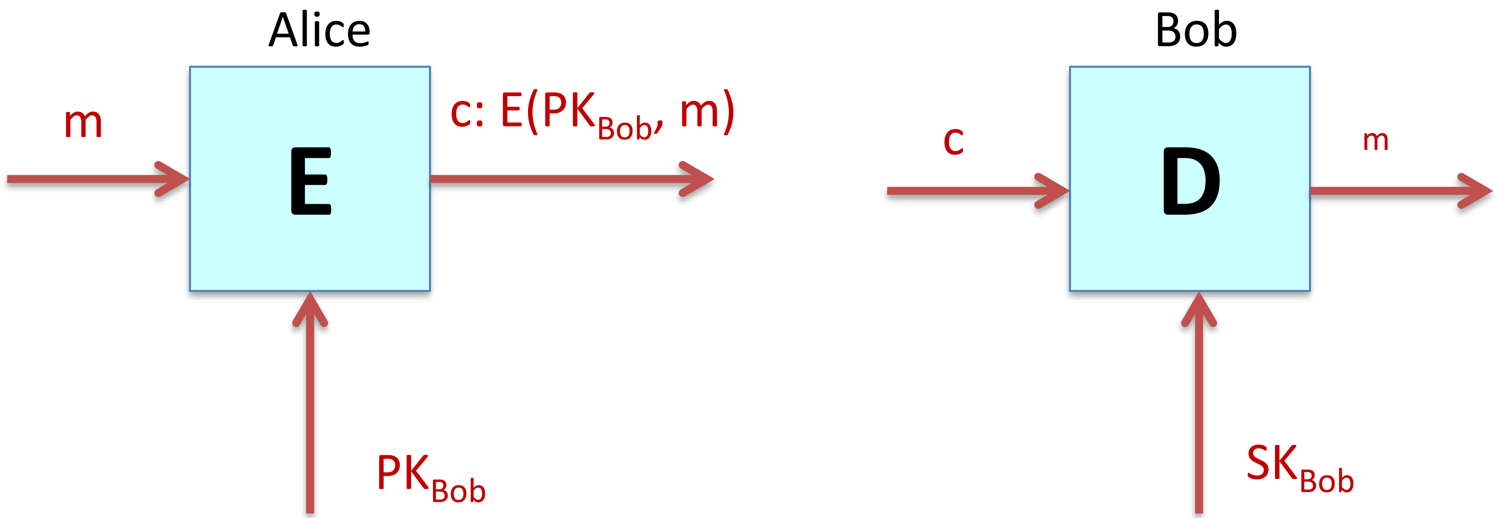
Public Key Encryption



PK: Public Key SK: Secret Key

Public Key Encryption

Def: a public-key encryption system is a triple of algs. (G, E, D)

G(): randomized alg. outputs a key pair (pk, sk)

E(pk, m): randomized alg. that takes $m \in M$ and outputs $c \in C$

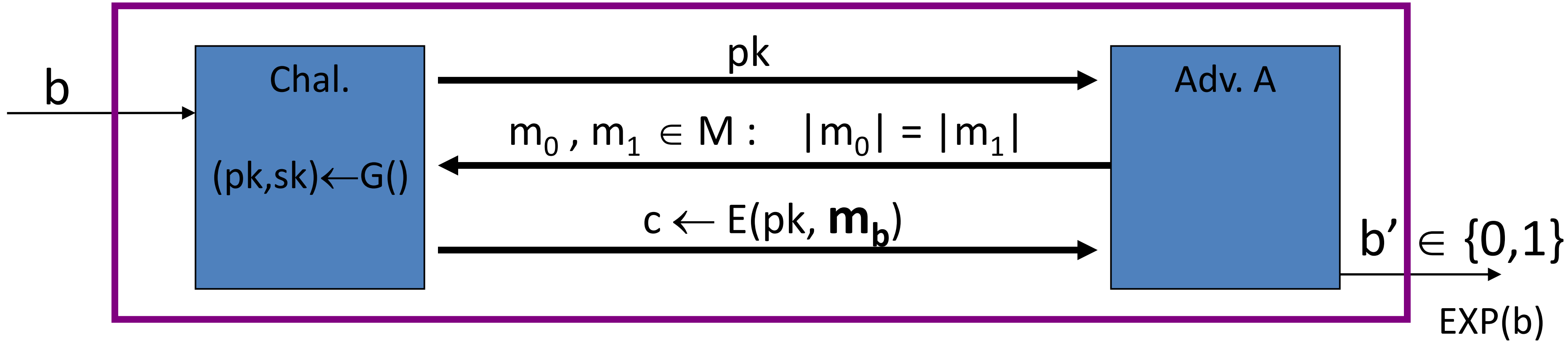
D(sk,c): det. alg. that takes $c \in C$ and outputs $m \in M$ or $\perp$

Consistency: $\forall$(pk, sk) output by G :

$$\forall m \in M: \quad D(sk, E(pk, m)) = m$$

Semantic Security

For b=0,1 define experiments EXP(0) and EXP(1) as:



Def: $\mathbb{E}$ =(G,E,D) is sem. secure (a.k.a IND-CPA) if for all efficient A:

$$Adv_{SS}[A,\mathbb{E}] = \left| \Pr[EXP(0)=1] - \Pr[EXP(1)=1] \right| < \text{negligible}$$

Establishing a shared secret

Alice **Bob**

$(pk, sk) \longleftarrow G()$

"Alice", pk $\longrightarrow$

choose random
$x \in \{0,1\}^{128}$

"Bob", $c \longleftarrow E(pk, x)$ $\longleftarrow$

$D(sk, c) \rightarrow x$

x: shared secret

Security (eavesdropping)

Adversary sees **pk, E(pk, x)** and wants $x \in M$

Semantic security $\Rightarrow$

adversary cannot distinguish

$\{ pk,\ E(pk, x),\ x \}$ from $\{ pk,\ E(pk, x),\ rand \in M \}$

$\Rightarrow$ can derive session key from x.

Note: protocol is vulnerable to man-in-the-middle

Insecure against man in the middle

As described, the protocol is insecure against **active** attacks

Alice | **MiTM** | **Bob**

$(pk, sk) \longleftarrow G()$ | $(pk', sk') \longleftarrow G()$

"Alice", pk $\longrightarrow$ "Alice", pk' $\longrightarrow$

choose random $x \in \{0,1\}^{128}$

"Bob", $E(pk, x)$ $\longleftarrow$ "Bob", $E(pk', x)$ $\longleftarrow$

x

x

Active attacks: symmetric vs. public-key

Recall: secure symmetric cipher provides **authenticated encryption**

[chosen plaintext security & ciphertext integrity]

Roughly speaking: **attacker cannot create new ciphertexts**

Implies security against chosen ciphertext attacks

In public-key settings:

Attacker **can** create new ciphertexts using pk !!

So instead: we directly require chosen ciphertext security

Public Key Encryption

Constructions generally rely on hard problems from number theory and algebra

Lecture 4.6: Constructions

Trapdoor functions (TDF)

Def: a trapdoor func. $X \longrightarrow Y$ is a triple of efficient algs. (G, F, F^{-1})

$G()$: randomized alg. outputs a key pair (pk, sk)

$F(pk,\cdot)$: det. alg. that defines a function $X \longrightarrow Y$

$F^{-1}(sk,\cdot)$: defines a function $Y \longrightarrow X$ that inverts $F(pk,\cdot)$

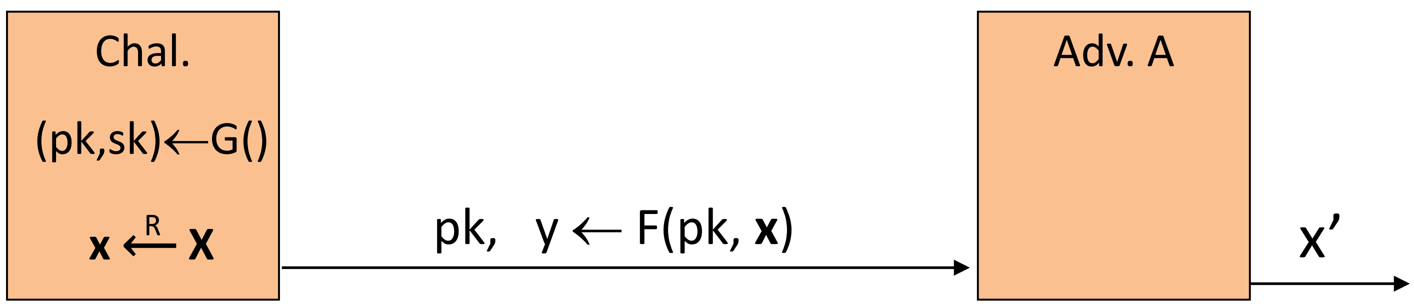
More precisely: $\forall (pk, sk)$ output by G

$$\forall x \in X: \quad F^{-1}(sk, F(pk, x)) = x$$

Secure Trapdoor functions (TDF)

(G, F, F^{-1}) is secure if $F(pk, \cdot)$ is a "one-way" function:

can be evaluated, but cannot be inverted without sk



<u>Def</u>: (G, F, F^{-1}) is a secure TDF if for all efficient A:

$$\mathrm{Adv}_{OW}[A,F] = \Pr[\, x = x' \,] < \text{negligible}$$

Public-key encryption form TDFs

(G, F, F^{-1}): secure TDF $X \longrightarrow Y$

(E_s, D_s) : symmetric auth. encryption defined over (K,M,C)

$H: X \longrightarrow K$ a hash function

We construct a pub-key enc. system (G, E, D):

Key generation G: same as G for TDF

Public-Key encryption from TDFs

- (G, F, F^{-1}): secure TDF $X \longrightarrow Y$
- (E_s, D_s) : symmetric auth. encryption defined over (K,M,C)
- $H: X \longrightarrow K$ a hash function

E(pk, m) :

$x \xleftarrow{R} X, \quad y \longleftarrow F(pk, x)$

$k \longleftarrow H(x), \quad c \longleftarrow E_s(k, m)$

output (y, c)

D(sk, (y,c)) :

$x \longleftarrow F^{-1}(sk, y),$

$k \longleftarrow H(x), \quad m \longleftarrow D_s(k, c)$

output m

In pictures:

$$\underbrace{F(pk,\ x)}_{\text{header}}\ \underbrace{E_s(\ H(x),\ \ m\ )}_{\text{body}}$$

<u>Security Theorem</u>:

If **(G, F, F^{-1})** is a secure TDF, **(E_s, D_s)** provides auth. enc.

and **H:** $X \longrightarrow K$ is a "random oracle"

then **(G,E,D)** is CCA^{ro} secure.

Incorrect use of a Trapdoor Function (TDF)

Never encrypt by applying F directly to plaintext:

E(pk, m) :	**D(sk, c) :**
output $c \longleftarrow F(pk, m)$	output $F^{-1}(sk, c)$

Problems:

Deterministic: cannot be semantically secure !!

Many attacks exist (next segment)

Number Theory

Let $N = p \cdot q$ where p,q are prime

$Z_N = \{0,1,2,\ldots,N\text{-}1\}$; $(Z_N)^*$ = {invertible elements in Z_N}

<u>Facts</u>: $x \in Z_N$ is invertible $\Leftrightarrow$ $\gcd(x,N) = 1$

– Number of elements in $(Z_N)^*$ is $\varphi(N) = (p-1)(q-1) = N-p-q+1$

<u>Euler's thm</u>: $\boxed{\forall\, x \in (Z_N)^* \;:\; x^{\varphi(N)} = 1}$

The RSA trapdoor permutation

First published: Scientific American, Aug. 1977.

Very widely used:

- SSL/TLS: certificates and key-exchange
- Secure e-mail and file systems

... many others

The RSA trapdoor permutation

G(): choose random primes p,q ≈1024 bits. Set **N=pq**.

choose integers **e , d** s.t. $\mathbf{e \cdot d = 1 \pmod{\varphi(N)}}$

output pk = (N, e) , sk = (N, d)

F(pk, x): $\mathbb{Z}_N^* \to \mathbb{Z}_N^*$; $\mathbf{RSA(x) = x^e}$ (in Z_N)

$\mathbf{F^{-1}(sk, y)} = y^d$; $y^d = \mathbf{RSA(x)^d} = x^{ed} = x^{k\varphi(N)+1} = \left(x^{\varphi(N)}\right)^k \cdot \mathbf{x} = x$

The RSA Assumption

RSA assumption: RSA is one-way permutation

For all efficient algs. A:

$$\Pr\left[\ A(N,e,y) = y^{1/e}\ \right] < \text{negligible}$$

where $p,q \xleftarrow{R}$ n-bit primes, $N \leftarrow pq$, $y \xleftarrow{R} Z_N^*$

Review: RSA public-Key Encryption

(E_s, D_s): symmetric enc. scheme providing auth. encryption.

H: $Z_N \to K$ where K is key space of (E_s, D_s)

G(): generate RSA params: pk = (N,e), sk = (N,d)

E(pk, m): (1) choose random x in Z_N

(2) $y \leftarrow RSA(x) = x^e$, $k \leftarrow H(x)$

(3) output $(y, E_s(k,m))$

D(sk, (y, c)): output $D_s(\ H(RSA^{-1}(y))\ ,\ c)$

Textbook RSA is insecure

Textbook RSA encryption:

- public key: **(N,e)** Encrypt: $\mathbf{c \leftarrow m^e}$ (in Z_N)
- secret key: **(N,d)** Decrypt: $\mathbf{c^d \rightarrow m}$

Insecure cryptosystem !!

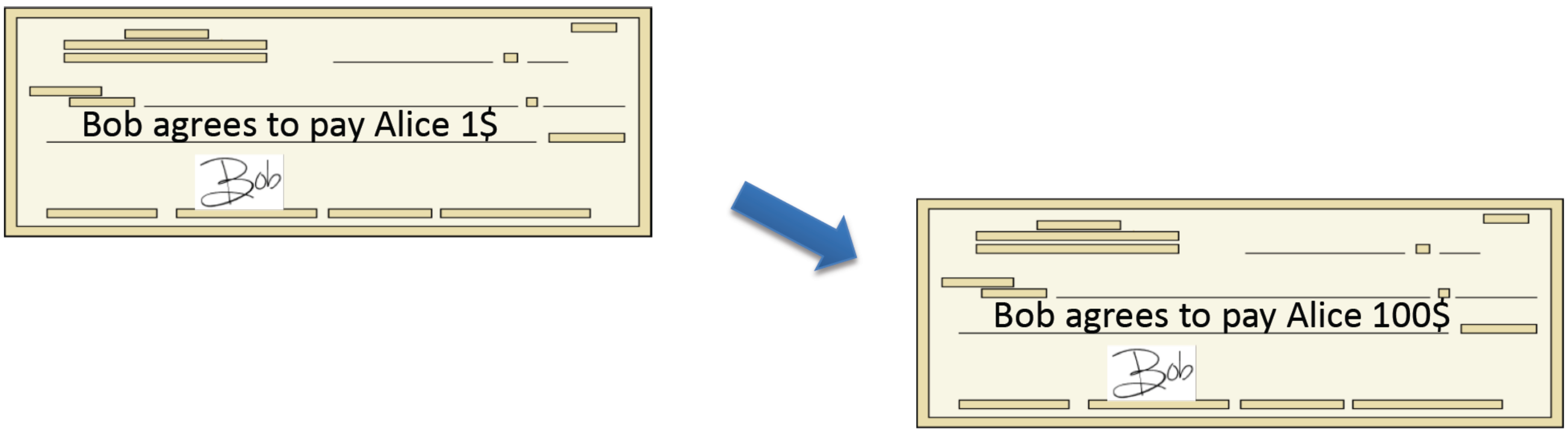
- Is not semantically secure and many attacks exist

$\Rightarrow$ The RSA trapdoor permutation is not an encryption scheme !

Lecture 4.7: Digital Signature

Physical Signature

Goal: bind document to author

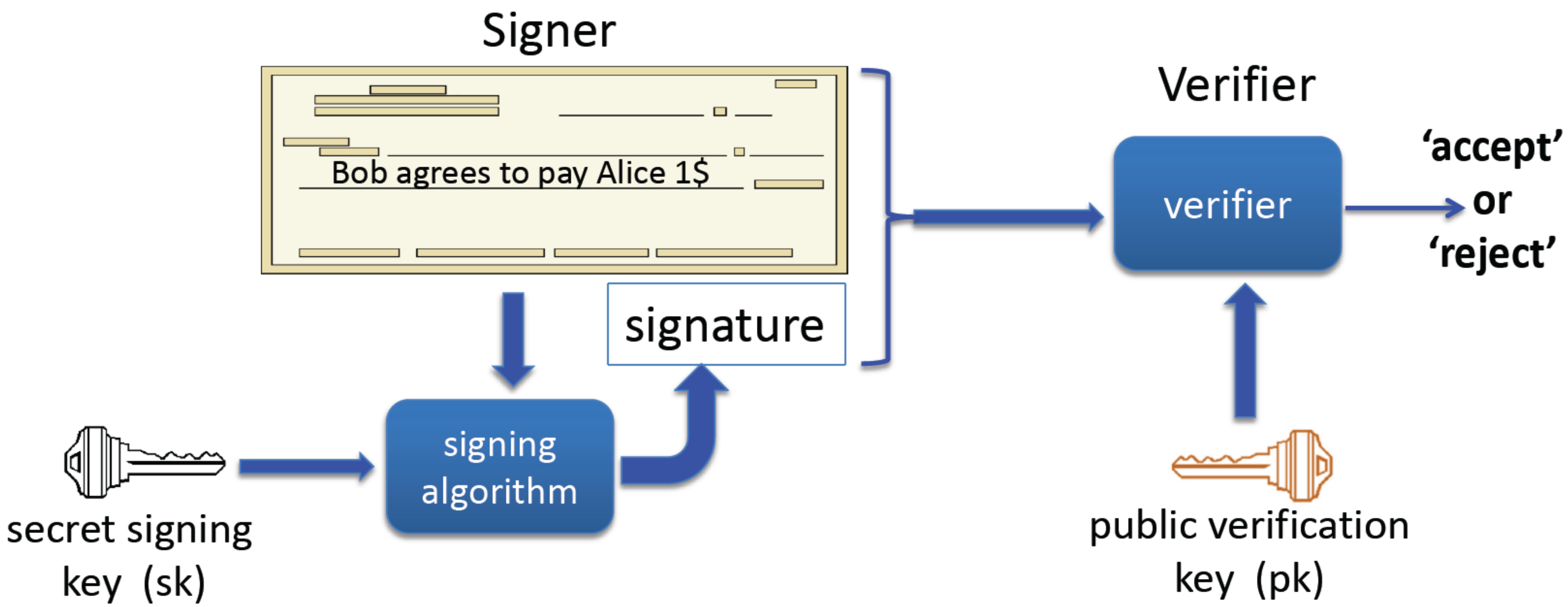


Problem in the digital world:

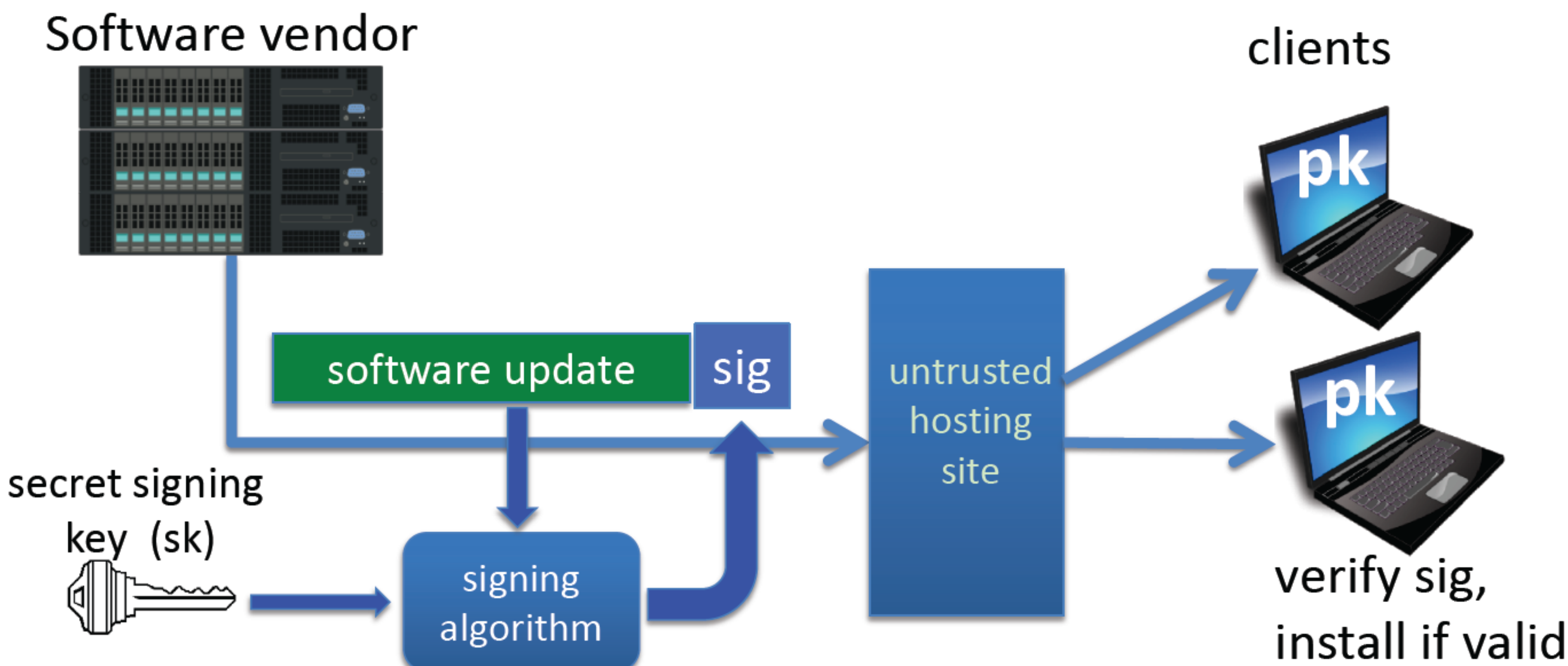
anyone can copy Bob's signature from one doc to another

Digital Signature

Solution: make signature depend on document



A more realistic example



Digital Signature: syntax

<u>Def</u>: a signature scheme (Gen,S,V) is a triple of algorithms:

- Gen(): randomized alg. outputs a key pair (pk, sk)
- S(sk, $m \in M$) outputs sig. σ
- V(pk, m, σ) outputs 'accept' or 'reject'

Consistency: for all (pk, sk) output by Gen :

$$\forall m \in M: \quad V\big(pk,\ m,\ S(sk, m)\big) = \text{'accept'}$$

Digital Signature: security

Attacker's power: **chosen message attack**

- for $m_1, m_2, \ldots, m_q$ attacker is given $\sigma_i \leftarrow S(sk, m_i)$

Attacker's goal: **existential forgery**

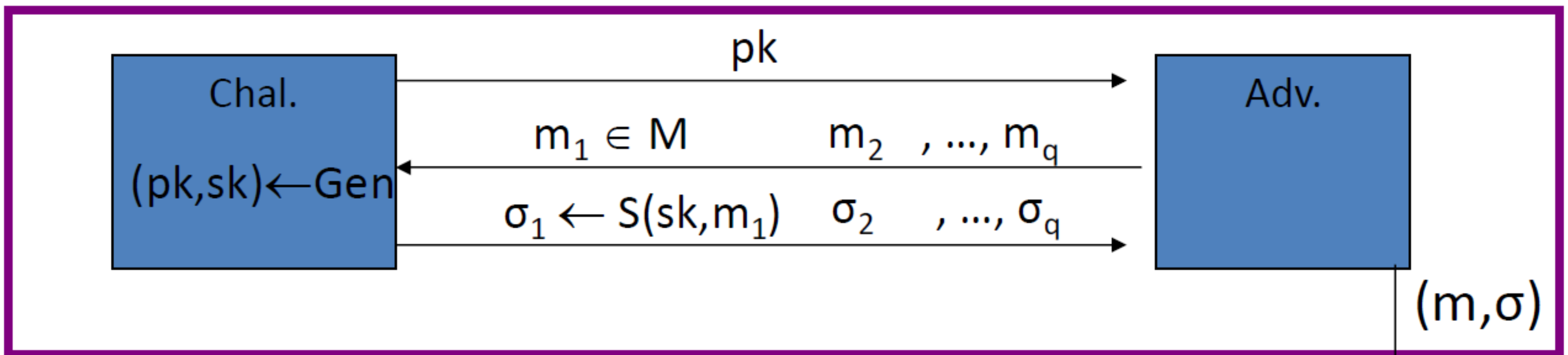
- produce some **new** valid message/sig pair (m, σ).

$$m \notin \{ m_1 , \ldots , m_q \}$$

$\Rightarrow$ attacker cannot produce a valid sig. for a new message

Secure Signature

For a sig. scheme (Gen,S,V) and adv. A define a game as:



Adv. wins if **V(pk,m,σ) = \`accept'** and $\mathbf{m \notin \{m_1, \ldots, m_q\}}$

Def: SS=(Gen,S,V) is **secure** if for all "efficient" A:

$$\mathrm{Adv}_{\mathrm{SIG}}[A,SS] = \Pr[\,A \text{ wins}] \quad \text{is "negligible"}$$

Secure Signature

Let (Gen,S,V) be a signature scheme.

Suppose an attacker is able to find $m_0 \neq m_1$ such that

$$\mathbf{V(pk, m_0, \sigma) = V(pk, m_1, \sigma)} \quad \text{for all } \sigma \text{ and keys } (pk, sk) \leftarrow \text{Gen}$$

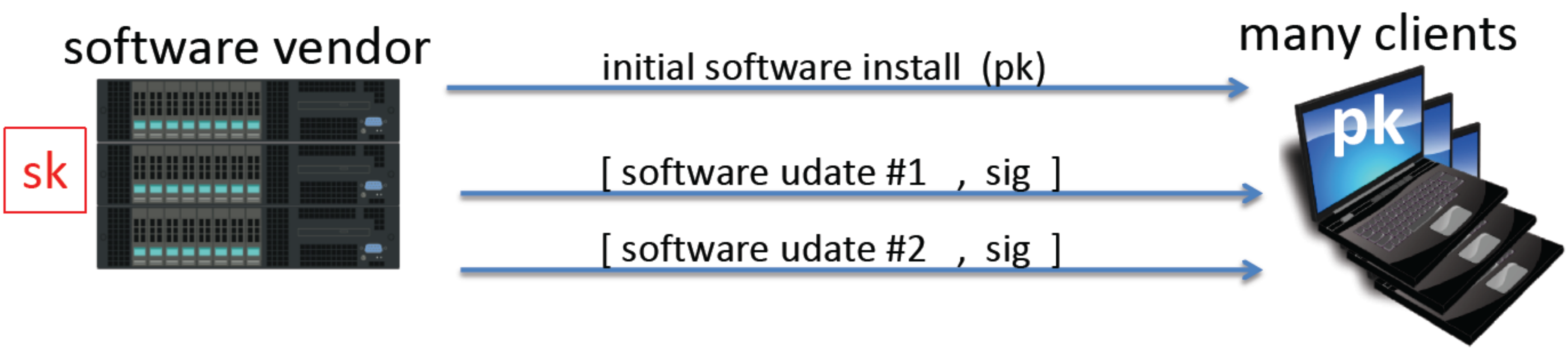
Can this signature be secure?

- Yes, the attacker cannot forge a signature for either m_0 or m_1
- No, signatures can be forged using a chosen msg attack
- It depends on the details of the scheme

Applications

Code signing:

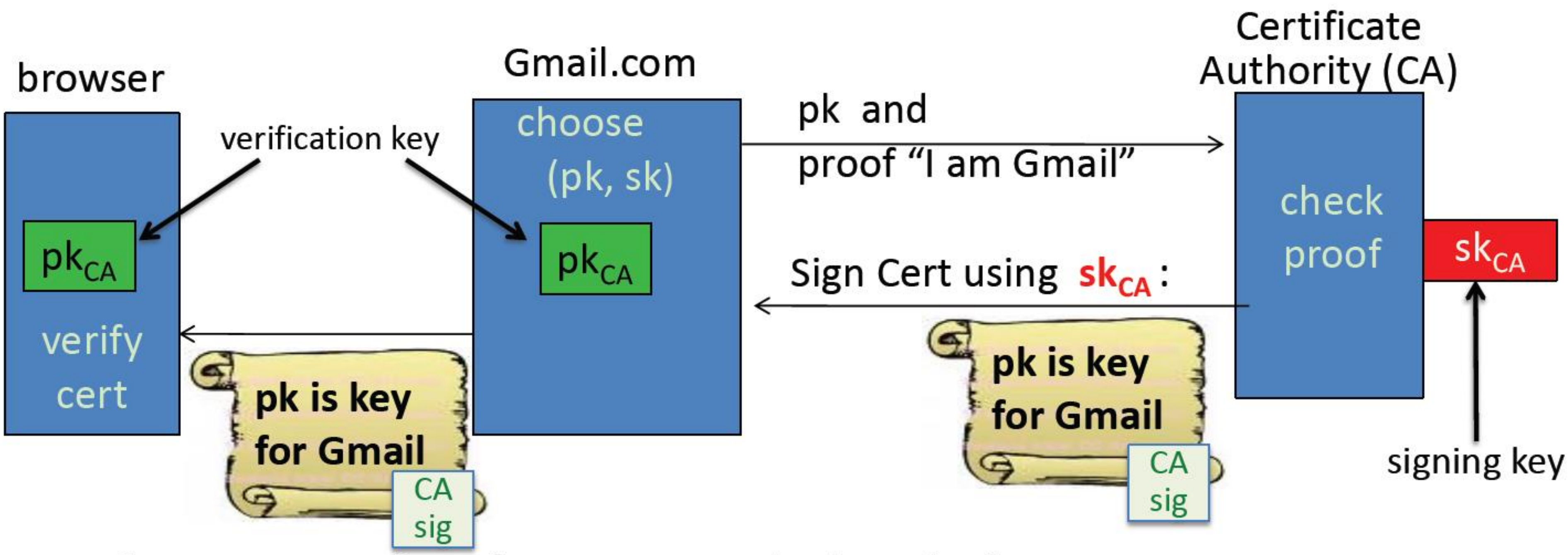
- Software vendor signs code
- Clients have vendor’s pk. Install software if signature verifies.



Important application: Certificates

Problem: browser needs server's public-key to setup a session key

Solution: server asks trusted 3rd party (CA) to sign its public-key pk



Server uses Cert for an extended period (e.g. one year)

Certificates: Example

Important fields:

Serial Number	5814744488373890497
Version	3
Signature Algorithm	SHA-1 with RSA Encryption (1.2.840.113549.1.1.5)
Parameters	none
Not Valid Before	Wednesday, July 31, 2013 4:59:24 AM Pacific Daylight Time
Not Valid After	Thursday, July 31, 2014 4:59:24 AM Pacific Daylight Time
Public Key Info	
Algorithm	Elliptic Curve Public Key (1.2.840.10045.2.1)
Parameters	Elliptic Curve secp256r1 (1.2.840.10045.3.1.7)
Public Key	65 bytes : 04 71 6C DD E0 0A C9 76 ...
Key Size	256 bits
Key Usage	Encrypt, Verify, Derive
Signature	256 bytes : 8A 38 FE D6 F5 E7 F6 59 ...

Equifax Secure Certificate Authority
↳ GeoTrust Global CA
↳ Google Internet Authority G2
↳ mail.google.com

mail.google.com
Issued by: Google Internet Authority G2
Expires: Thursday, July 31, 2014 4:59:24 AM Pacific Daylight Time
This certificate is valid

Details

Subject Name	
Country	US
State/Province	California
Locality	Mountain View
Organization	Google Inc
Common Name	mail.google.com
Issuer Name	
Country	US
Organization	Google Inc
Common Name	Google Internet Authority G2

Certificates: Example

What entity generates the CA's secret key sk_{CA} ?

- the browser
- Gmail
- the CA
- the NSA

Applications

<u>EMV payments</u>:

(greatly simplified)

Point of Sale terminal

transaction details

(nonce, PIN)

signature

(card cert)

sk

transaction details and signature

<u>Signed email</u>: sender signs email it sends to recipients

- Every recipient has sender's public-key (and cert).

  A recipient accepts incoming email if signature verifies.

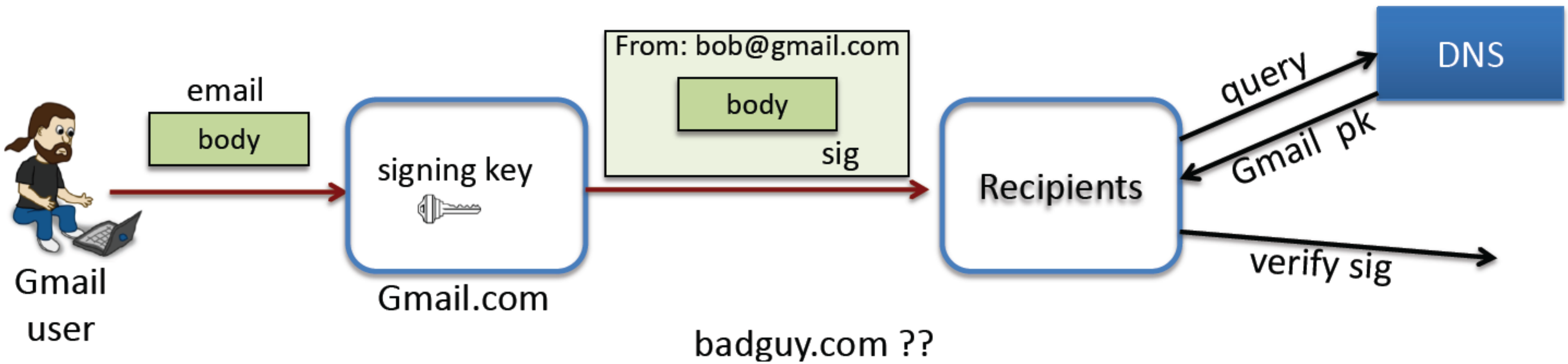
Signing email: DKIM (domain key identified mail)

Problem: bad email claiming to be from **someuser@gmail.com**

but in reality, mail is coming from domain **baguy.com**

⇒ Incorrectly makes gmail.com look like a bad source of email

Solution: **gmail.com** (and other sites) sign every outgoing mail



Example DKIM header from gmail.com

X-Google-DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed;

d=1e100.net; s=20130820; (lookup 20130820._domainkey.1e100.net in DNS for public key)

h=x-gm-message-state:mime-version:in-reply-to:references:from:date:
message-id:subject:to:content-type;

bh=MDr/xwte+/JQSgCG+T2R2Uy+SuTK4/gxqdxMc273hPQ=; (hash of message body)

b=dOTpUVOaCrWS6AzmcPMreo09G9viS+sn1z6g+GpC/ArkfMEmcffOJ1s9u5Xa5KC+6K
XRzwZhAWYqFr2a0ywCjbGECBPIE5ccOi9DwMjnvJRYEwNk7/sMzFfx+0L3nTqgTyd0ED
EGWdN3upzSXwBrXo82wVcRRCnQ1yUlTddnHgEoEFg5WV37DRP/eq/hOB6zFNTRBwkvfS
0tC/DNdRwftspO+UboRU2eiWaqJWPjxL/abS7xA/q1VGz0ZoI0y3/SCkxdg4H80c61DU
jdVYhCUd+dSV5fISouLQT/q5DYEjlNQbi+EcbL00liu4o623SDEeyx2isUgcvi2VxTWQ
m80Q==

Gmail's signature on headers, including DKIM header (2048 bits)

Applications: Summary

- Code signing
- Certificates
- Signed email (e.g. DKIM)
- Credit-card payments: EMV

and many more.

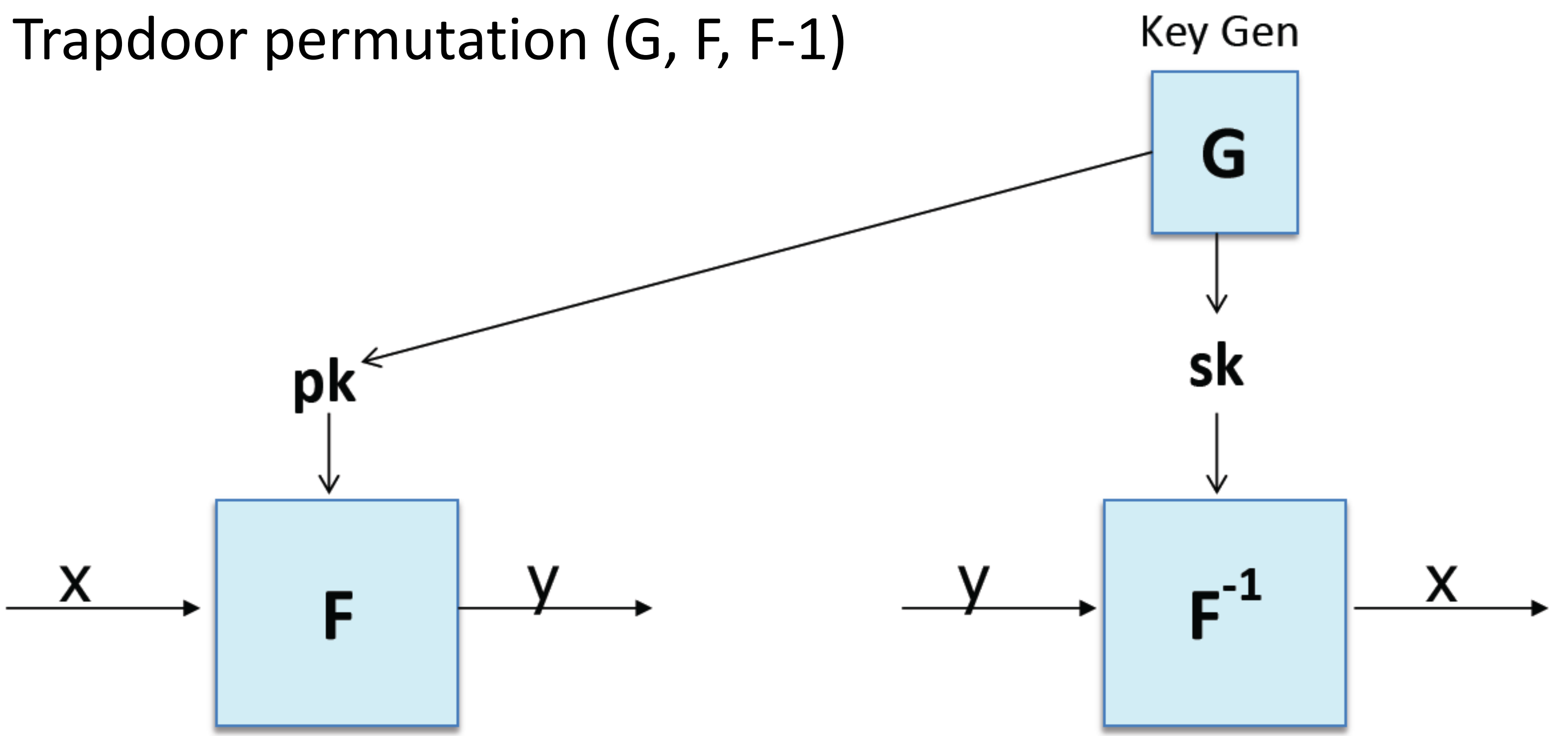
When to use signatures

Generally speaking:

- If one party signs and **one** party verifies: **use a MAC**
  - Often requires interaction to generate a shared key
  - Recipient can modify the data and re-sign it before passing the data to a 3rd party
- If one party signs and **many** parties verify: **use a signature**
  - Recipients **cannot** modify received data before passing data to a 3rd party (non-repudiation)

Signatures from Trapdoor Permutation

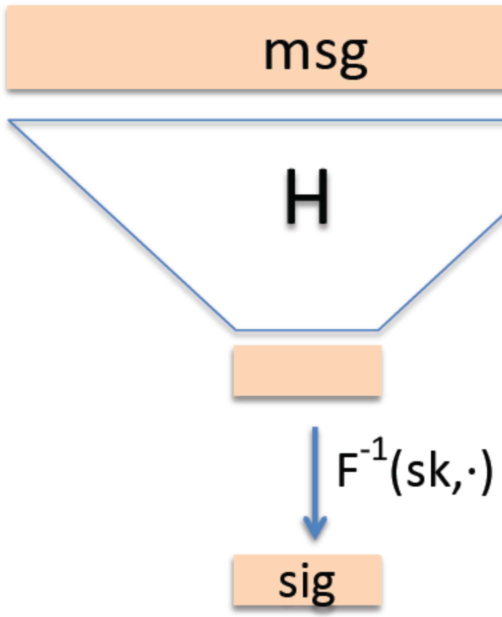
Trapdoor permutation (G, F, F-1)



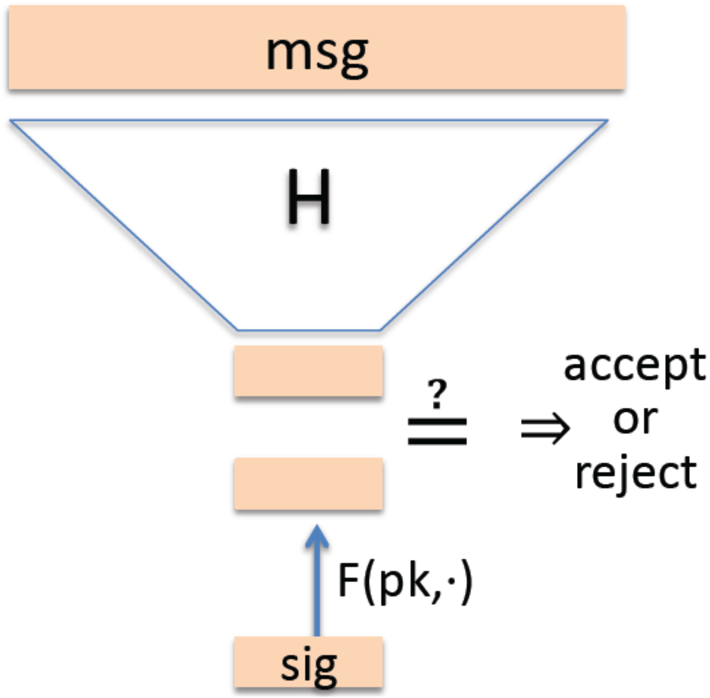
f(x) = F(pk, x) is one-to-one ($X \longrightarrow X$) and is a **one-way function.**

Full Domain Hash Signatures: pictures

S(sk, msg):



V(pk, msg, sig):



Full Domain Hash (FDH) Signatures

(G_{TDP}, F, F^{-1}): Trapdoor permutation on domain X

$H: M \longrightarrow X$ hash function (FDH)

(Gen, S, V) signature scheme:

- **Gen**: run G_{TDP} and output pk, sk
- **S(sk, m∈M)**: output $\sigma \longleftarrow F^{-1}(sk, H(m))$
- **V(pk, m, σ)**: output $\begin{cases} \text{'accept' if } F(pk, \sigma) = H(m) \\ \text{'reject' otherwise} \end{cases}$

Why hash the message?

Suppose we define NoHash-FDH as:

- **S'(sk, m∈X)**: output **$\sigma \longleftarrow F^{-1}(sk, m)$**
- **V'(pk, m, σ):** output 'accept' if **$F(pk, \sigma) = m$**

Is this scheme secure?

- ○ Yes, it is not much different than FDH
- ○ No, for any $\sigma \in X$, σ is a signature forgery for the msg $m = F(pk, \sigma)$
- ○ Yes, the security proof for FDH applies here too
- ○ It depends on the underlying TDP being used

RSA-FDH

Gen: generate an RSA modulus $N = p \cdot q$ and $e \cdot d = 1 \bmod \phi(N)$

construct CRHF $H: M \longrightarrow Z_N$

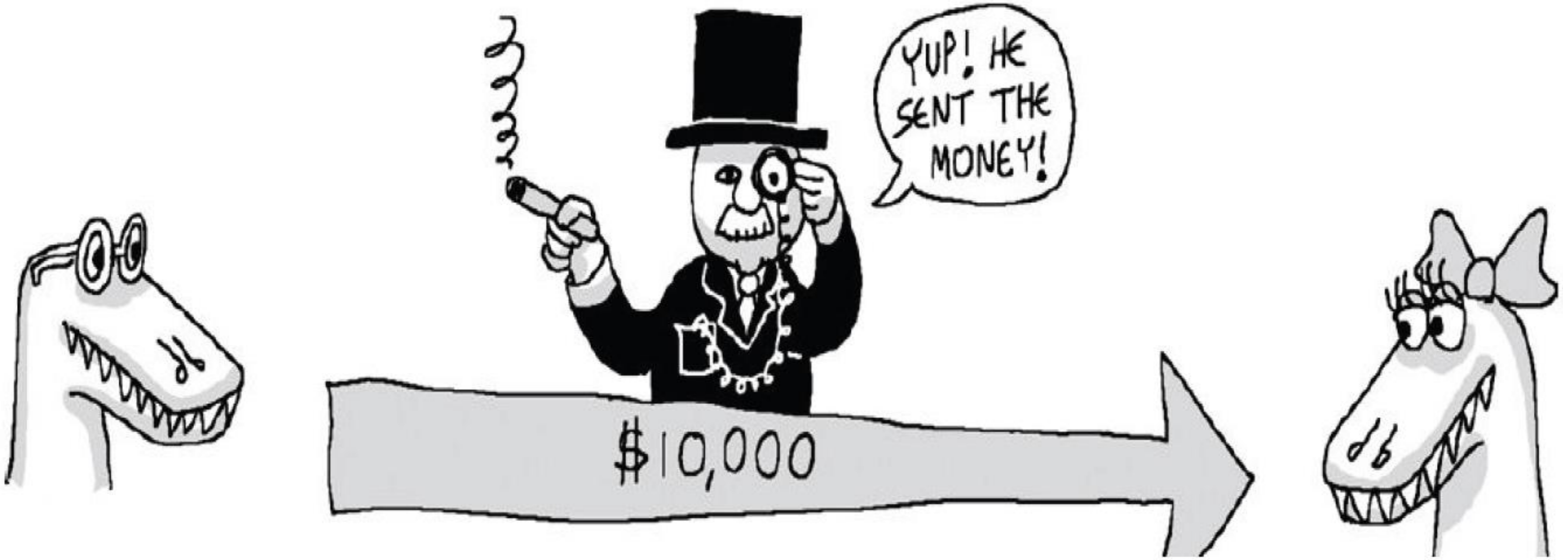
output $pk = (N,e,H)$, $sk = (N,d,H)$

- **S(sk, m∈M)**: output $\sigma \longleftarrow H(m)^d \bmod N$
- **V(pk, m, σ)**: output 'accept' if $H(m) = \sigma^e \bmod N$

Lecture 4.8 : Block Chain(Cryptocurrency)

Traditional online financial transactions using third trusted party

1. Validate Entries
2. Safeguard Entries
3. Preserve Historic Record



Bitcoin

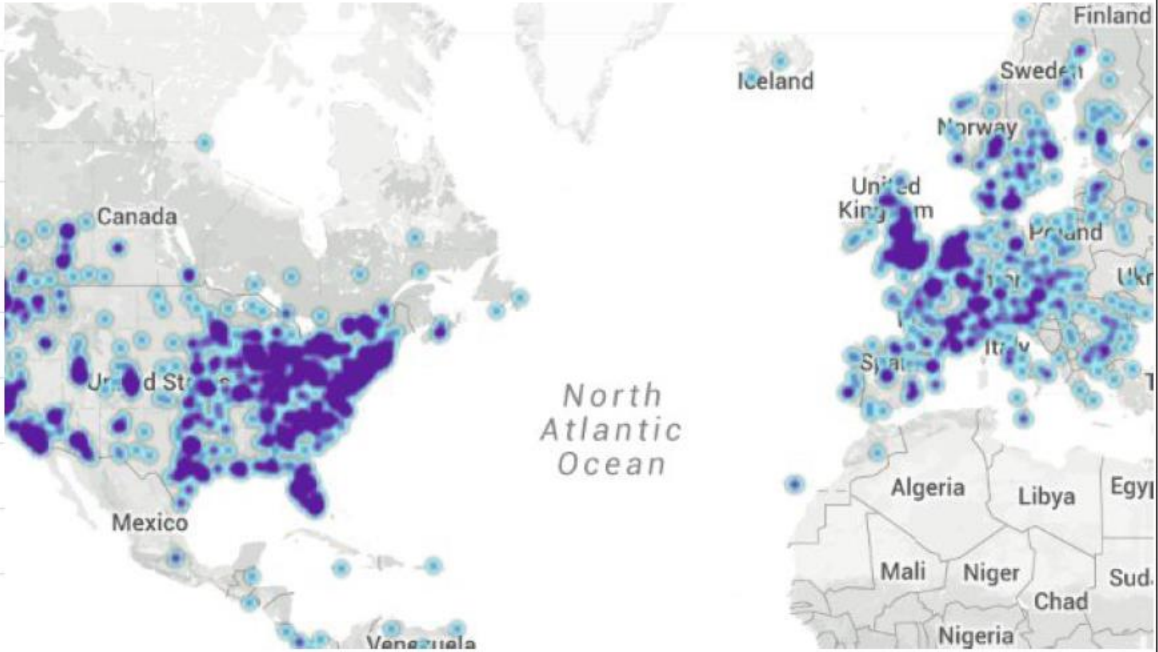
In his announcement of Bitcoin in late 2008, Satoshi said he developed “A Peer-to-Peer Electronic Cash System.“

The single most important part of Satoshi‘s invention was that he found a way to build a decentralized digital cash system. In the nineties, there have been many attempts to create digital money, but they all failed.

After seeing all the centralized attempts fail, Satoshi tried to build a digital cash system without a central entity. Like a Peer-to-Peer network for file sharing.

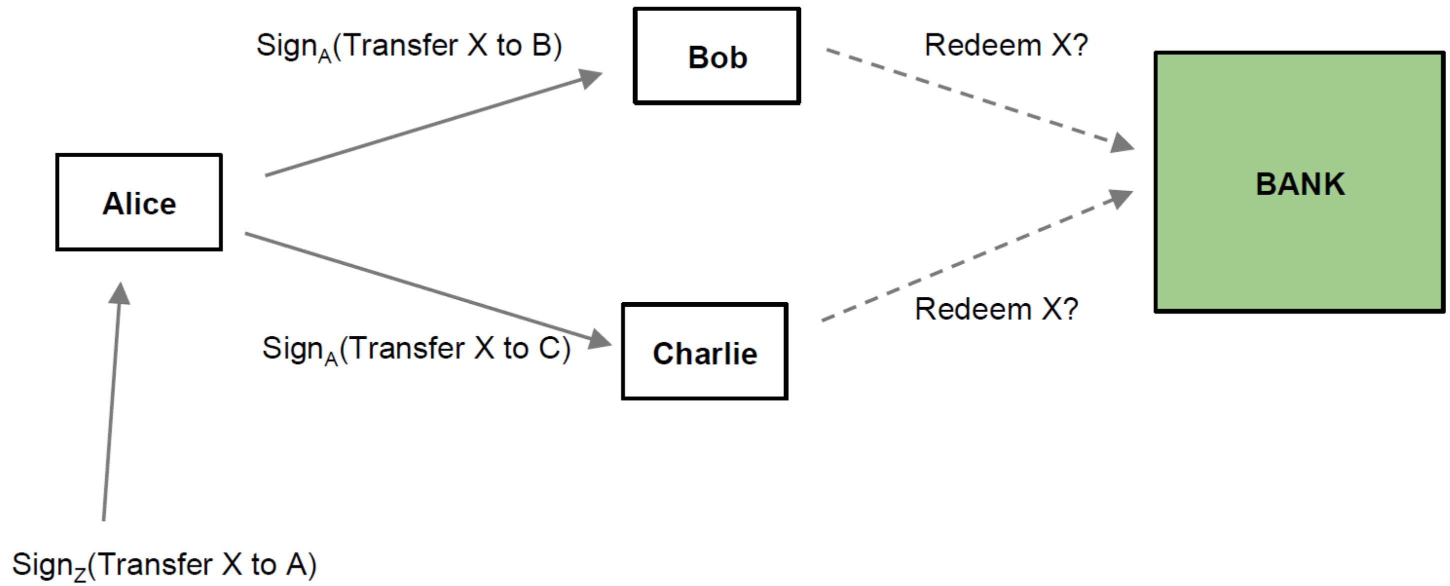
Distribution of (Reachable) Nodes

RANK	COUNTRY	NODES
1	United States	2685 (41.81%)
2	Germany	519 (8.08%)
3	Canada	412 (6.42%)
4	France	360 (5.61%)
5	United Kingdom	353 (5.50%)
6	Netherlands	271 (4.22%)
7	Russian Federation	221 (3.44%)
8	China	161 (2.51%)




Google
1 btc to usd
全部
新闻
图片
地图
图书
更多
设置
工具
找到约 4,750,000 条结果 （用时 0.47 秒）
1 Bitcoin 等于
7,285.00 美元
1
Bitcoin
7285.01
美元
免责声明
Bitcoin (USD) Price, Market Cap, Charts, News - CoinDesk
https://www.coindesk.com/price/ ▾ 翻译此页
May 25, 2018 at 15:15 | Omkar Godbole. Shadowing the losses in bitcoin, the top-25 cryptocurrencies have all fallen over the last seven days – all bar one, that ...
Bitcoin Calculator · Ripple (XRP) · Ethereum Price · Bitcoin Bull Trap? Not So ...

Double Spending: Why ecash is hard

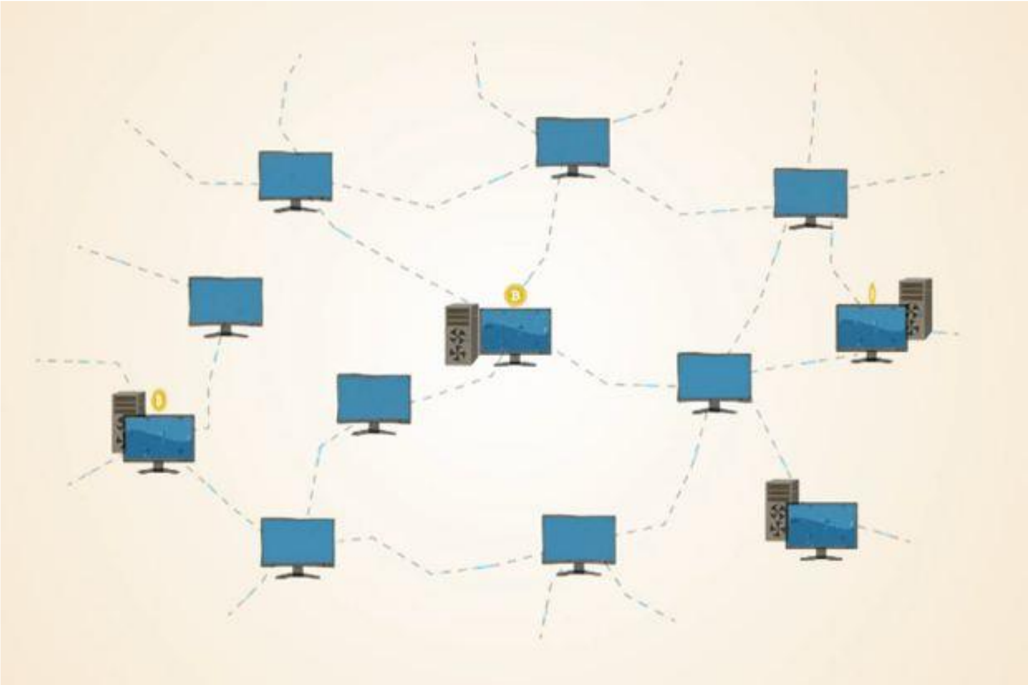


To realize digital cash you need a payment network with accounts, balances, and transaction.

One major problem every payment network has to solve is to prevent the so-called double spending: to prevent that one entity spends the same amount twice. Usually, this is done by a central server who keeps record about the balances.

Double Spending: Why ecash is hard

In a decentralized network, you don‘t have this server. So you need every single entity of the network to do this job. Every peer in the network needs to have a list with all transactions to check if future transactions are valid or an attempt to double spend.



Digital Signature: syntax

<u>Def</u>: a signature scheme (Gen,S,V) is a triple of algorithms:

- Gen(): randomized alg. outputs a key pair (pk, sk)
- S(sk, $m \in M$) outputs sig. σ
- V(pk, m, σ) outputs 'accept' or 'reject'

Consistency: for all (pk, sk) output by Gen :

$$\forall m \in M: \quad V\big(pk,\ m,\ S(sk, m)\big) = \text{'accept'}$$

Bitcoin is a list of transactions

Bitcoin is using signature algorithm (ECSDA/ P256)

Private Key: like a password for traditional account, sign transactions

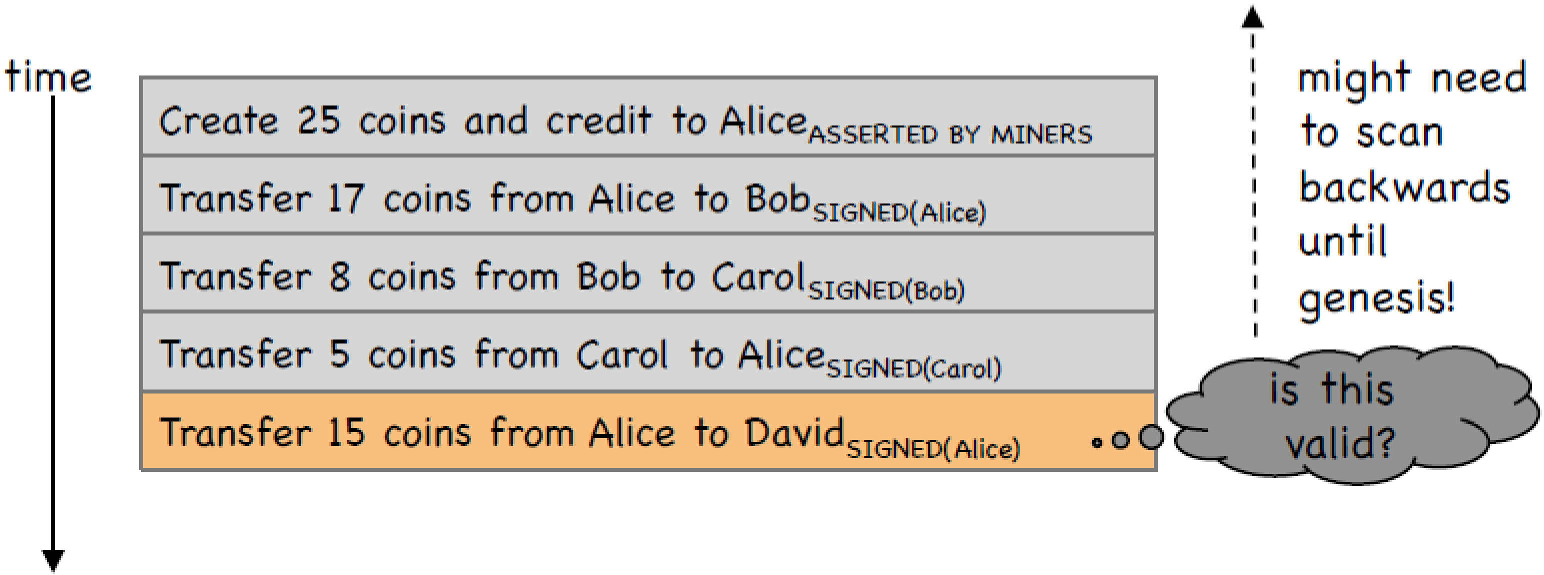
Public Key: like an account no.

A transfers 10 coins to B

A announce:
transfer 10 coins to Public Key of B, signed with the Private Key of A

Bitcoin is a list of transactions

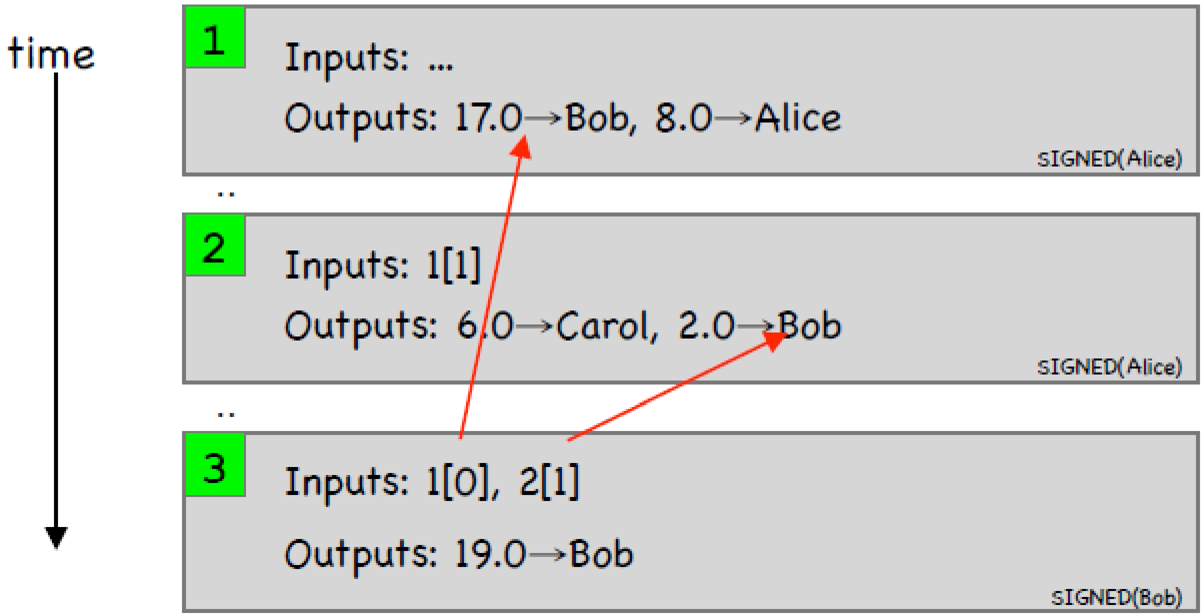
An account-based ledge



SIMPLIFICATION: only one transaction per block

Bitcoin is a list of transactions

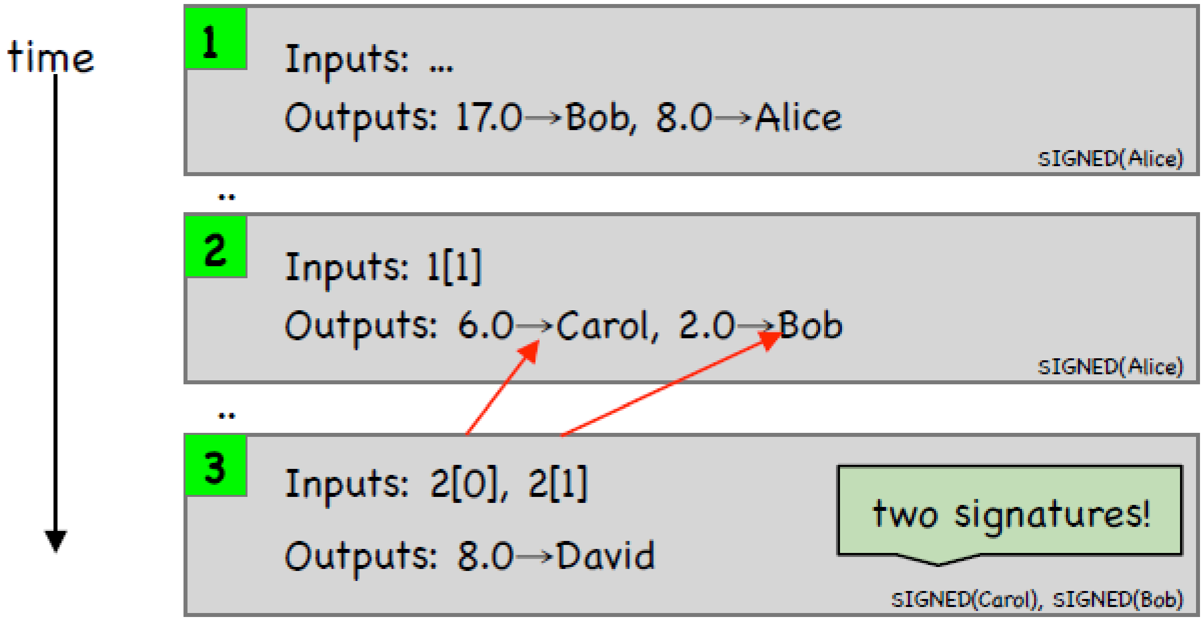
Merge Value (Bitcoin)



SIMPLIFICATION: only one transaction per block

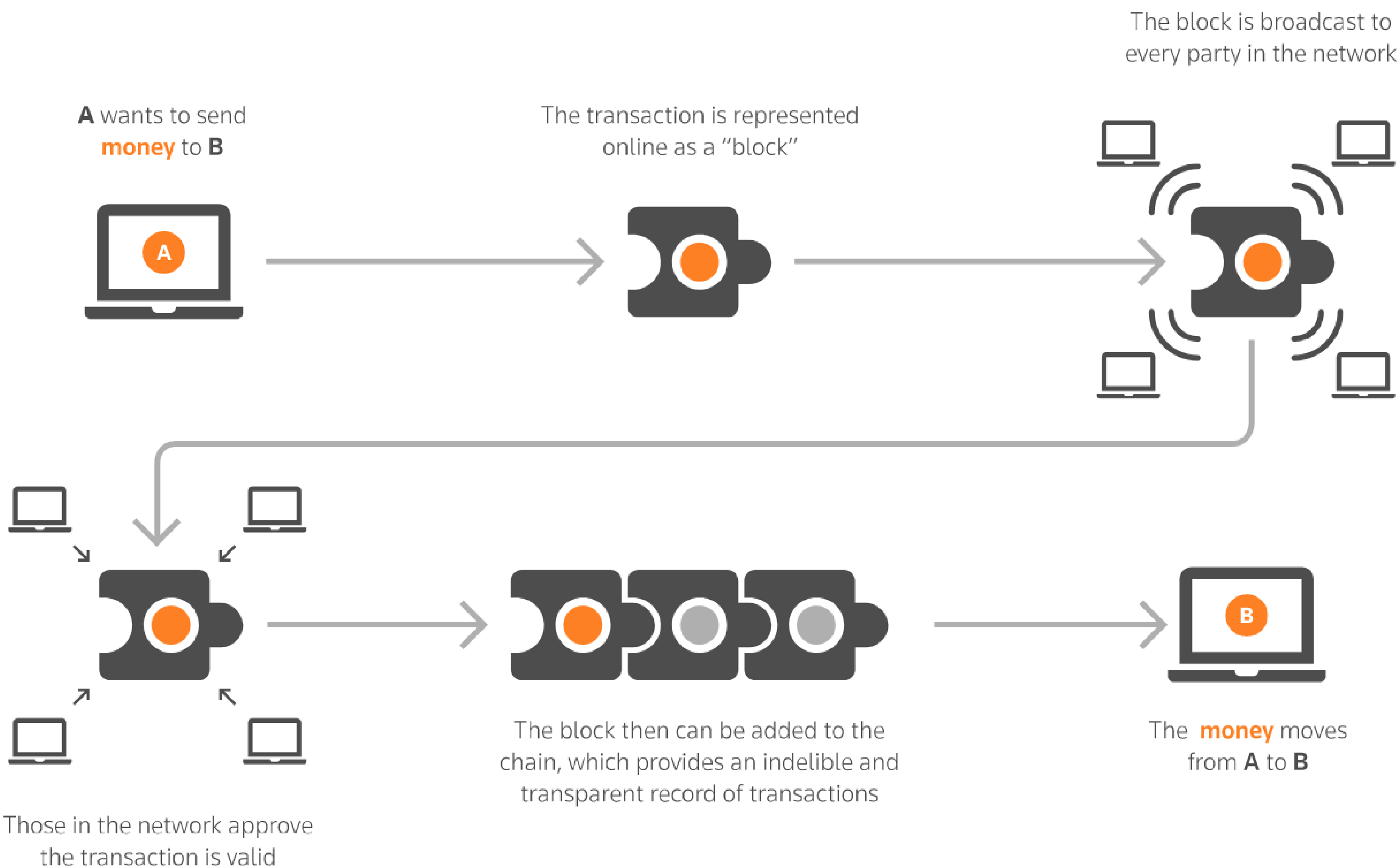
Bitcoin is a list of transactions

Joint Payment (Bitcoin)



SIMPLIFICATION: only one transaction per block

How a Block Chain works?

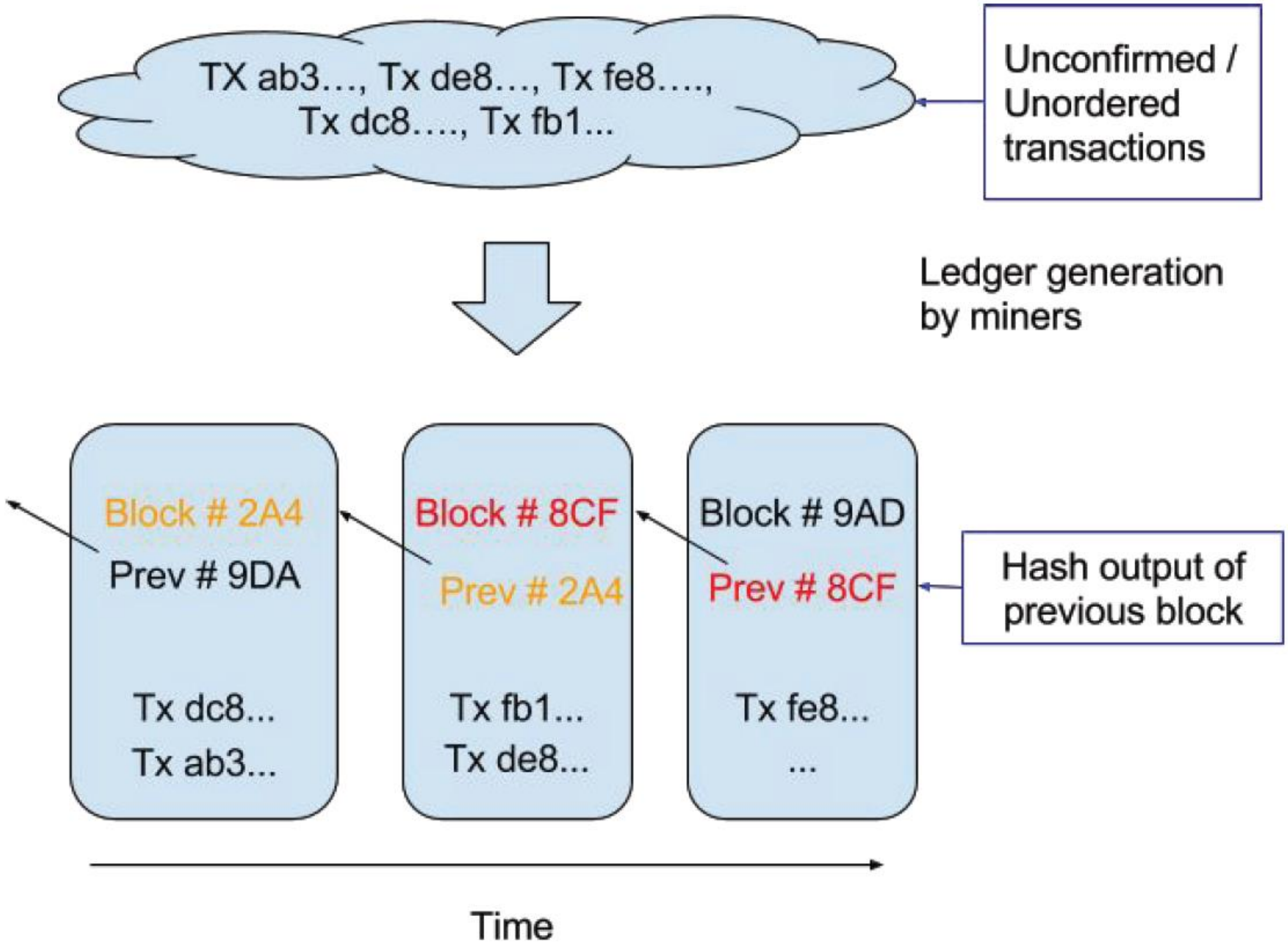


Maintain order of transactions

Considering that the transactions are passed node by node through the Bitcoin network, there is no guarantee that orders in which they are received at a node are the same order in which these transactions were generated.

The Bitcoin system orders transactions by placing them in groups called blocks and then linking these blocks through what is called Block Chain.

Maintain order of transactions



What is the next Block?

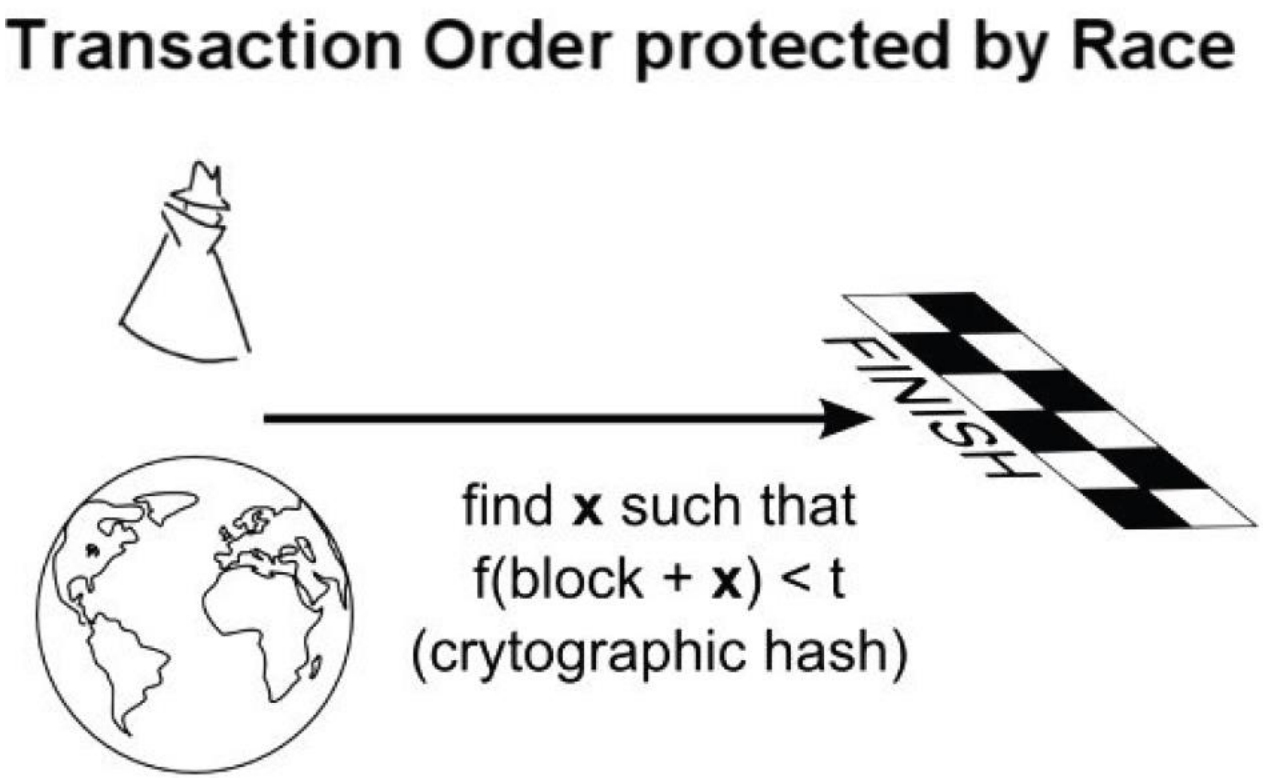
Any node in the network can collect unconfirmed transactions and create a block and then broadcasts it to rest of the network as a suggestion as to which block should be the next one in the block chain.

which block should be next in the block chain?

Bitcoin solve this problem by introducing a mathematical puzzle: each block will be accepted in the block chain provided it contains an answer to a very special mathematical problem.

The nodes donating their computing resources to solve the puzzle and generate block are called “miner” nodes” and are financially awarded for their efforts.

What is the next Block?



The complexity of the problem can be adjusted so that on average it takes ten minutes for a node in the Bitcoin network to make a right guess and generate a block.

Mathematical race to protect transactions

The network only accepts the longest block chain as the valid one.

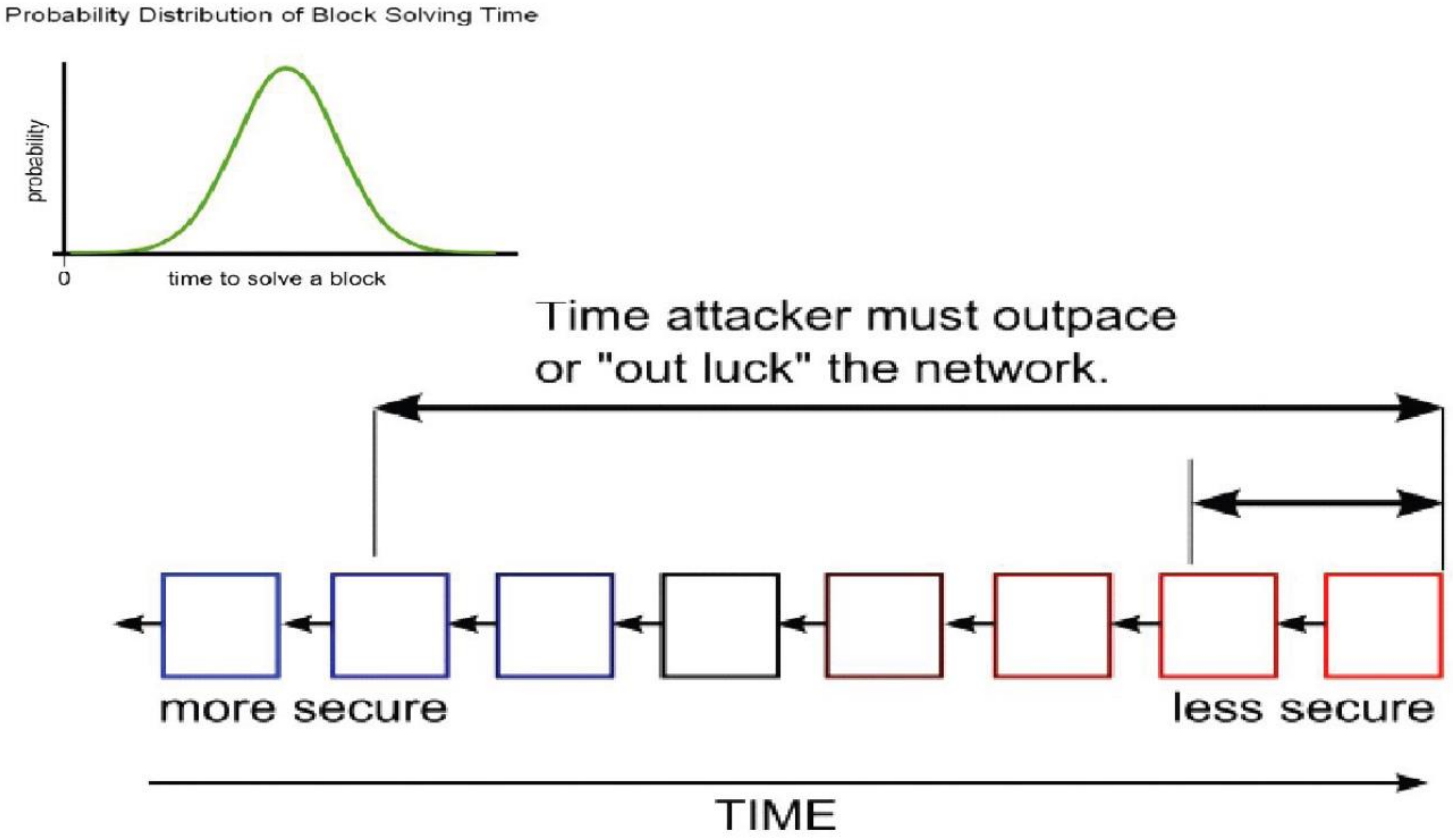
It is next to impossible for an attacker to introduce a fraudulent transaction.

1. has to generate a block by solving a mathematical puzzle

2. at the same time mathematically race against the good nodes to generate all subsequent blocks

3. more difficult since blocks in the block chain are linked cryptographically together

What is the next Block?



Bitcoin Blocks

Q: Why bundle transactions together?

1. Requiring consensus for each transaction separately would reduce transaction acceptance rate.

2. Hash-chain of blocks is much shorter.

3. Faster to verify history.

Transactional Properties of CRYPTOcurrencies

Blockgeeks

Cryptocurrency opens the door for revolutionary technological possibilities

Irreversible

After a confirmation a transaction can't be reversed. By nobody. And nobody means nobody. Not you, not your bank, not the president of the United States, not Satoshi, not your miner. Nobody. If you send money, you send it. Period. No one can help you, if you sent your funds to a scammer or if a hacker stole them from your computer. There is no safety net

Pseudonymous

Neither transactions nor accounts are connected to real world identities. You receive Bitcoins on so-called addresses, which are randomly seeming chains of around 30 characters. While it is usually possible to analyze the transaction flow, it is not necessarily possible to connect the real world identity of users with those addresses

Fast and global

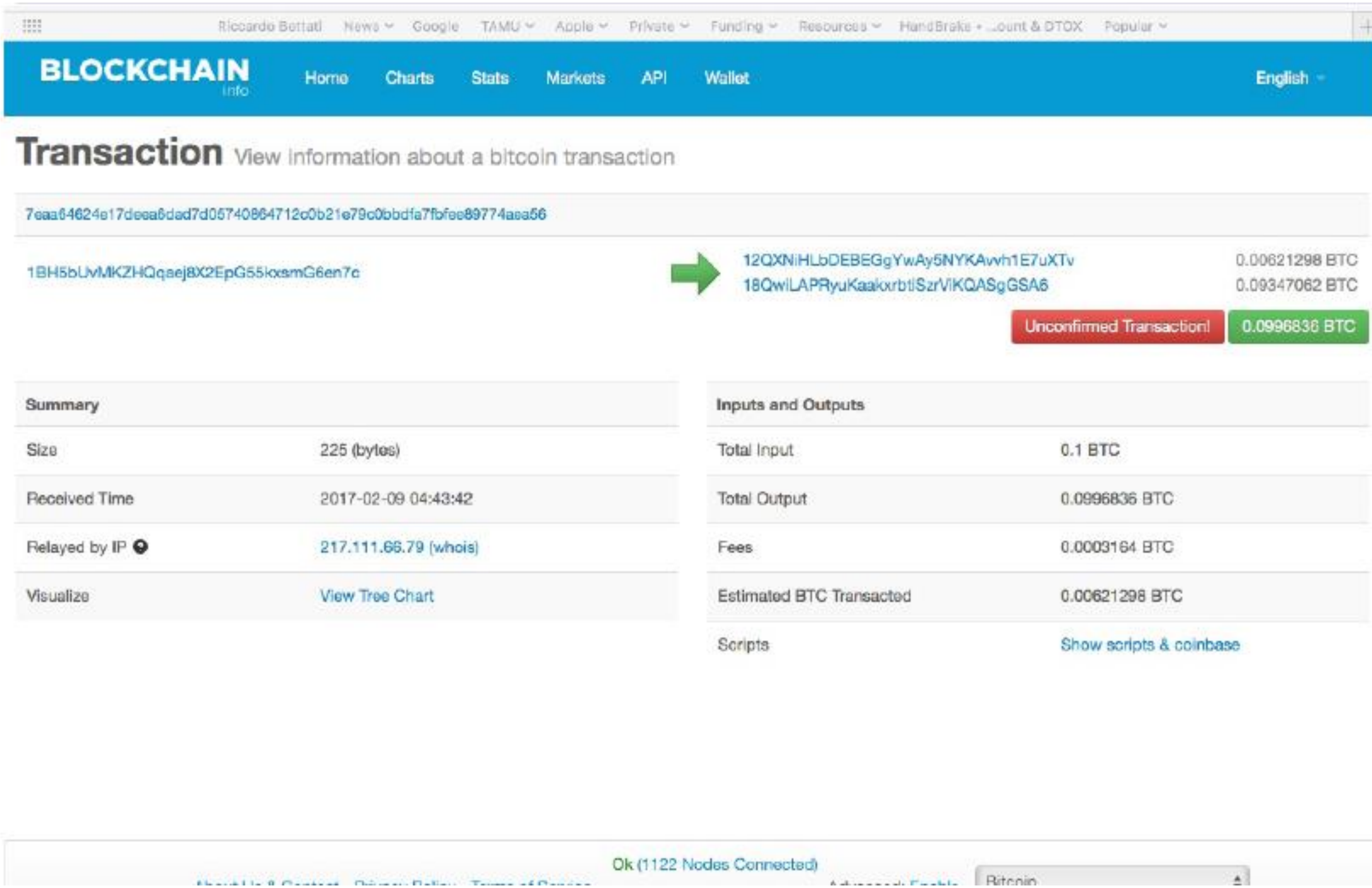
Transaction are propagated nearly instantly in the network and are confirmed in a couple of minutes. Since they happen in a global network of computers they are completely indifferent of your physical location. It doesn't matter if I send Bitcoin to my neighbour or to someone on the other side of the world.

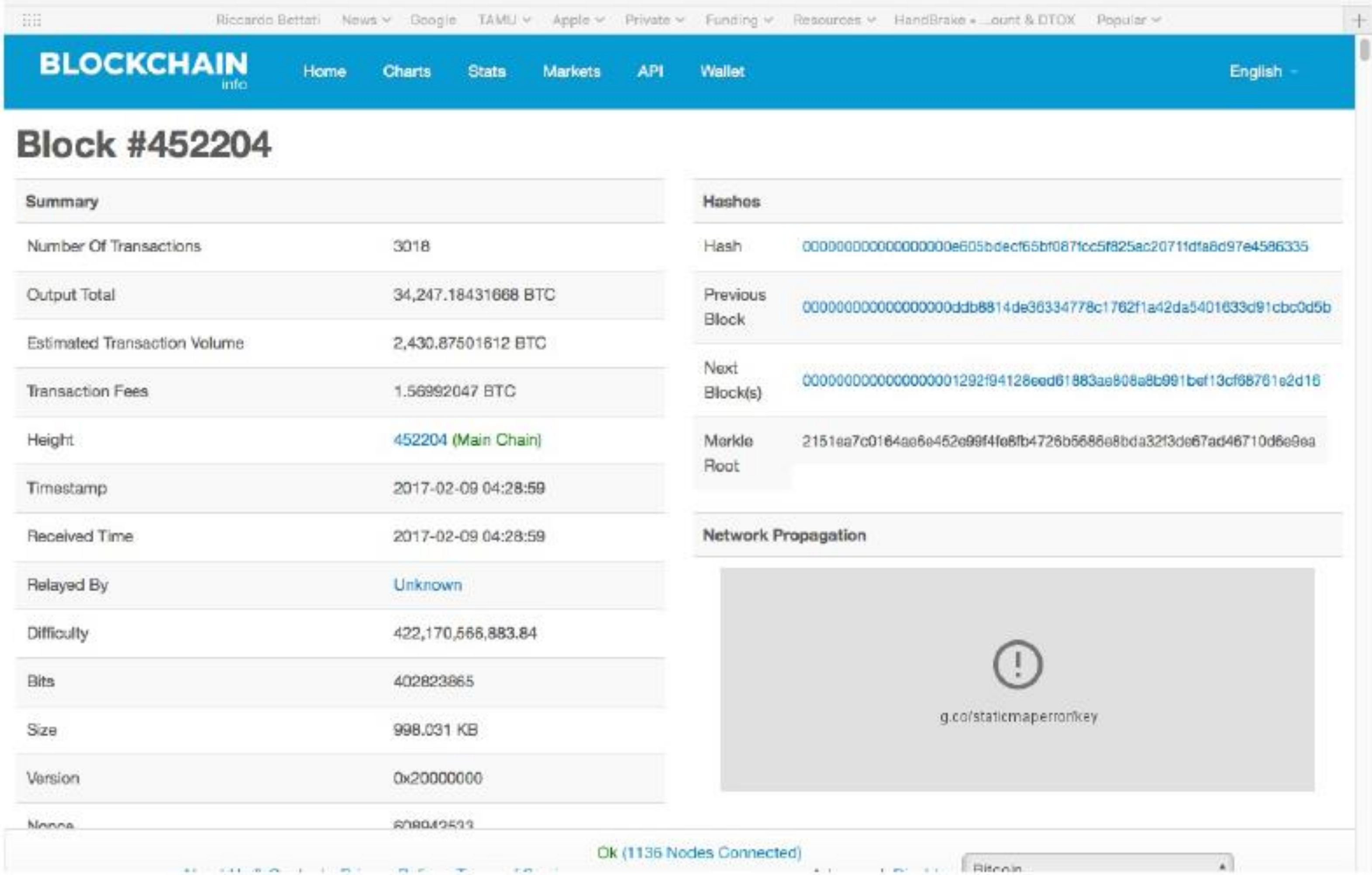
Secure

Cryptocurrency funds are locked in a public key cryptography system. Only the owner of the private key is able to send cryptocurrency. Strong cryptography and the magic of big numbers makes it impossible to break this scheme. A Bitcoin address is more secure than Fort Knox.

Permissionless

You don't have to ask anybody to use cryptocurreny. It's just a software that everybody can download for free. After you installed it, you can receive and send Bitcoins or other cryptocurrency. No one can prevent you. There is no gatekeeper.


BLOCKCHAIN info
Home
Charts
Stats
Markets
API
Wallet
English
Transaction View information about a bitcoin transaction
7eaa64624e17deea6dad7d05740864712c0b21e79c0bbdfa7fbfee89774aea56
1BH5bUvMKZHQqaej8X2EpG55kxsmG6en7c
12QXNiHLbDEBEGgYwAy5NYKAvvh1E7uXTv
0.00621298 BTC
18QwiLAPRyuKaakxrbtiSzrViKQASgGSA6
0.09347062 BTC
Unconfirmed Transaction!
0.0996836 BTC
Summary
Size
225 (bytes)
Received Time
2017-02-09 04:43:42
Relayed by IP
217.111.66.79 (whois)
Visualize
View Tree Chart
Inputs and Outputs
Total Input
0.1 BTC
Total Output
0.0996836 BTC
Fees
0.0003164 BTC
Estimated BTC Transacted
0.00621298 BTC
Scripts
Show scripts & coinbase
Ok (1122 Nodes Connected)


Riccardo Bettati
News
Google
TAMU
Apple
Private
Funding
Resources
HandBrake • ...ount & DTOX
Popular
BLOCKCHAIN
info
Home
Charts
Stats
Markets
API
Wallet
English
Block #452204
Summary
Number Of Transactions
3018
Output Total
34,247.18431668 BTC
Estimated Transaction Volume
2,430.87501612 BTC
Transaction Fees
1.56992047 BTC
Height
452204 (Main Chain)
Timestamp
2017-02-09 04:28:59
Received Time
2017-02-09 04:28:59
Relayed By
Unknown
Difficulty
422,170,566,883.84
Bits
402823865
Size
998.031 KB
Version
0x20000000
Hashes
Hash
000000000000000000e605bdecf65bf0871cc5f825ac20711dfa8d97e4586335
Previous Block
000000000000000000ddb8814de36334778c1762f1a42da5401633d91cbc0d5b
Next Block(s)
0000000000000000001292f94128eed61883ae808a8b991bef13cf68761e2d16
Merkle Root
2151ea7c0164ae6e452e99f4fe8fb4726b5686e8bda32f3de67ad46710d6e9ea
Network Propagation
g.co/staticmaperrorkey
Ok (1136 Nodes Connected)

(year 2016)

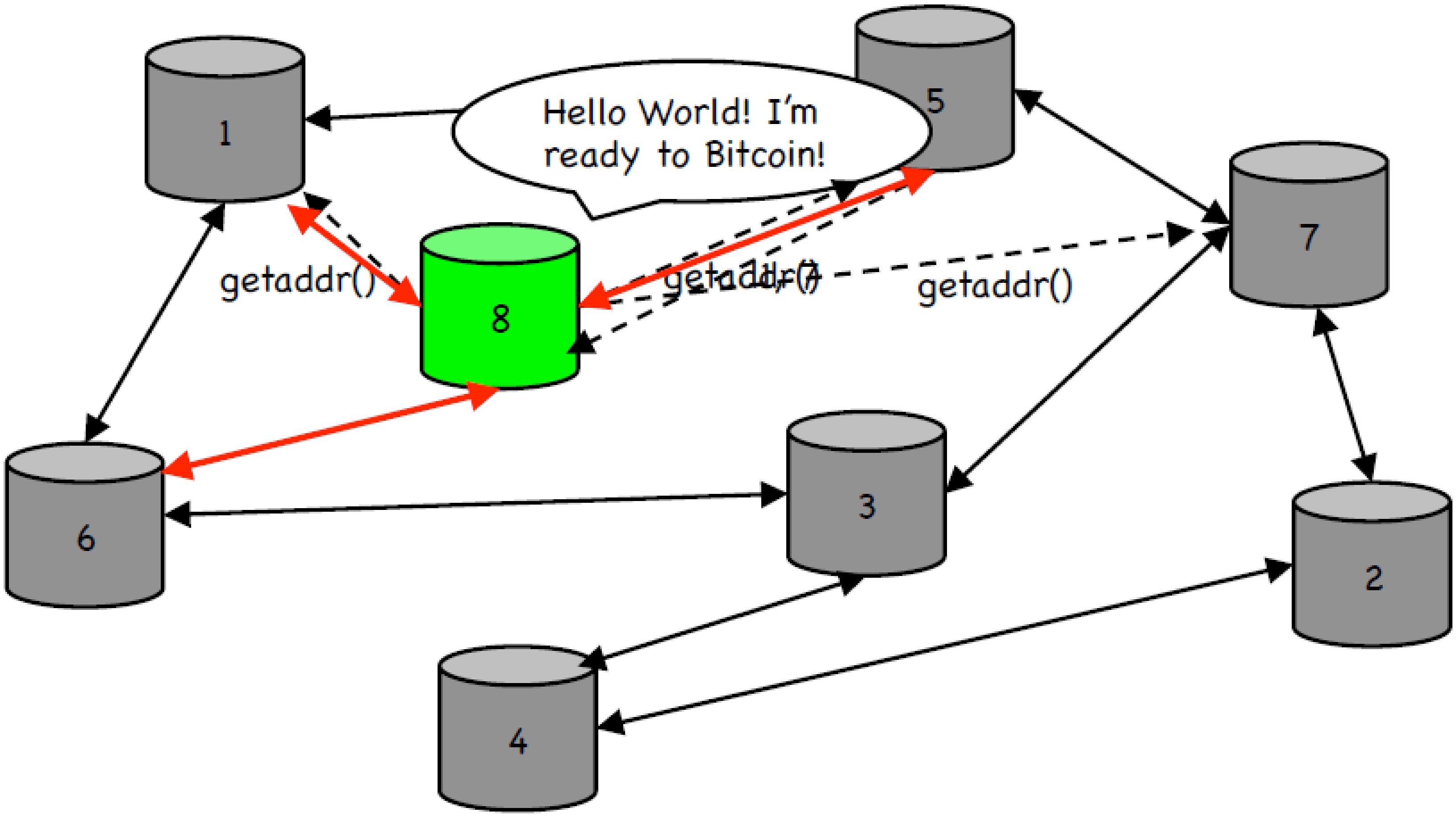
Examples of Cryptocurrency

Bitcoin Market Cap
$11,322,347,786

Ethereum Market Cap
$928,068,434

Ripple Market Cap
$293,888,278

The Bitcoin Network



Joining the Bitcoin P2P Network

Should a Node relay a proposed transaction?

Transaction valid with current block chain

- (default) script matches a whitelist
  – Avoid unusual scripts
- Haven't seen before
  – Avoid infinite loops
- Doesn't conflict with others I've relayed
  – Avoid double-spends

Sanity checks only...
Some nodes may ignore them!

Block Propagation

Propagation of blocks is nearly identical:

Relay a new block when you hear it if:

1. Block meets the hash target

2. Block has all valid transactions
– Run all scripts, even if you wouldn't relay

3. Block builds on current longest chain
– Avoid forks

Existing Market

There was another application "Smart Contracts" that was invented in year 1994 by Nick Szabo. It was a great idea to automatically execute contracts between participating parties.

Now two programs block chain and smart contract can work together to trigger payments when a preprogrammed condition of a contractual agreement is triggered.

Open source companies like Ethereum and Codius are enabling Smart Contracts using blockchain technology.

Ethereum allows anyone to create their own cryptocurrency and use that to execute, pay for smart contracts.

Risk Adoptions

Scaling: Imagine yourself executing a Block Chain transaction for the first time. You will have to go through downloading the entire set of existing Block Chains and validate before executing your first transaction. This may take hours or longer as the number of blocks increase exponentially.

Bootstrapping: Moving the existing contracts or business documents/frameworks to the new Block Chain based methodology presents a significant set of migration tasks that need to be executed.

Fraudulent Activities: Given the pseudonymous nature of BlockChain transactions, coupled with ease of moving valuables, the bad guys may misuse this for fraudulent activities like money trafficking.

Throughput Limit of Bitcoin

Blocks are limited to 1 M bytes each (10 min)

With

at least 250 bytes/transaction

this gives about

7 transactions/sec!

Compare to:

- VISA: **2,000-10,000** transactions/sec
- PayPal: **50-100** transaction/sec

Crytographic Limit of Bitcoin

1. Only 1 signature algorithm (ECDSA/P256)
2. Hard-coded hash functions

Crypto primitives might break by 2040…

Fundamentals of Information Science II:

- Inference / Statistical Signal Processing
- Machine Learning and Graphical Model
- Deep Learning