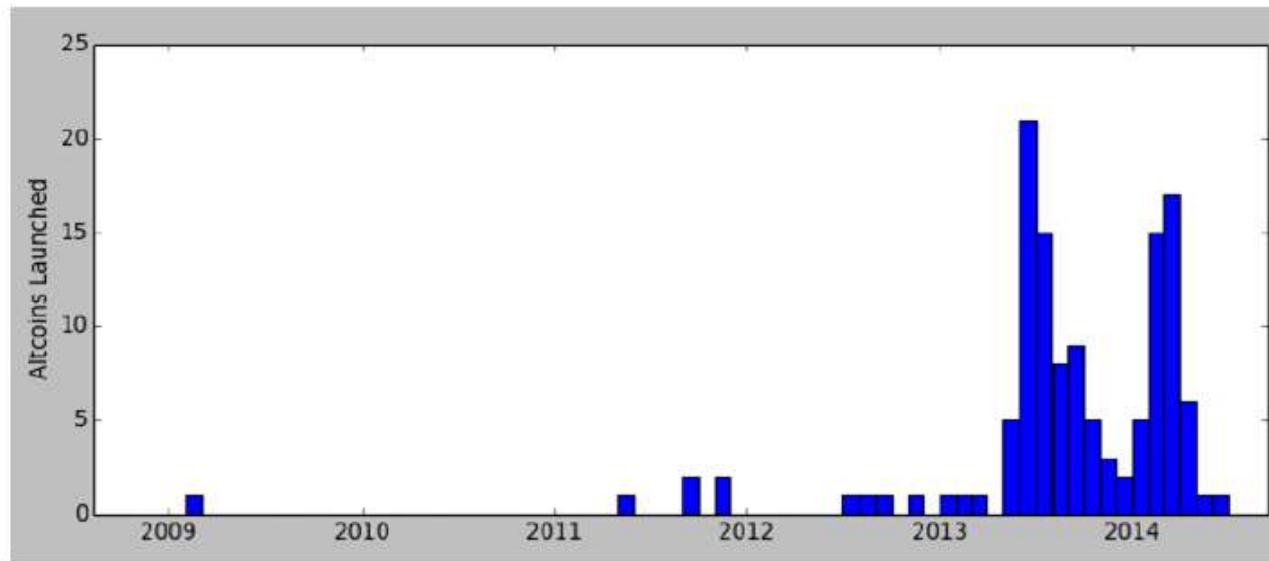


Figure 10.1: Altcoins launched per month (measured by genesis block creation).

RAHUL GUPTA, IIT-BHU

# coinmarketcap.com (June 2019)

Cryptocurrencies ▾		Exchanges ▾	Watchlist	USD ▾		← Back to Top 100			
#	Name	Symbol	Market Cap	Price	Circulating Supply	Volume (24h)	% 1h	% 24h	% 7d
1	Bitcoin	BTC	\$165,077,711,090	\$9,289.95	17,769,500	\$15,963,551,584	0.00%	1.31%	14.42%
2	Ethereum	ETH	\$28,646,835,680	\$268.82	106,564,409	\$5,698,132,180	-0.15%	-0.03%	3.81%
3	XRP	XRP	\$18,354,267,950	\$0.431845	42,501,950,124 *	\$1,163,581,342	-0.30%	-0.05%	6.99%
4	Litecoin	LTC	\$8,472,996,429	\$135.99	62,305,175	\$4,375,058,549	-0.16%	-0.04%	0.72%
5	Bitcoin Cash	BCH	\$7,387,059,935	\$413.89	17,847,675	\$1,268,834,487	-0.24%	-0.45%	3.50%
6	EOS	EOS	\$6,297,199,427	\$6.85	919,863,767 *	\$1,610,938,429	-0.16%	-0.47%	4.98%
7	Binance Coin	BNB	\$4,892,282,345	\$34.65	141,175,490 *	\$493,217,863	0.10%	-1.86%	-1.74%
8	Bitcoin SV	BSV	\$4,005,430,599	\$224.45	17,845,598	\$279,365,343	-0.06%	-0.08%	11.17%
9	Tether	USDT	\$3,537,593,389	\$1.00	3,534,666,959 *	\$15,173,019,906	0.02%	-0.05%	-0.34%
10	Stellar	XLM	\$2,384,033,130	\$0.122832	19,408,944,303 *	\$272,458,616	-0.25%	-2.34%	-2.58%
2241	Hilux	HLX	\$?	\$0.014386	?	\$?	0.00%	0.00%	12.76%
2242	Stellar Gold	XLMG	\$?	\$0.008425	? *	\$?	0.00%	0.00%	19.61%
2243	Gratz	GRAT	\$?	\$0.003233	? *	\$?	0.00%	0.00%	-99.94%
2244	TRUNK COIN	TRO	\$?	\$0.005912	? *	\$?	0.00%	0.00%	-29.27%

# coinmarketcap.com (July 2020)

Cryptocurrencies ▾		Exchanges ▾		Watchlist		Filters	USD ▾	Next 100 →	View All
Rank	Name	Market Cap	Price	Volume (24h)	Circulating Supply	Change (24h)	Price Graph (7d)		
1	Bitcoin	\$170,068,491,885	\$9,227.43	\$14,035,852,406	18,430,756 BTC	0.12%			
Rank	Name	Market Cap	Price	Volume (24h)	Circulating Supply	Change (24h)			
3	Tether	\$9,206,016,854	\$1.00	\$17,808,525,614	9,187,991,663 USDT *	0.21%			
4	XRP	\$8,817,362,562	\$0.199227	\$866,195,996	44,257,803,618 XRP *	-0.43%			
5	Bitcoin Cash	\$4,327,471,022	\$234.42	\$852,186,905	18,459,969 BCH	-0.72%			
6	Bitcoin SV	\$3,396,585,080	\$184.01	\$991,965,457	18,458,471 BSV	1.34%			
7	Cardano	\$3,161,400,239	\$0.121934	\$348,226,830	25,927,070,538 ADA	1.28%			
8	Litecoin	\$2,874,001,045	\$44.21	\$1,599,512,910	65,013,454 LTC	0.06%			
9	Binance Coin	\$2,822,931,174	\$18.15	\$256,706,012	155,536,713 BNB *	4.64%			
10	Crypto.com Coin	\$2,620,783,989	\$0.144671	\$59,208,669	18,115,525,114 CRO *	0.83%			
2710	Egas	\$?	\$0.000437	\$?	? EGAS *	11.07%			
2711	BTCUP	\$?	\$8.53	\$?	? BTCUP *	0.00%			
2712	BTCDOWN	\$?	\$11.07	\$?	? BTCDOWN *	0.00%			
2713	Vether	\$?	\$1.94	\$?	? VETH *	0.00%			

\* Not Mineable

← Previous 100 | View All

# coinmarketcap.com (Oct 5, 2020)

		Watchlist	Cryptocurrencies	Derivatives	DeFi	Storage	Yield Farming	Show rows 100	Filters
#	Name		Price	24h	7d	Market Cap	Volume	Circulating Supply	
1	Bitcoin BTC		\$10,668.33	▲ 1.04%	▼ 2.08%	\$197,450,773,581	\$56,564,904,549 5,302,135 BTC	18,508,131 BTC	
2	Ethereum ETH		\$352.33	▲ 1.6%	▼ 1.7%	\$39,769,462,837	\$11,159,572,628 31,673,251 ETH	112,874,230 ETH	
# ▾	Name		Price	24h	7d	Market Cap	Volume	Circulating Supply	
3							31,429,204,013 USDT		
4	XRP XRP		\$0.249611	▲ 7.21%	▲ 1.87%	\$11,273,045,675	\$2,086,205,059 8,357,816,131 XRP	45,162,407,484 XRP	
5	Binance Coin BNB		\$29.06	▲ 2.63%	▲ 9.77%	\$4,195,941,050	\$539,545,905 18,568,890 BNB	144,406,560 BNB	
6	Bitcoin Cash BCH		\$221.14	▲ 1.01%	▼ 3.29%	\$4,099,056,432	\$1,192,014,290 5,390,286 BCH	18,535,925 BCH	
7	Polkadot DOT		\$4.19	▲ 3.58%	▼ 2.97%	\$3,570,655,086	\$301,345,555 71,959,232 DOT	852,647,705 DOT	
8	Chainlink LINK		\$9.47	▲ 2.8%	▼ 11.87%	\$3,313,463,869	\$752,326,032 79,467,929 LINK	350,000,000 LINK	
9	Cardano ADA		\$0.098568	▲ 6.12%	▼ 4.71%	\$3,066,687,311	\$598,725,378 6,074,252,843 ADA	31,112,484,646 ADA	
10	Litecoin LTC		\$46.33	▲ 1.59%	▼ 0.68%	\$3,039,306,365	\$2,178,305,292 47,016,659 LTC	65,600,553 LTC	

- Many altcoins
  - borrow concepts from Bitcoin,
  - often directly forking its code base
  - otherwise adopting some of Bitcoin's code.
- Some make only very minor modifications to Bitcoin
  - changing the value of some parameters of the system,
  - continue to incorporate changes made by Bitcoin's developers.

- All altcoins that we know of
  - begin with a new genesis block
  - their own alternate view of transaction history,
  - (rather than forking Bitcoin's block chain after a certain point in history)
- For our purposes, we don't need a precise definition of an altcoin.
- Instead we'll loosely refer to any cryptocurrency launched since Bitcoin as an altcoin.

- Non-altcoin systems like Ripple and Stellar:
  - these are distributed consensus protocols
- They achieve consensus in a model where
  - nodes have identifiers
  - need to be aware of each other
- Bitcoin is not like that
- Despite other similarities, not Altcoins

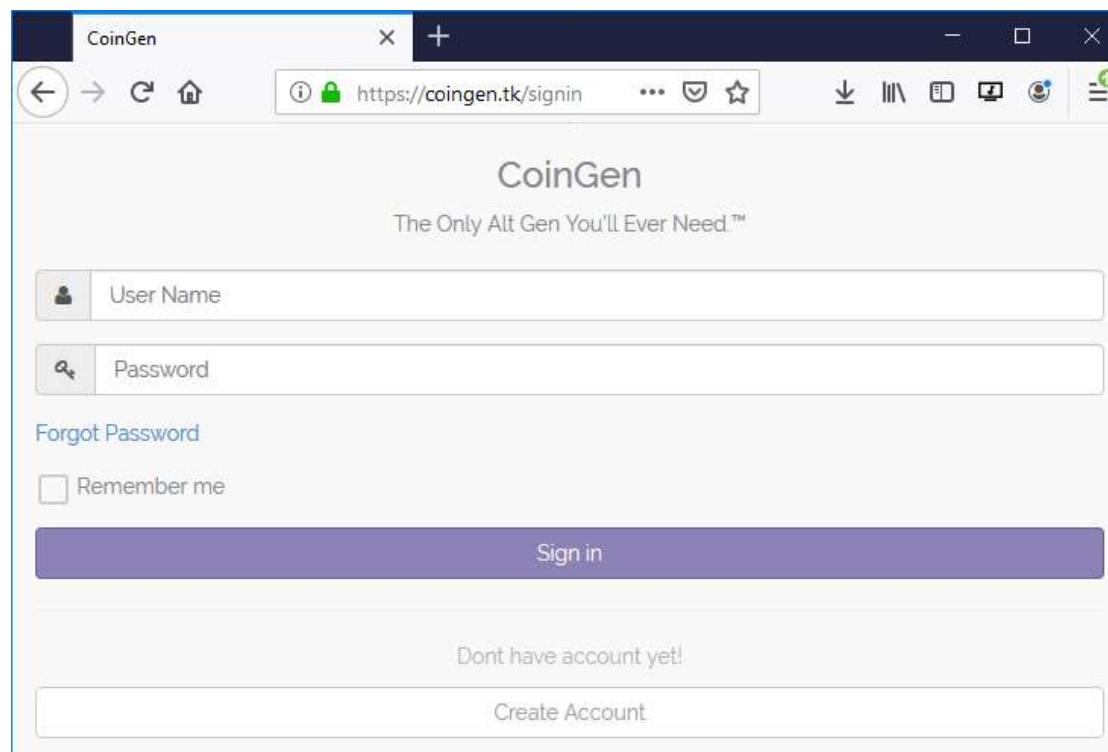
# Why launch an altcoin?

- Some kind of story to tell
- Reason to exist:
  - Claim some characteristic that distinguishes it
- Simply change some built-in parameters of Bitcoin.
  - average time between blocks,
  - block size limit,
  - schedule of rewards being created,
  - inflation rate of the altcoin.
- Or give the community special roles

# How to launch an altcoin

- Forking existing code base of some existing
- **Easy part:** Add/modify technical features
- Coingen, for a small fee:
  - specify various parameters
  - the proof-of-work algorithm
  - name, currency code, logo.
  - Click to fork&download Bitcoin

- Meet Coingen, the tool that brings out the Satoshi Nakamoto in you

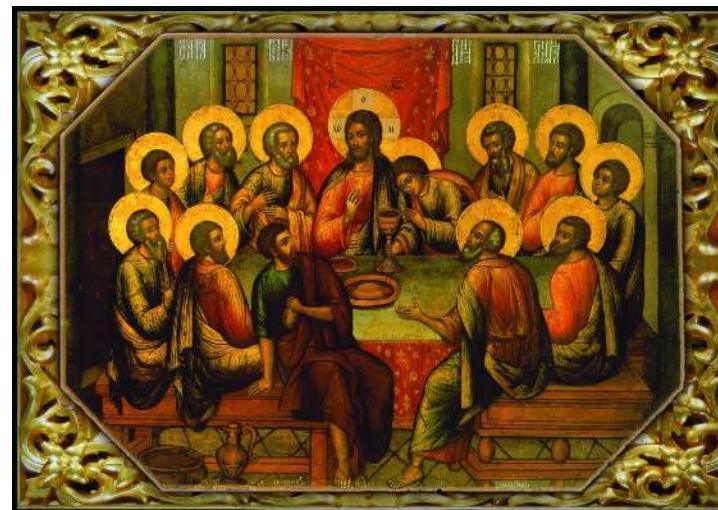
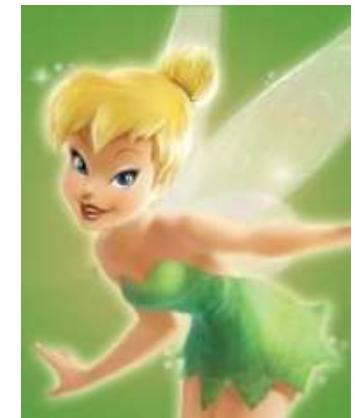


- Hard part
- Bootstrapping!
  - Nobody uses your coins
  - Nobody wants ‘cos no value
  - No security ‘cos no miners
- Need to attract all types stakeholders coin
  - developers, miners, investors,
  - merchants, customers, payment services

- Similar to any other platform
- e.g. a new smartphone operating system, attract
  - users,
  - device manufacturers,
  - app developers
  - various others
- Each of these groups needs the others
- But usually just economic, not security issues
  - No hash power → double spends, forkings..

- No simple recipe for bootstrapping adoption,
- But in general **miners** will come once they believe
  - coinbase rewards they receive
  - will be worth the effort.
- To encourage, many give early miners great rewards.
- Bitcoin, pioneered this approach
- Some altcoins more aggressive

- Basic trick
  - Make people believe it is/will be valuable
  - Need a good story
  - Believers will make others believe



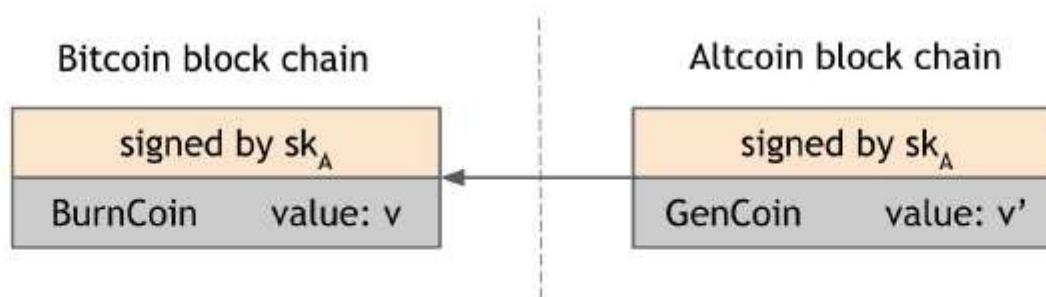
# Pump-and-dump scams

- If creator succeed in bootstrapping → Richie Rich \$
- Attractive for everyone, including scams
  - buy up shares of some obscure altcoin,
  - then convince the public of this coin's supposed undiscovered potential (“pump it”).
  - If succeed unload their shares and reap a profit (“dump it”)
  - They left with \$, many left with no-value coins
- Common in stock market
- Many altcoins:
  - Not real innovation
  - Just “me-too”

# Initial allocation

- In Bitcoin, (*new*) coins only through mining
- Some altcoins, for some reasons, initial alloc.
  - “pre-mine”
  - Give incentives to developers to create&bootstrap
  - Allocate to a diverse community
  - Clever way: Give your coins to Bitcoin owners ☺
  - Use proof-of-burn: Gain some altcoin by destroying yours bitcoins

- Here, proof-of-burn, also called
  - One way peg
  - Price ceiling
- Associate 1 altcoin = 1 bitcoin
  - Is actually  $1 \text{ altcoin} \leq 1 \text{ bitcoin}$



**Figure 10.2: Allocating altcoins via proof-of-burn.** The altcoin supports a GenCoin transaction that takes a *Bitcoin* transaction as input. GenCoin is signed by the same private key that signed the proof-of-burn (and using the same signature scheme). This ensures that the same user who burned bitcoins also created the GenCoin. If the peg ratio is 1:1, then  $v'$  must be no greater than  $v$ .

## Other ways to increase the diversity of an altcoin

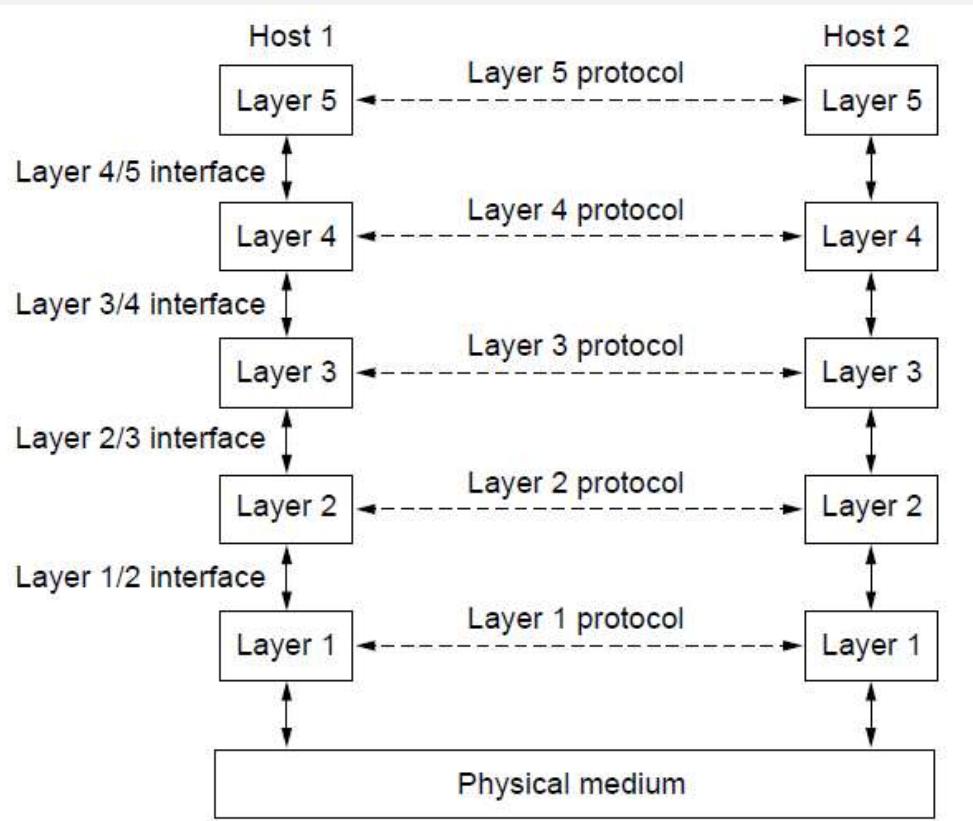
- Tipping
  - Make others hear about your altcoin
  - Require to login (via socialmedia?)
  - Install wallet software etc
- Faucet
  - Give anyone who visits your website/enter e-mail address

*Before starting NameCoin:*

*A quick review of DNS in slides with gray background*

*From “Introduction to Computer Networking” course*

# The Application Layer



Uses transport services to build distributed applications

Application
Transport
Network
Link
Physical

# DNS – Domain Name System

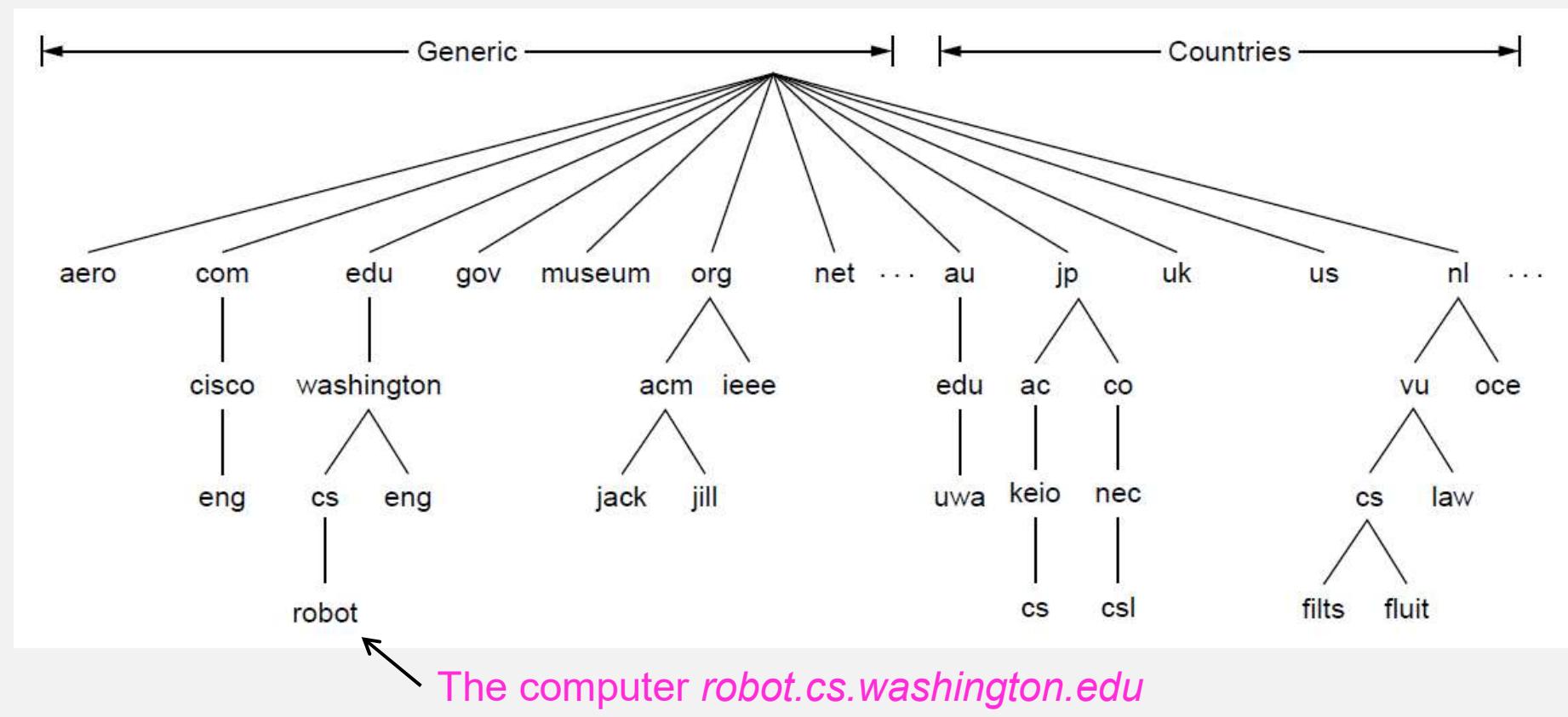
The DNS resolves high-level human readable names for computers to low-level IP addresses

- DNS name space »
- Domain Resource records »
- Name servers »

# The DNS Name Space (1)

DNS namespace is hierarchical from the root down

- Different parts delegated to different organizations



The computer *robot.cs.washington.edu*

# The DNS Name Space (2)

Generic top-level domains are controlled by ICANN who appoints registrars to run them

This one was controversial

Domain	Intended use	Start date	Restricted?
com	Commercial	1985	No
edu	Educational institutions	1985	Yes
gov	Government	1985	Yes
int	International organizations	1988	Yes
mil	Military	1985	Yes
net	Network providers	1985	No
org	Non-profit organizations	1985	No
aero	Air transport	2001	Yes
biz	Businesses	2001	No
coop	Cooperatives	2001	Yes
info	Informational	2002	No
museum	Museums	2002	Yes
name	People	2002	No
pro	Professionals	2002	Yes
cat	Catalan	2005	Yes
jobs	Employment	2005	Yes
mobi	Mobile devices	2005	Yes
tel	Contact details	2005	Yes
travel	Travel industry	2005	Yes
xxx	Sex industry	2010	No

# Domain Resource Records (1)

The key resource records in the namespace are IP addresses (A/AAAA) and name servers (NS), but there are others too (e.g., MX)

Type	Meaning	Value
SOA	Start of authority	Parameters for this zone
A	IPv4 address of a host	32-Bit integer
AAAA	IPv6 address of a host	128-Bit integer
MX	Mail exchange	Priority, domain willing to accept email
NS	Name server	Name of a server for this domain
CNAME	Canonical name	Domain name
PTR	Pointer	Alias for an IP address
SPF	Sender policy framework	Text encoding of mail sending policy
SRV	Service	Host that provides it
TXT	Text	Descriptive ASCII text

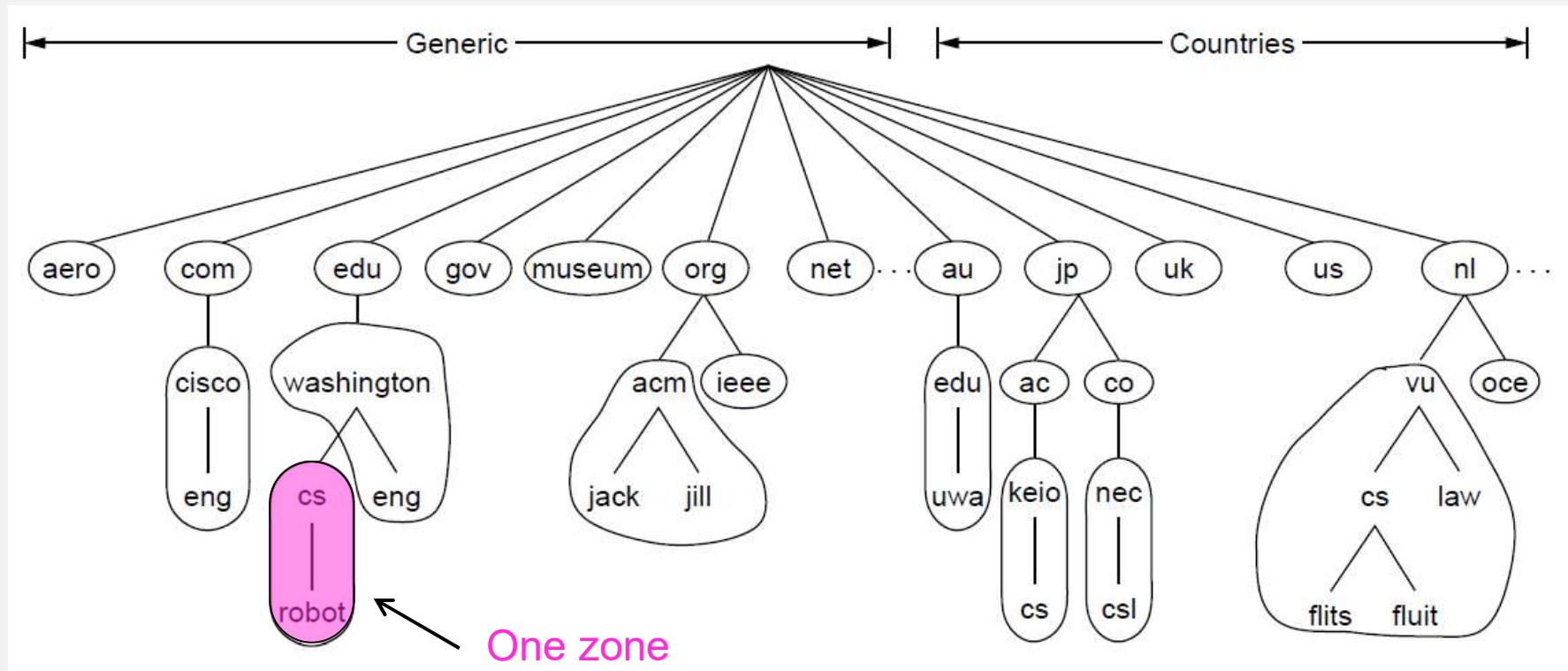
# Domain Resource Records (2)

; Authoritative data for cs.vu.nl				
cs.vu.nl.	86400	IN	SOA	star boss (9527,7200,7200,241920,86400)
cs.vu.nl.	86400	IN	MX	1 zephyr
cs.vu.nl.	86400	IN	MX	2 top
cs.vu.nl.	86400	IN	NS	star
				← Name server
star	86400	IN	A	130.37.56.205
zephyr	86400	IN	A	130.37.20.10
top	86400	IN	A	130.37.20.11
www	86400	IN	CNAME	star.cs.vu.nl
ftp	86400	IN	CNAME	zephyr.cs.vu.nl
flits	86400	IN	A	130.37.16.112
flits	86400	IN	A	192.31.231.165
flits	86400	IN	MX	1 flits
flits	86400	IN	MX	2 zephyr
flits	86400	IN	MX	3 top
rowboat		IN	A	130.37.56.201
		IN	MX	1 rowboat
		IN	MX	2 zephyr
little-sister		IN	A	130.37.62.23
laserjet		IN	A	192.31.231.216

A portion of a possible DNS database for cs.vu.nl.

# Name Servers (1)

Name servers contain data for portions of the name space called zones (circled).



# Name Servers (2)

Finding the IP address for a given hostname is called resolution and is done with the DNS protocol.

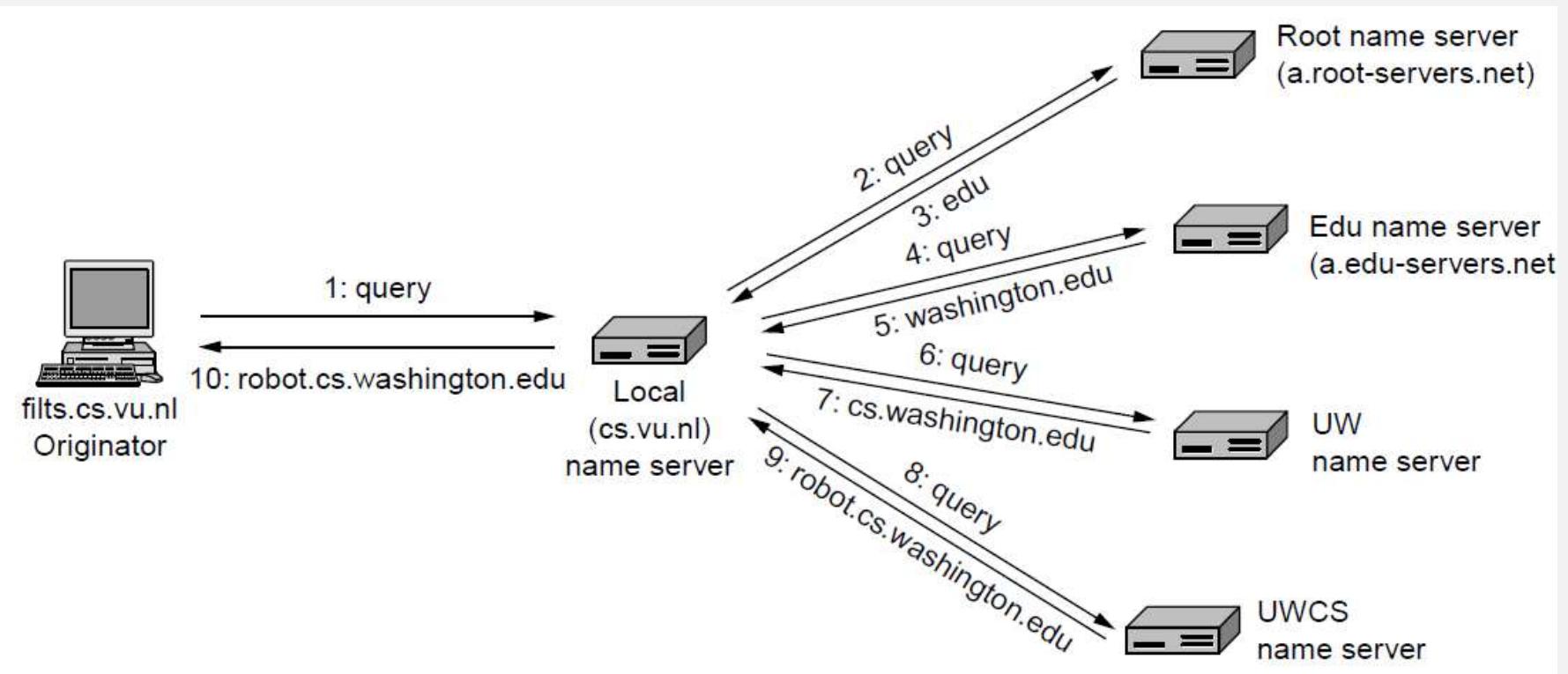
Resolution:

- Computer requests local name server to resolve
- Local name server asks the root name server
- Root returns the name server for a lower zone
- Continue down zones until name server can answer

DNS protocol:

- Runs on UDP port 53, retransmits lost messages
- Caches name server answers for better performance

# Name Servers (3)



Example of a computer looking up the IP for a name

# Namecoin '2011

- Bitcoin:
  - Secure global network
  - Tamper-proof (append-only-log)
  - Can't we use it for another app
  - e.g. Naming system? Decentralizing DNS

# Namecoin: Rules

- View data entries as name/value pairs
  - names being globally unique
  - everyone allowed to look up the value mapped to a name
  - If name/value pair has the same name as a previous database entry → view it as an update
- Only the initial creator of the entry can update
  - associate each name with a Bitcoin address
  - update transactions to be signed by the  $p_k$  for that address.

## *Namecoin cont'*

- It can be built on Bitcoin: Overlay currency
- Simpler to do in an altcoin
  - Write these rules as the altcoin's rules
  - So, rather than checking by each user,
  - Enforced by miners
- Global name/value store: Users can
  - define one or more names (fees applied)
  - transfer the control
  - Transfer the domain from A to B & Namecoins from B to A

## *Namecoin cont'*

- DNS: {Name, Value} pair
  - Names: Domains
  - Values: IP's
- Browser plug-in
  - example.bit
  - Searches in namecoin registry, not DNS
- Interesting, technically & historically
- Against centralization of DNS
- Used mostly by “squatters” (buy lots of and then sell)

# Litecoin '2011

- Most popular altcoin for years
- Most forked codebase
  - Even more than Bitcoin
- Basic distinction between Bitcoin
  - Memory-hard mining puzzle
- When Litecoin launched, Bitcoin was in GPU era
  - Was able to mine Litecoin with CPU (so, GPU-resistant)
  - Eventually GPU → FPGA → ASIC
  - Transition took longer than Bitcoin, why?
  - Just the hard puzzle, or the exchange rate?

## *Litecoin cont'*

- So, it failed
  - But the idea for a “more” decentralized CC
  - Worked for bootstrapping!
- 
- Only a few small differences from Bitcoin
  - Mining a block in each 2.5 mins
  - Followed & adapted Bitcoin’s modifications

# Proof-of-Stake and Virtual Mining



- Why not simply allocate mining “power” directly to all currency holders in proportion to how much currency they actually hold?



# Advantages

- Remove wasteful right half → Environment
- No ASIC, No AR → Centralization
- CC's value → Miners tend to behave good

# Implementing Virtual Mining

- Not
  - researched scientifically
  - analyzed practically
  - (Bitcoin is too dominant)
- Peercoin (2012)
  - Hybrid: proof-of-work & proof-of-stake
  - Coin-age, coin-stake: solving & adjusting difficulty

# Peercoin '2012, vs. Oct 2020

341	 Peercoin	PPC	\$10,182,880	\$0.400338	25,435,700	\$256,700	-0.59%	-3.66%	0.44%	...
588	 Peercoin PPC		\$0.235893	▲ 10.45%	▼ 8.97%	\$6,257,543		\$30,099.55	127,598 PPC	

- First proof-of-stake altcoin
- One more interesting aspect
  - Admins kept a trusted public key as a safeguard
  - Destroyed decentralization!
  - It can be removed
  - But it is there! So, proof-of-stake did not work
  - Will it work if safeguard removed?

# Dogecoin '2013



- CTRL-F “Doge”: 8 matches

30 Dogecoin DOGE \$377,797,911 \$0.003148 120,003,321,307 \$33,868,429 -0.27% 0.57% 3.12% ...

- Close fork of Litecoin → Technically not interesting
- Interesting community values
  - tipping,
  - generosity,
  - not taking cryptocurrency so seriously

# Dogecoin '2013

- Named after Doge,
  - an Internet meme
  - featuring a grammatically-challenged Shiba Inu dog
- several interesting & successful marketing campaigns
  - Sponsored a NASCAR driver
  - raised over \$30,000 to support the Jamaica National Bobsled Team in 2014 Winter Olympics
- Bootstrapping could be successful with a non-technical narrative



# Relationship Between Bitcoin and Altcoins

- Market capitalization:
  - Traditionally: price of a share \* total number of shares
  - Altcoins: price of a coin \* total number of coins
  - Bitcoin: %90 of the market!
  - Not the most important thing
  - Interpretation difficult: Exchange rate, future, new coins
  - Some coins may be lost → not in circulation

# Relationship Between Bitcoin and Altcoins

- Mining Power:
  - If two altcoins using the same puzzle
  - Compare the hash rate
  - Zetacoin, same puzzle with Bitcoin
  - Hash rate was 1/100K of Bitcoin
  - Not only the current hash rate but its change by time
  - This way, we can compare coins with different puzzles

# Relationship Between Bitcoin and Altcoins

- Other indicators:
  - Change in the exchange rate by time
  - In the long run, it is related to change in hash rate
  - Exchange volume with third parties
  - Not the transaction volume; why?
  - Number and size of the merchants and payment systems

# Relationship Between Bitcoin and Altcoins

- “Network effect”
  - If there are two similar options, only one will win
  - May, or may not be technically superior
  - e.g. Blue-ray vs. HD-DVD
  - Bitcoin dominates (despite technically superiors)
  - But not as simple as disc formats!
  - Low Switching Cost
  - Various features, some even complementary → supports
  - Again over simplification. What is the value, risks, updates

# Merge Mining

- Bitcoin's hash power of miners so big
  - Even more than all the power of altcoins
  - Altcoin Infanticide
  - They can attack?
  - Why? No gain?
- 
- 2012, Eligius (bitcoin mining pool) decided that CoiledCoin is a scam.
  - They attacked and destroyed it

# Merge Mining

- If an altcoin just forks from Bitcoin (no change)
- Mining power can be allocated to a single coin
- Bootstrapping problem
- Merging the mining?
- Design the altcoin
- Place some info to Bitcoin blocks (technical)
- Drawback: too slow. Solution?

# Adjust your altcoin's mining target

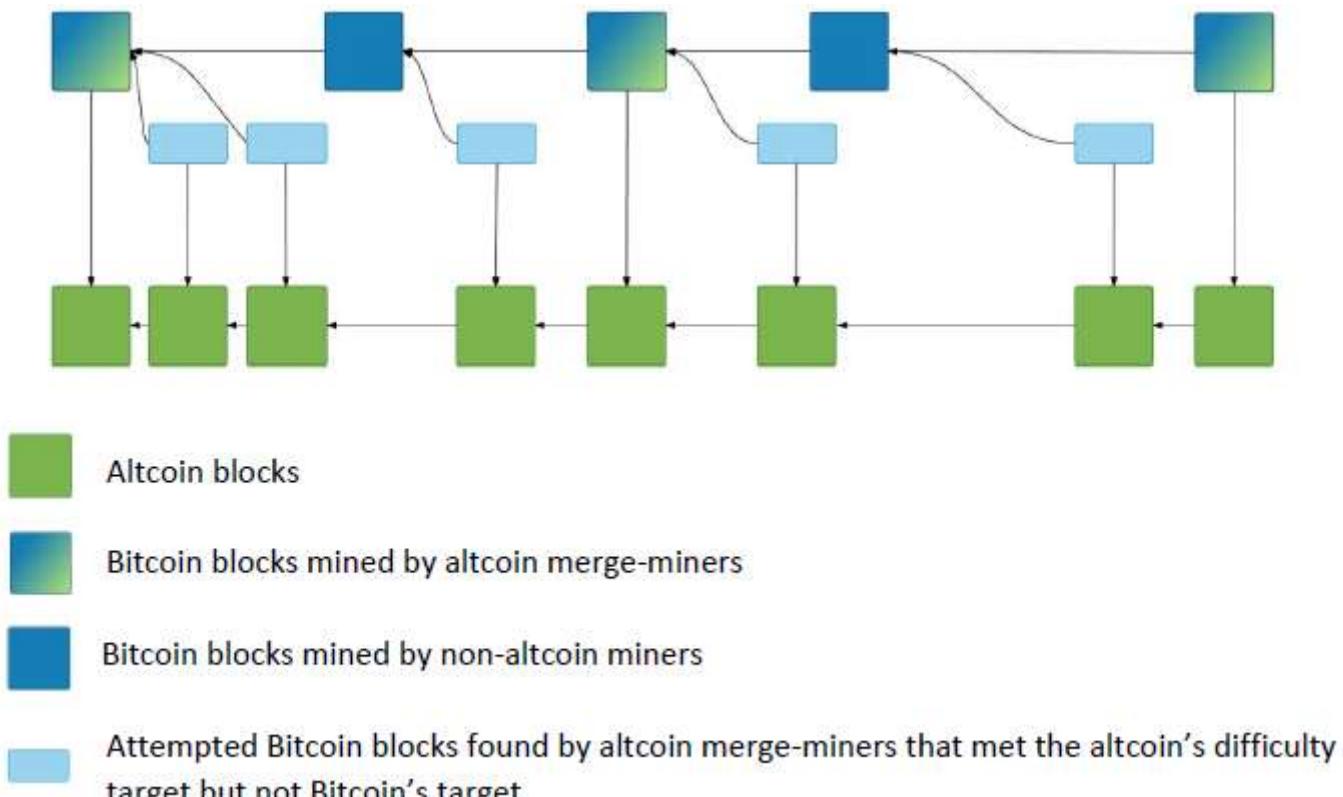


Figure 10.5: merge mining.

- Merge mining & Security
- Altcoin got some mining power of Bitcoin
- Huge investment required to attack altcoin
- Not really
- Eligius was a merge-mining!
- Complicated,
- Not clear whether merging is a good idea!

# Bitcoin Scripts

- Script Language built specifically for Bitcoin, called “Script”
  - Stack-based
  - Simple & Compact
  - Native support for cryptographic operations
    - Compute Hash functions
    - Compute signatures
    - Verify signatures

# Script

- Stack-based
  - Each instruction executed once
  - No loops
  - Number of instructions
    - How long it can take
    - How much memory it can use
  - Not Turing-complete, by design!
    - Arbitrary miners can't submit infinite-loops

```
for i in range(0,10,2):
    print i
```

```
0
2
4
6
8
```

- Only two possible outcomes:
  - Execute normally with no errors → Valid,
  - Error → Invalid transaction
- Each instruction represented by 1 Byte
  - 256 in total ( $2^8=256$ )
  - 15 disabled, 75 reserved (to be added later)

# Ethereum & Smart Contracts

- Bitcoin: “Turing-incomplete”
- More apps
- Instead of a new altcoin for each app
- Why not make a Turing-complete coin?
- Ethereum: Provides TC programming language

# Ethereum & Applications

- Namecoin

```
contract NameRegistry {
    mapping(bytes32 => address) public registryTable;
    function claimName(bytes32 name) {
        if (msg.value < 10) {
            throw;
        }
        if (registryTable[name] == 0) {
            registryTable[name] = msg.sender;
        }
    }
}
```

**Figure 10.8:** A simple Ethereum smart contract implementing a name registry.

# Ethereum

- A contract: A program that lives on the blockchain
- Anybody can create an Ethereum contract, for a small fee,
  - by uploading its program code in a special transaction.
- This contract is
  - written in bytecode
  - executed by a special Ethereum-specific virtual machine: EVM
- Once uploaded, the contract will live on the blockchain.
  - It has its own balance of funds,
  - Other users can make procedure calls through API
  - The contract can send and receive money.

# Ethereum

- Turing Completeness allows loops → ∞ loops!
- Contracts may run forever!
- A way to check whether a program runs forever?
  - No! (the undecidability of the Halting Problem)
- How to prevent:
  - Define “GAS”
  - Each VM instruction runs for some fee (gas)

- Basic operations (addition, comparison): 1 gas,
- Computing a SHA-3 hash: 20 gas
- Writing a 256-bit word to storage: 100 gas
- Transaction: 21,000 gas
- These prices are fixed.
  - Modifying → Forking
- Like a cheap airline: You pay for everything ;)

- Gas
  - paid to execute contracts
  - paid to use “ether”, the currency
- Every transaction can specify how much gas
- Miners are free to choose any transaction
  - A free market
- Every call specifies how much gas to spend
  - When run out of gas, execution halts

- Each computation instruction costs some gas
- Not suitable for large computations
  - Use clouds instead; Amazon's Computing Cloud
- Suitable to implement security logic
  - Allows many parties to count on to behave
- Its security too complex, not as Bitcoin
  - Difficult to analyze
- Rewarding complex, not clear
  - It goes to who includes the transaction, not to miners

# Chess in Ethereum

- Alice and Bob live in different countries
- Neither trusts each other to pay if they lose
- Ethereum can solve this!
- Alice
  - Writes a chess program, uploads to Ethereum
  - Bets some money (sends a contract with some ether)
- Bob
  - Should be sure that contract works
  - Bets money

# Chess in Ethereum

- Once both bet, game starts
  - Noone can take the money without winning
- Playing
  - Send transactions to the contract: Moves
- Contract must
  - ensure, play only in order,
  - check the rules for each move
  - End the game. Win, draw; send the ether

# Chess in Ethereum

- What if the loser stops playing?
  - Contract gives the ether to other after some time
- Who moves first?
  - White more advantageous.
  - No randomness source
  - Randomness beacons?
  - Use some Hash? (Convince only A&B, not whole world)

# Other Applications

- Chess is fun, but real life:
  - Smartcontracts
  - Escrowed payments
  - Micropayments
  - Mixing services
  - Auctions
- Ledger
  - Bitcoin: Transaction-based
  - Ethereum: Account-based