Crypto-Hashing and applications

Curs 2017

Cryptography

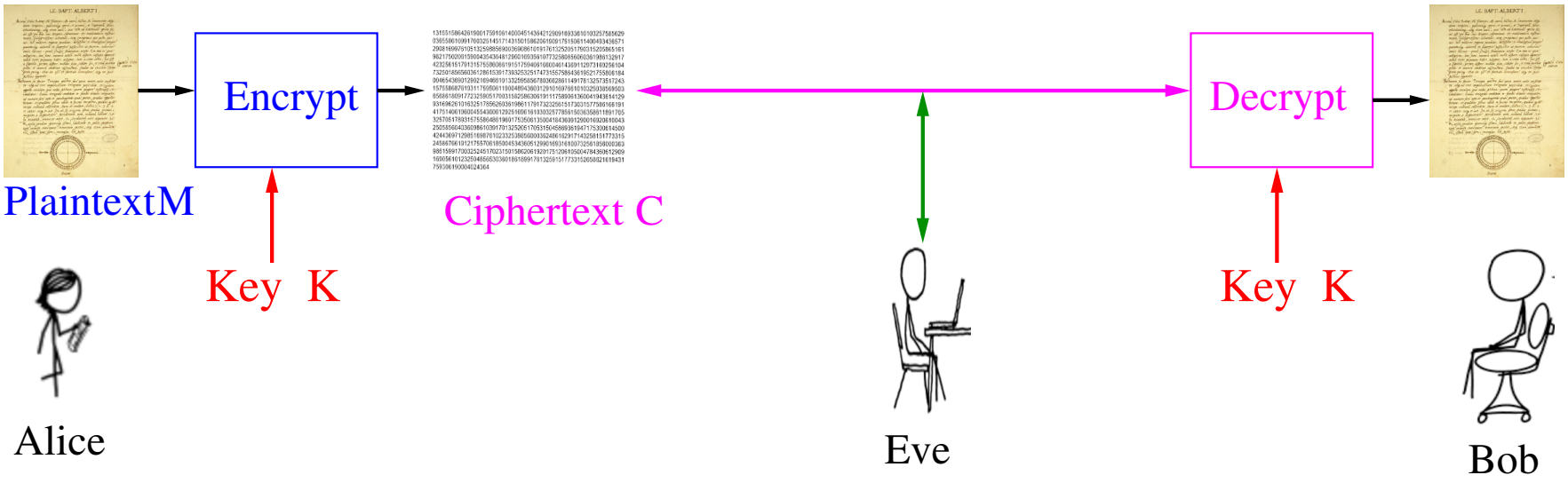
Cryptography is the study of techniques for secure communication in the presence of adversaries.

- Ciphertext (cryptogram): encrypted text
- Encryption (enciphering): plaintext $\rightarrow$ a ciphertext.
- Decryption (deciphering): ciphertext $\rightarrow$ plaintext.
- Cryptographic system (cipher): encrypting or decrypting.
- Cryptanalysis: the process to break the code.
- Key: symbols used to encrypt and decrypt.
- Key space: The total number of keys that can be used in a cryptographic system.

Cryptography is as old as tribal fights, see for ex:
The Code Book, Simon Singh. Fourth State, 1999.

Cryptography

Cryptography is the study of techniques for secure communication in the presence of adversaries.



The adversary Eve can eavesdrop all communication between Alice and Bob so if Alice an Bob want to keep secret the contends, they must encrypt the plaintext messages into a ciphertext which Eve can't break.

To make clear the meaning of encryption

- Encoding: Transform data so that it can be properly consumed by a different type of system, (binary data being sent over email, ASCII, URL Encoding, ...). The goal is not to keep information secret, but rather to ensure that can be properly consumed. Encoding works by using a scheme that is publicly available so that it can easily be reversed. It does not require a key as the only thing required to decode it is the algorithm that was used to encode it.
- Encryption: Transformation of data in order to keep it secret from others, (sending a password over internet using PGP). The goal is to ensure the data cannot be consumed by anyone other than the intended agent. Encryption transforms data into another format in such a way that only specific individual(s) can reverse the transformation. It uses a key, which is kept secret.

Public Key Cryptography: One way functions

A one-way function is a function that is feasible to compute, but hard to invert.

- Multiplication: Given $x, y \in \mathbb{Z}$ finding $x \cdot y = z$ is easy to do but given z it is hard to find x, y. Need factorization, which is not know to be in P. RSA based in the difficulty given the product of 2 very large primes, find the primes from the product.
- Modular exponentiation Given a prime p and an integer $x \in [0, p-1]$, compute $2^x \mod p$. Easy to compute in $O((\lg n)^n)$. discrete logarithm given p and $y \in [0, p-1]$ find x s.t. $y = 2^x \mod p$
- Elliptic Curve sum: Given an elliptic curve defined by a cubic equation on $\mathbb{R}$ of the type $y^2 = x^3 + ax + b$, a point p on it, and given a $x \in \mathbb{Z}^+$, define the elliptic sum $x * p$ as x times the additive sum of p (within the curve). Finding $s = x * p$ is easy, given p and s finding x is hard.

Public key encryption

Useful for digital signatures

$\mathcal{A}$ send a message M to $\mathcal{B}$, Eve can eavesdrop M

How can we assure Eve can not recover M?

Private-Key Systems: Key F (one-way function) is secret. Both, $\mathcal{A}$ and $\mathcal{B}$ have a copy of F and F^{-1} (dangerous)
To encrypt message M: compute $X = F(M)$
To decrypt: compute $M = F^{-1}(X)$

Public-Key Systems:

Diffie-Hellman

$S = F$ is private and secret, $P = F^{-1}$ is public. To know P does not help in discovering S.

$M : \mathcal{A} \to \mathcal{B}$. Eve eavesdropper.

Public Key: $P_{\mathcal{A}}, P_{\mathcal{B}}$,

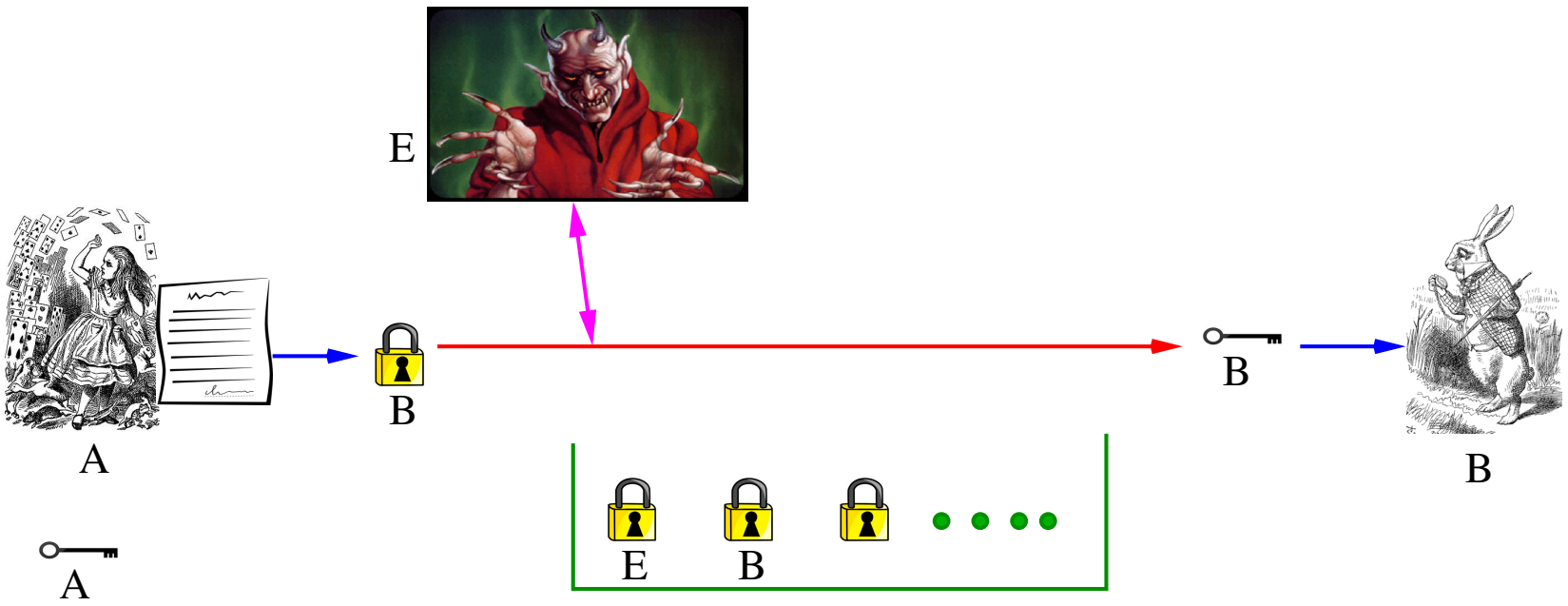
Secret Key: $S_{\mathcal{A}}, S_{\mathcal{B}}$,

Secret and Public keys must have the following property: for any person A we must have $M = S_{\mathcal{A}}(P_{\mathcal{A}}(M)) = P_{\mathcal{A}}(S_{\mathcal{A}}(M))$.

Public-Key Systems:

To send $M : \mathcal{A} \to \mathcal{B}$,
(1.-) $\mathcal{A}$ gets $P_{\mathcal{B}}$,
(2.-) $\mathcal{A}$ computes the ciphertext $C = P_{\mathcal{B}}$,
(3.-) $\mathcal{A}$ sends C to $\mathcal{B}$.

When $\mathcal{B}$ gets C: $S_{\mathcal{B}}(C) = S_{\mathcal{B}}(P_{\mathcal{B}}(M)) = M$



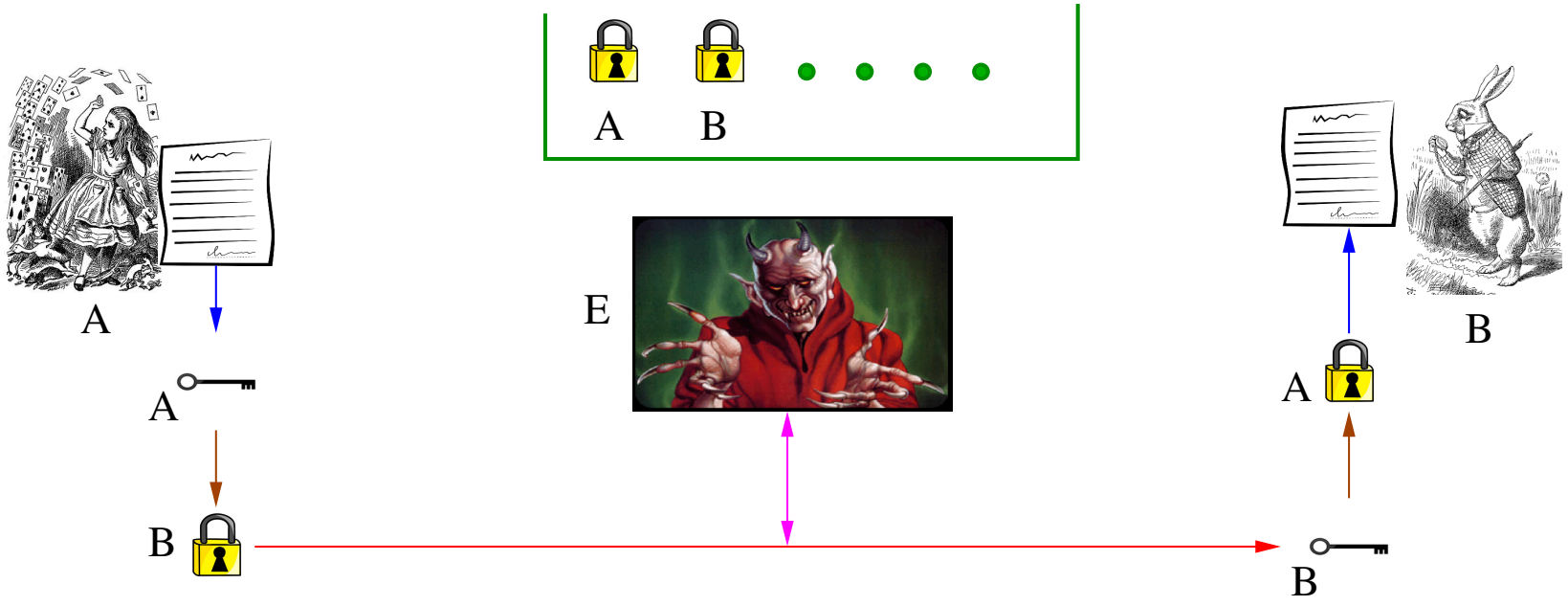
Digital signature

$\mathcal{A}$ sends to $\mathcal{B}$ (M, σ) such that $\mathcal{B}$ knows that only $\mathcal{A}$ could have send M.

σ is called the signature

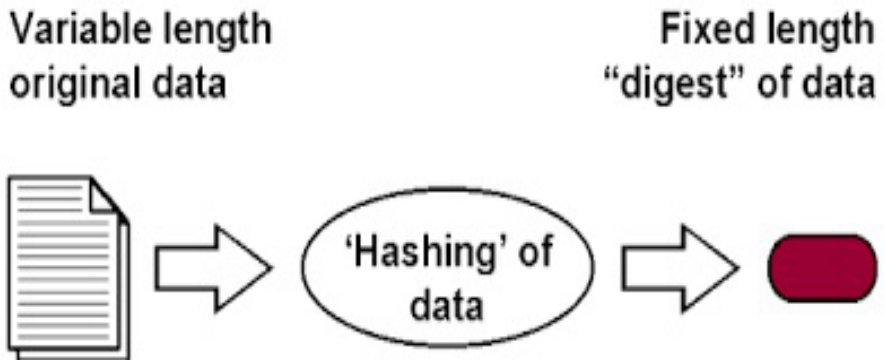
(1.-) $\mathcal{A}$ computes $\sigma = S_{\mathcal{A}}(M)$ and sends to $\mathcal{B}$ $C = (P_{\mathcal{B}}(\sigma))$,

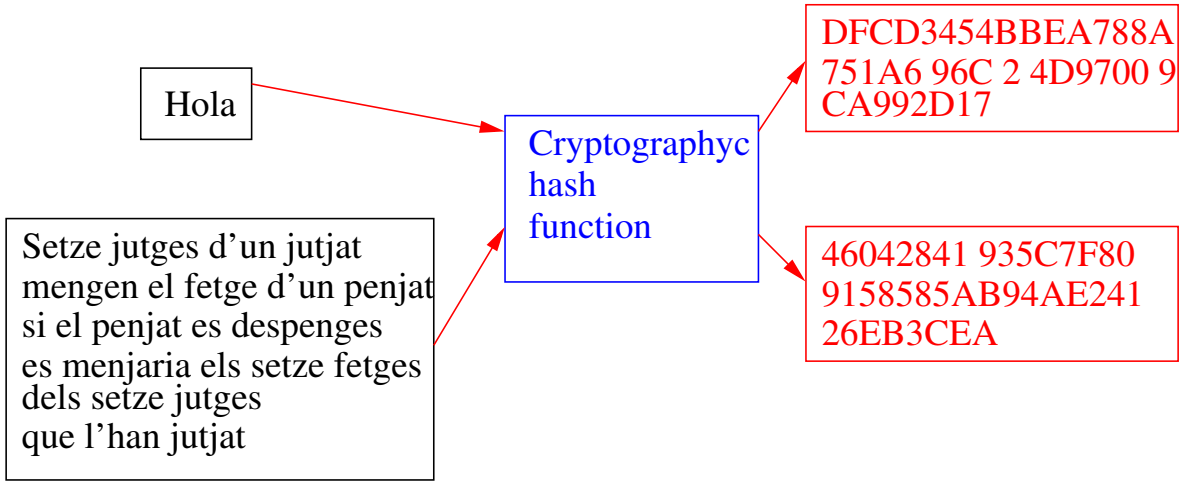
(2.-) $\mathcal{B}$ decrypts $M = S_{\mathcal{B}}(P_{\mathcal{A}}(C))$ (only $\mathcal{A}$ knows $S_{\mathcal{A}}$ so only $\mathcal{A}$ could compute σ)



Cryptographic hash functions

Cryptographic hash functions: One way hash functions





Cryptographic hash

Cryptographic hash functions have to string of any length and output a fixed-length hash value, in general in hexadecimal (radix 16).

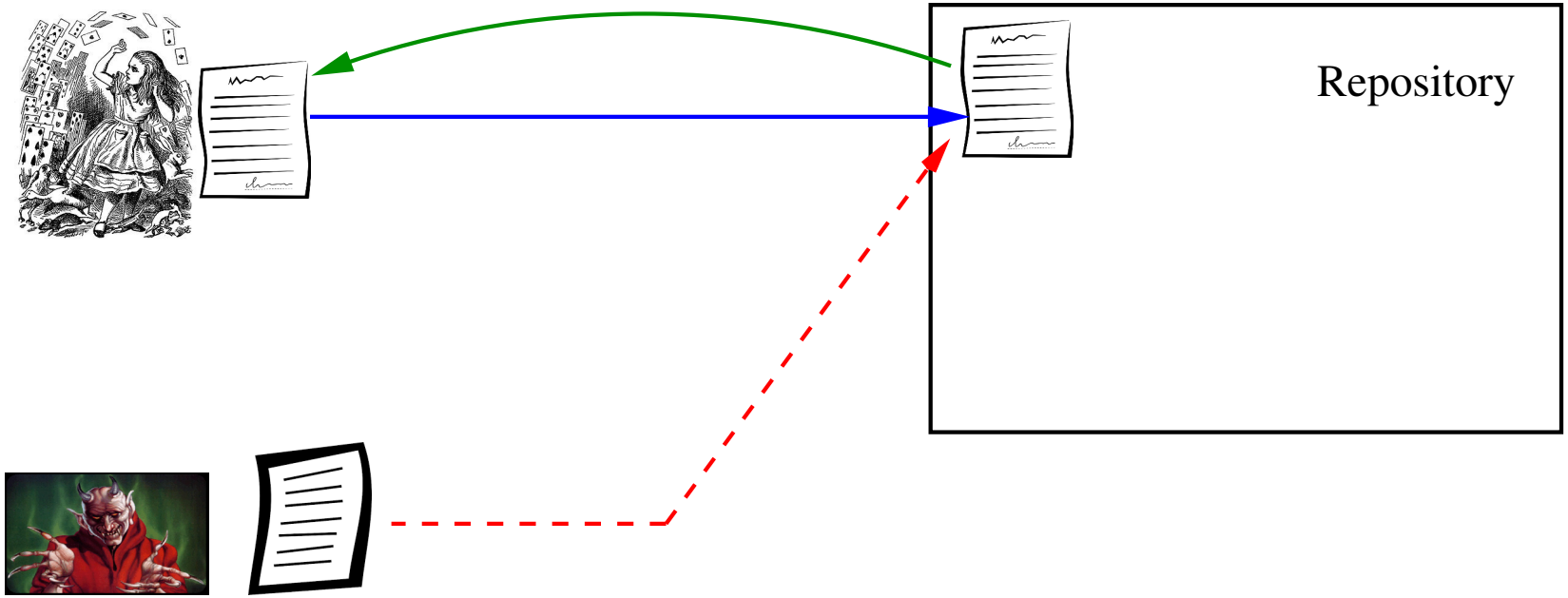
Hexadecimal= Radix 16
$(4CF5)_{16} = (4 \times 16^3 + 12 \times 16^2 + 15 \times 16^1 + 5 \times 16^0) = 19701$

For security reasons, modern crypto-hash implementations give yield very large integers, for instance SHA256 yields a 256 hexadecimal digits uoutput.

Binary	Hex	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

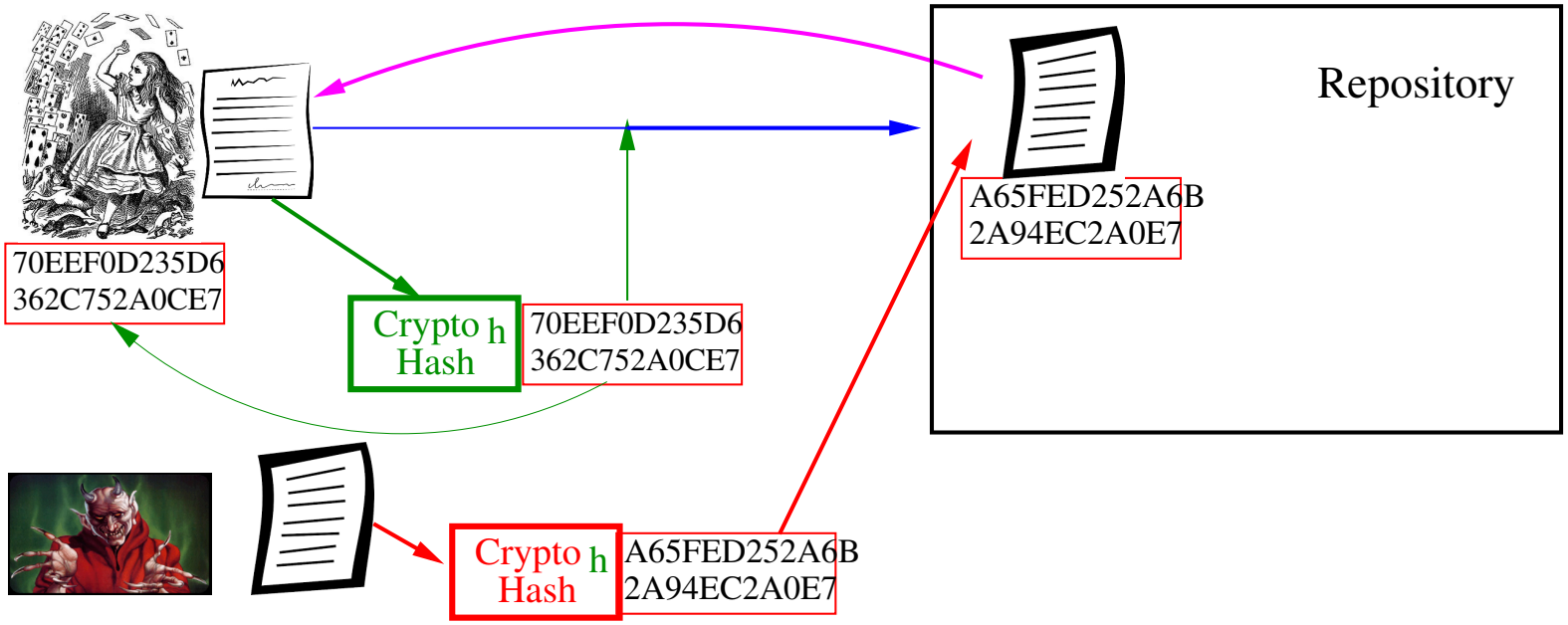
Applications: Message Digest

Alice wants to update a very large document in Dropbox like repository.
She wants to be sure when she download the document it is exactly the same document.
An adversary wants to substitute Alice's document for a forged one.



Applications: Message Digest

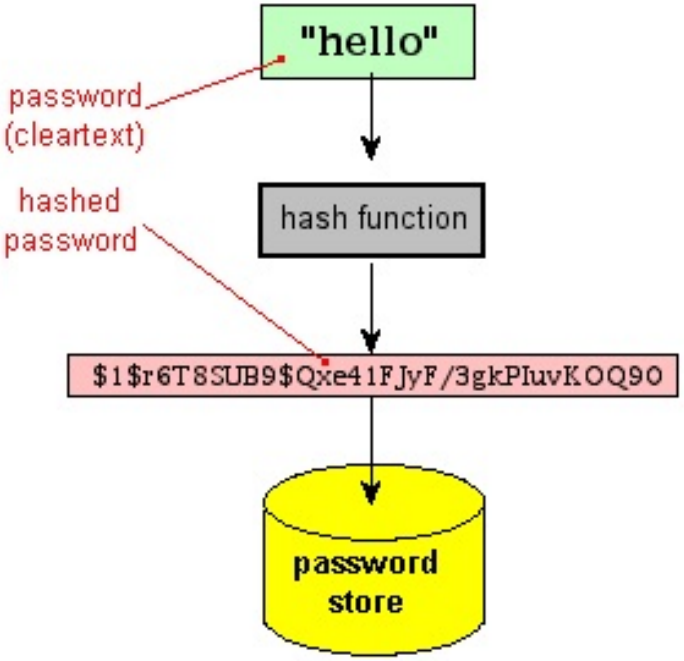
Alice appends a cryptohash h of the document to the stored document, and keeps a copy of the hash digest for her (very short).
The adversary has access to h but as soon as he tampers with the document the digest of the document will be different than the one append to the original document.
When Alice retrieves the document she just have to compare the digest of the document with the copy that she kept.

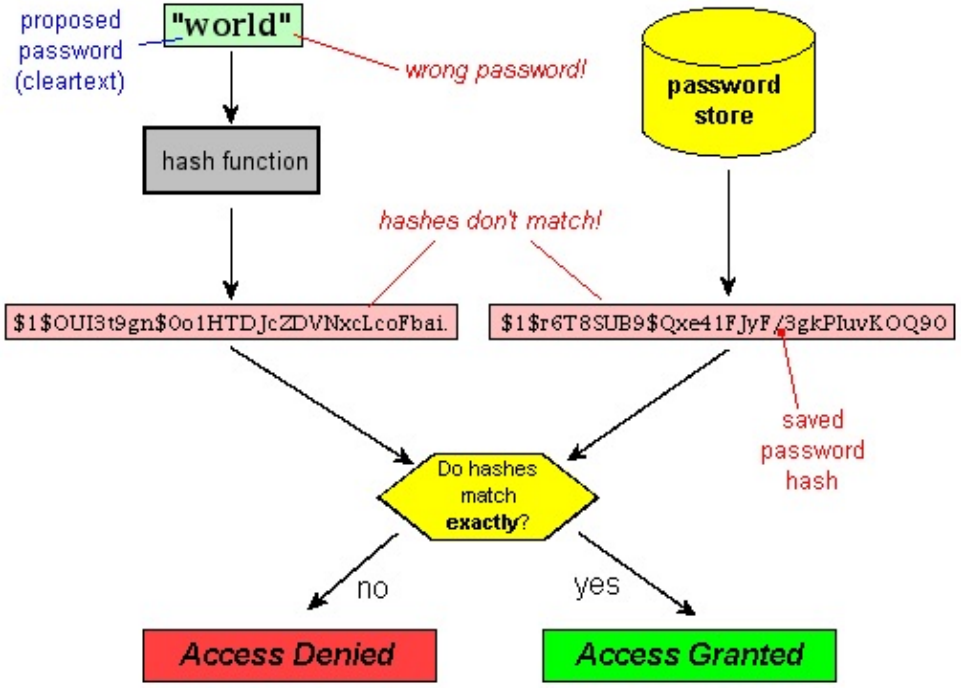


Applications: Password verification

Reduce security breach for passwords storing.

Store the hash digest on a table with users names



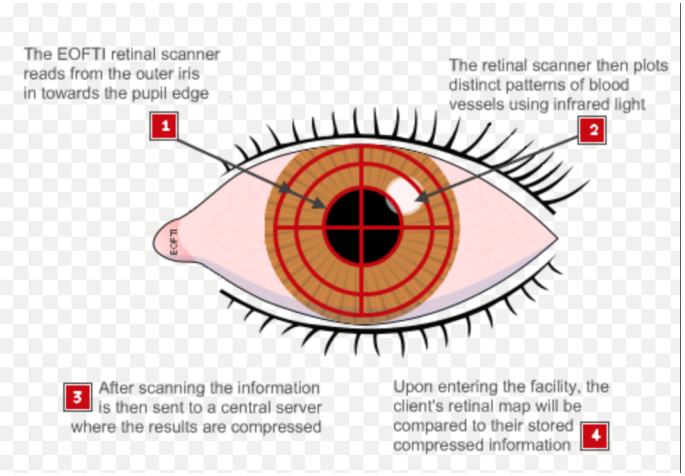


Applications: Password verification

There are Cryptography Hash functions to get a cryptographic integer from the personal biometrics data: fingerprint, retinal-scan, etc.

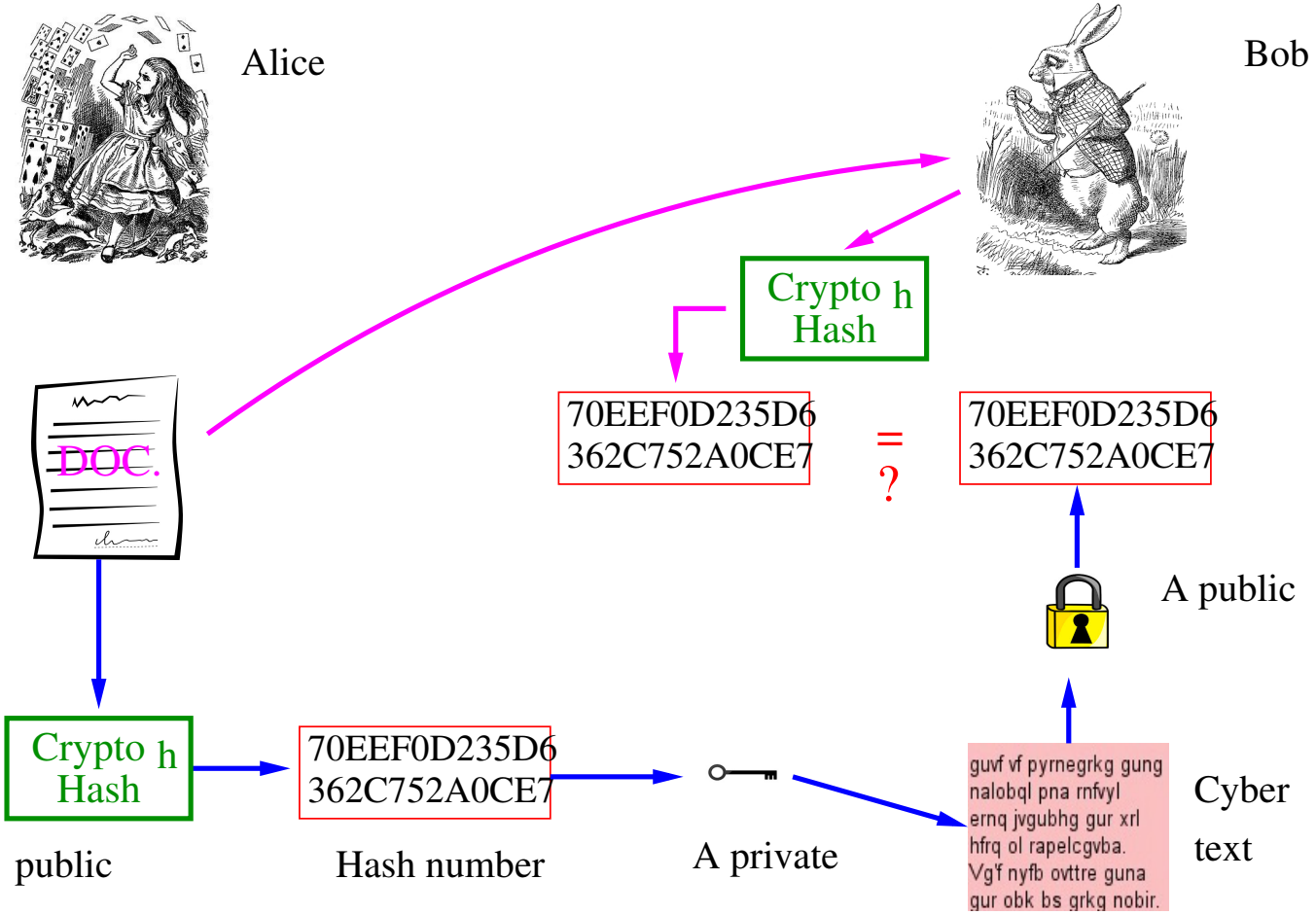
=

```
79054025
255fb1a2
6e4bc422
aef54eb4
```



Application: Digital signature and verifying the integrity of files or messages

Determining whether any changes have been made to a message or file. Confirm the sender is Alice.



Main Properties of a Cryptographic Hash Function h_c

A crypto-hash function is a digital fingerprint of the input: for any input length it always yields an output with of the same length.

Crypto-hash functions should behave randomly while being deterministic and efficiently computable.

1. For any message M, $h_c(M)$ is fast to compute.
2. Pre-image resistance: It is not feasible to recover M from $C = h_c(M)$. (Computing h_c^{-1} should not be feasible).
3. Collision resistance: Given a text M_1, it should not be feasible to find a M_2 s.t. $h_c(M_1) = h_c(M_2)$.
4. Sensitivity: If we slightly modify M to M' then $h_c(M) \neq h_c(M')$.

Example cryptographic hash function SHA-1

Home Page | SHA1 in JAVA | Secure password generator | Linux

SHA1 and other hash functions online generator

Aixo es un password | hash

sha-1

Result for sha1: 140456bfc911f9a95cb3579db52633ac5c5d9f97

Home Page | SHA1 in JAVA | Secure password generator | Linux

SHA1 and other hash functions online generator

Aixo es un Password | hash

sha-1

Result for sha1: 669e117b4662ce8e641a37a556cfb3848e7f9a59

SHA-1 MD5 on Wikipedia

https:www.sha1−online.com

A possible crypto function

Recall **o-exclusive** $\oplus$: $0 \oplus 1 = 1 \oplus 0 = 1$ and $0 \oplus 0 = 1 \oplus 1 = 0$. Given same length bit chains B_1, B_2, $B_1 \oplus B_2$ the bit to bit application of $\oplus$.

Define the following h_c: Given a binary message M (chain of bits),

1. Partition M into k blocks $M = B_1||B_2||\cdots||B_k$, where each $|B_i| = 5$. If $|M| \neq 5k$, we add 0's to the right.
2. $h_c(M) = B_1 \oplus B_2 \oplus \cdots \oplus B_k$.

Example: Given $M = 10101110$ to define $h_c(M)$,

1. convert $M \to M' = 1010111000$,
2. Break M' int $B_1 = 10101$ and $B_2 = 11000$,
3. $h_c(M) = B_1 \oplus B_2 = 01101$.

Is h_c a good crypto-function?

A possible crypto function

1. $h_c(x)$ easy to compute
2. (Pre-image resistance) Given an output x, $|\{h_c^{-1}(x)\}| = \binom{5!}{k}$, computing $h_c^{-1}(x)$ is not difficult but it could be time consuming,
3. (Collision resistance) NO, taking $M_1 = (10101010, 00001111)$ and $M_2 = (00001111, 10101010)$ then $h_c(M_1) = h_c(M_2)$,
4. (Sensitivity) One bit change in M changes only one bit in $h_c(M)$.

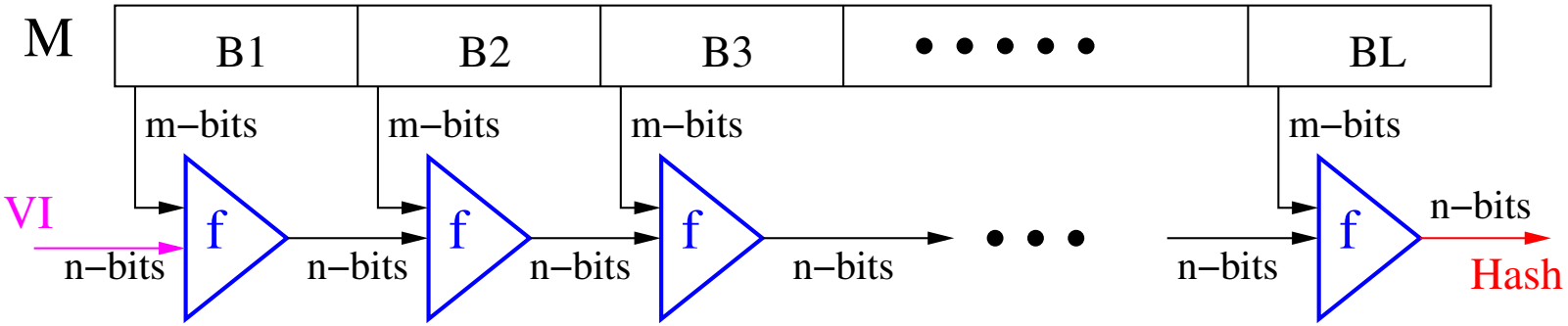
Therefore this h_c IS NOT a good crypto-hash function.

How to construct secure h_c: Merkel's scheme

Assume the M is the message that we want to compute its crypto hash function, and M is already in binary.

- The input message M is partitioned into L bit blocks B_i, each of size exactly m bits.
- For extra security, the ending block includes the total length of the message whose hash function is to be computed.
- The scheme has L sequential stages one for each block.
- The i-stage has as input the m bits from B_i and the n-bit output of the previous stage. The 1-stage is provided with a fixed n-bit input vector, the Initialization Vector (IV)
- The key is the compression function f, which depend on each implementation.

Merkel's scheme



The Secure Hash Algorithm (SHA) family

The Secure Hash Algorithm is a family of cryptographic hash functions developed by the *National Institute of Standards and Technology* (NIST) as a U.S.

Technical ideas based in previous work of several cryptographers: Ron Rivest, Ralf Merkel and others.

SHA-1 was designed by NIST in 1993. it is still the crypto hash of choice in many systems Microsoft, Google, Apple and Mozilla, they announced changes during 2017-19.

The Secure Hash Algorithm (SHA-1)

Given a message with size $< 2^{64}$-bits, SHA-1 produces an message digest (Hash output) exactly of 160-bits.

1. Pad M so its length K is such that $K \mod 512 = 448$. Append 64-bit with the length of M. Break the padded M into blocks of size 512.
2. Produce an initial vector of size 160-bits: 5 words of 32 bits each.
3. Process sequentially each block using the compression function f. Notice the input is an m bit and a n-bit and the output is an n-bit which feeds next stage.
4. The output of the last stage is the message digest, which has 160-bits.

Padding of the input M

The main steps of SHA-1 on input M

Let K = length of M. We want to break the input into equal 16 words (32bit) blocks.In total each B_i has 512-bits.

Append one 1-bit followed by Z 0-bits, where Z is the smallest non-negative solution to $K + 1 + Z \equiv 448 \mod 512$.

Add to the end 64 bits containing the binary representation of K. As the length field is 64 bits long, the longest M must be $< 2^{64}$ bits long.

Toy Example of Padding of the input M

Let $M = abc$, so $M = \underbrace{01100001}_{a}\underbrace{01100010}_{b}\underbrace{01100011}_{c}$

the whole padding to form a block:

$$\underbrace{\underbrace{01100001}_{a}\underbrace{01100010}_{b}\underbrace{01100011}_{c}\,1\,\underbrace{00\cdots0}_{423}\,\underbrace{00\cdots0\,\underbrace{11000}_{24}}_{64}}_{512}$$

Initial Vector

The initial vector (Hash Buffer) is given by the concatenation of 5 words, each one of 32 bits, named a, b, c, d, e. In total the IV has length 160 bits.

Each one of the 5 registers is initialized by the first 64 bits of the fractional parts of the square-roots of the first 5 primes (2,5,7,11,13):

The values are (in hex):
a = 67452301,
b = efcdab89,
c = 98badcfe,
d = 10325476,
e = c3d2e1f0.

Compression Function f

On input M, the SHA-1 algorithm breaks M into L blocks, each of size 512 bits.

Each block is partitioned in 16 words: $w_j[0]||w_j[1]||\cdots||w_j[15]$.

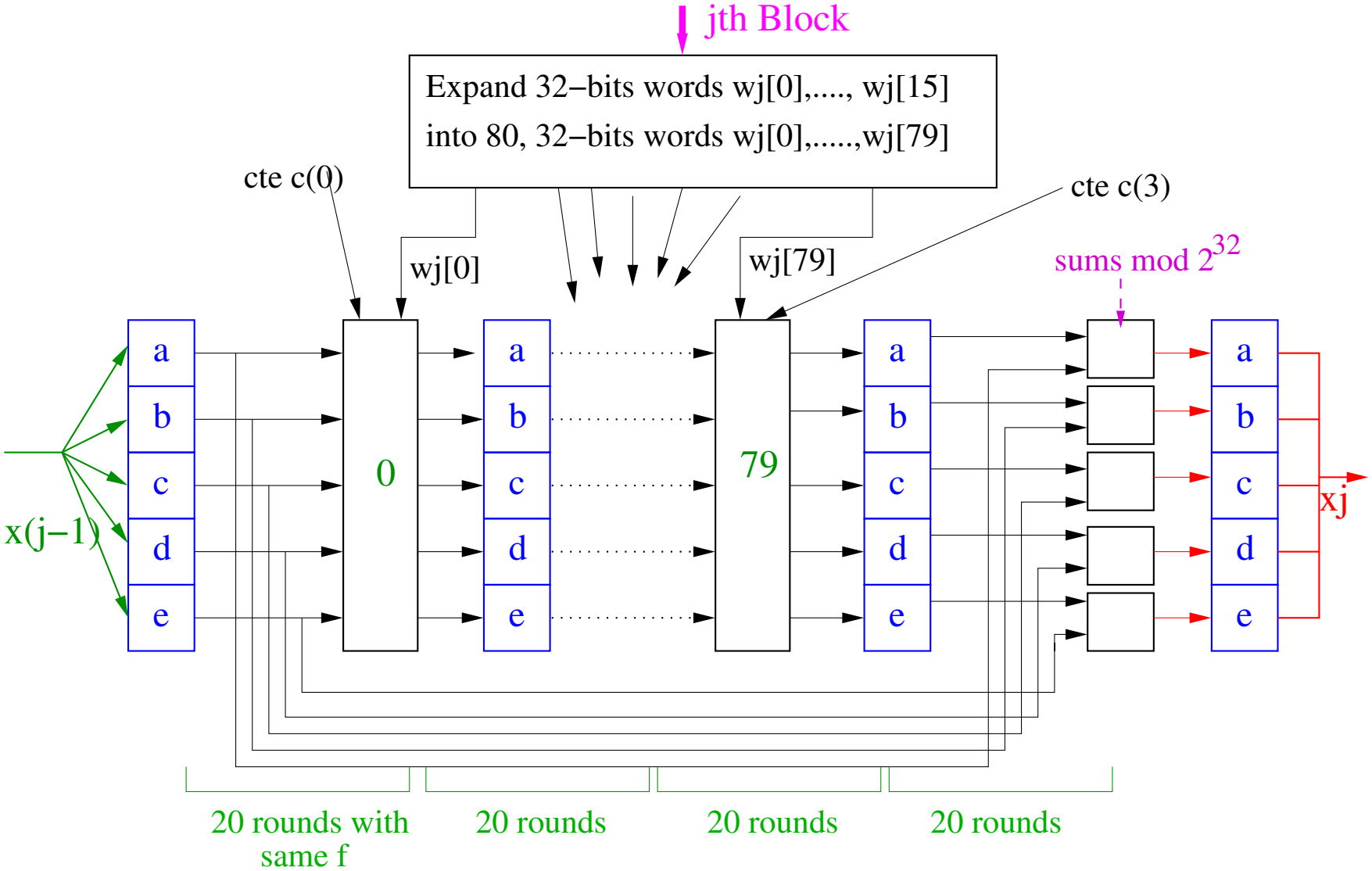
For each block there is a preprocessing phase that transform those 16 words into 80 words, creating new 63 words $w_j[0], \ldots, w_j[15], w_j[16], \ldots, w_j[79]$, by $w_j[i] = w[i-3] \oplus w_j[i-8] \oplus w_j[i-14] \oplus w_j[1-16]$, for$(16 \leq i \leq 79)$.

By feeding those w_j's as inputs to different stages of f, it will assure that al bits in B_j play a relevant role in the output of f for block B_j.

The compression function for each 512-bit B_j works in 80 rounds. Doing binary operations so $f(M)$ comply with the 3 properties of crypto hash functions.

The output to the last block is the message digest.

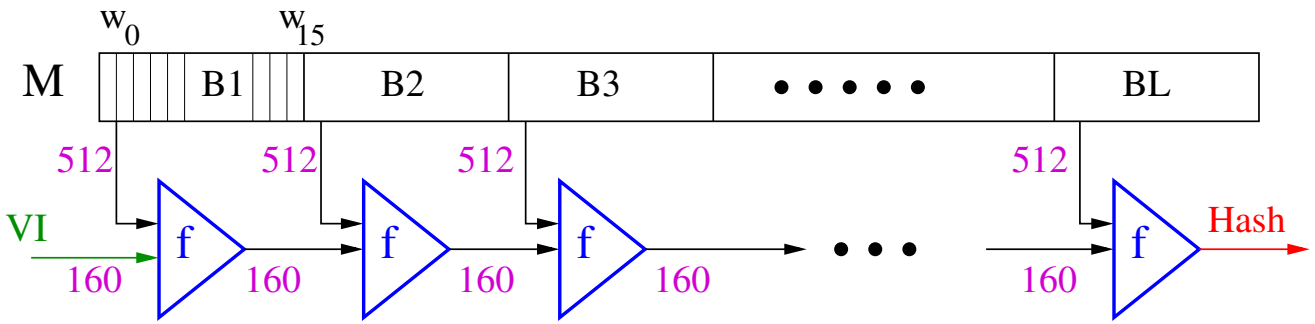
High level picture of f for B_j



80 Rounds of the Compression Function

Notice a,b,c,d,e are 32 bit words

Wrapping up



Given an input M the SHA-1 yields a 160 bit crypto hash of M in hexadecimal by:

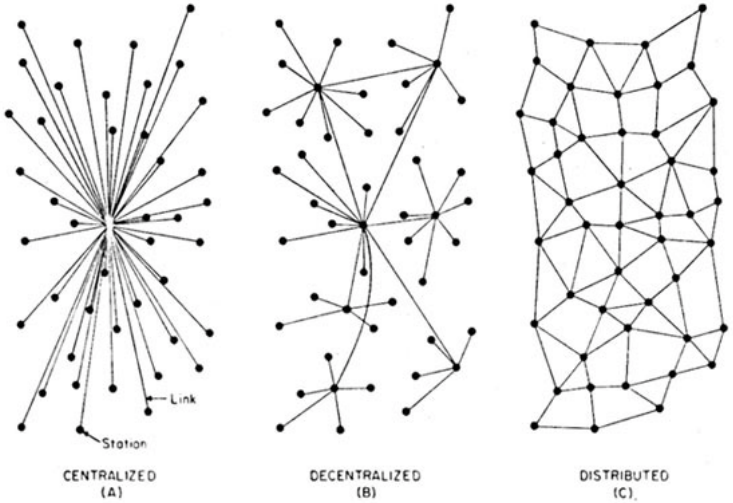
1. Padding M as a binary string multiple of 512. Partition into L blocs of size 512 bits.
2. Computing in cascade fashion the Compression Function for each B_j, it takes as input the 5 words hash buffer, from B_{j-1} and also the B_j itself, and returns the new values $a||b||c||d||e$, with total length 160 bits, which will be part of the input for the computation on B_{j+1}.
3. The output for the last block is the message digest, i.e. the crypto hash function for M.

Security of the SHA family

- At the moment SHA-1 is the crypto hash algorithm of choice in a myriad of systems, for ex. the browsers of Microsoft and Mozilla.
- Recent theoretical results have indication the possibility of breaking SHA-1,in particular the collision resistance property.
- One of the most clear choices at the moment is the SHA-512. The basic scheme is the same than the SHA-1, but uses a larger block size (1024 bits), can process documents up to 2^{128} and the output is exactly 512. For a few years, SHA-512 seem it will be secure choice.
- Another recent alternative is the SHA-256. It's the cryptohash for BitCoin.

Distributed Consensus

Distributed consensus protocol: A P2P network with n nodes each with an input, a few of those nodes are malicious or faulty.



A distributed consensus protocol must have the following two properties:

- It must terminate with all honest nodes in agreement,
- the value must have been generated by an honest node.

Distributed consensus with malicious nodes has been studied in the framework of classical distributed computing.
For some cases it is impossible to achieve consensus
The Byzantine General Problem.

Hash Pointer

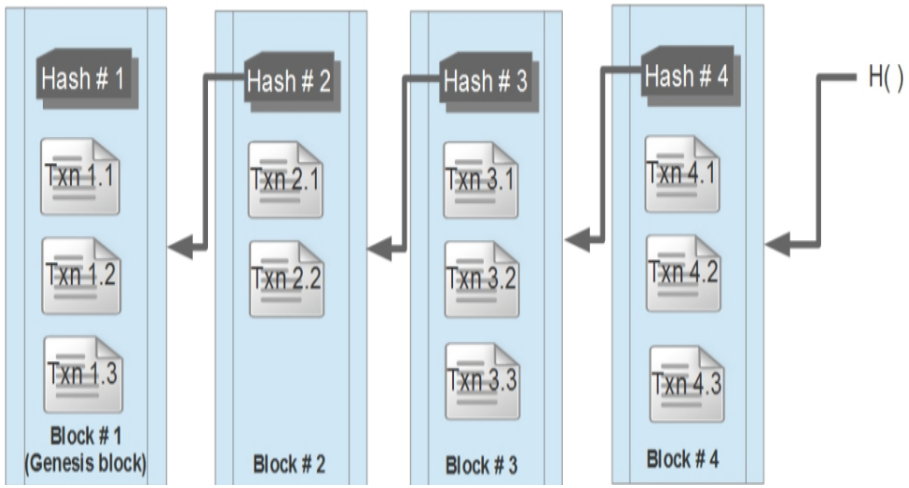
A block pointer is a data structure similar to the pointers in a linked list, but each pointer besides the address of the previous block, the pointer also contains a cryptographic hash of the information (or a part of it) contained in the previous block.

Whereas a "regular" pointer gives a way to retrieve the information, a hash pointer also gives you a way to verify that the information has not changed.

Notice we can use any know type of data structure, for ex. lists or trees, and and substitute the pointers by hash pointers.
Main hash-based data structures: Blockchains and Merkel Trees.

Blockchain

Blockchain: linked list data structure where the links are hash pointers.



Uses a crypto hash function H (usually a SHA−256)

Notice the digest H() conatained in block m are the cryptogram of the data contained in block m−1

The contains of a block include its hash pointer to the previous block

We can build a blockchain as large as we want, going back to a first initial block denoted the genesis block.

Blockchain: Impossibility of tampering the data

An adversary can't tamper data in any block of the chain without getting detected.

If Eve wants modifies one file in block m, the hash of this block (in Block $m+1$) invalid.

Therefore Eve has to modify also that hash, which in turn changes the contents of block $m+1$, and will become different of the H in block $m+2$.

This goes all way until the root hash pointer $H()$ to the head block, which is difficult to change as every user of the blockchain has their own copy of the root hash pointer.

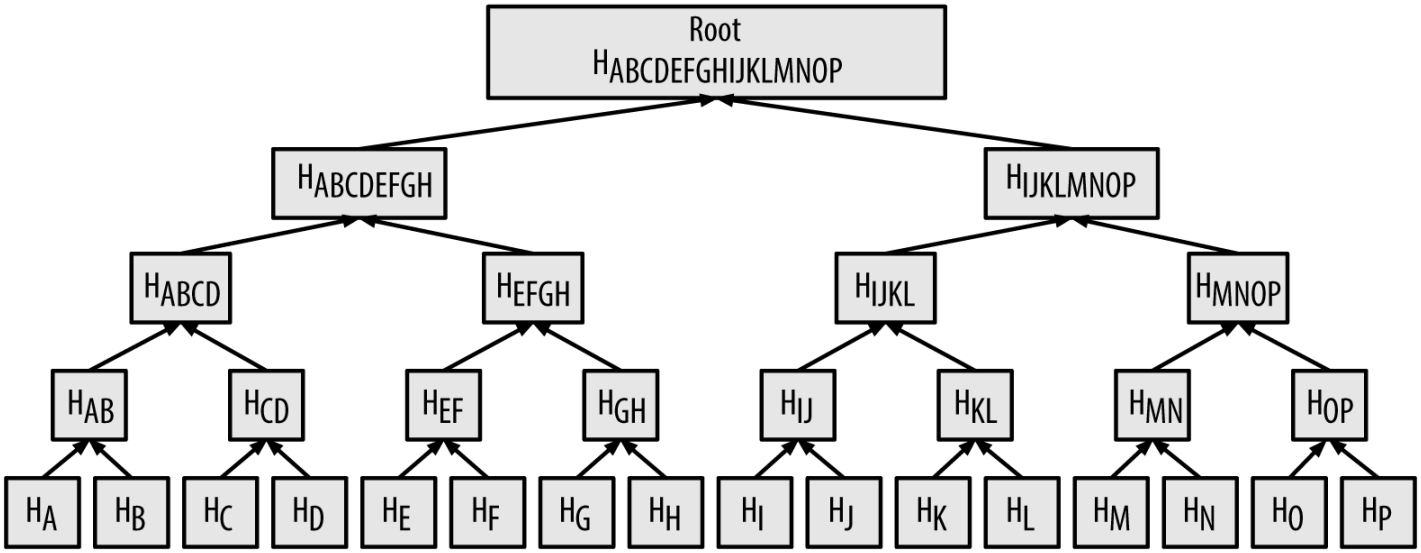
Every time a new block is added to the chain, the new $H()$ is broadcasted to all users.

Merkle trees

Another useful hash pointer data structure is the Markle tree. A Markle tree is a data structure used for efficiently verifying the integrity of large sets of data.

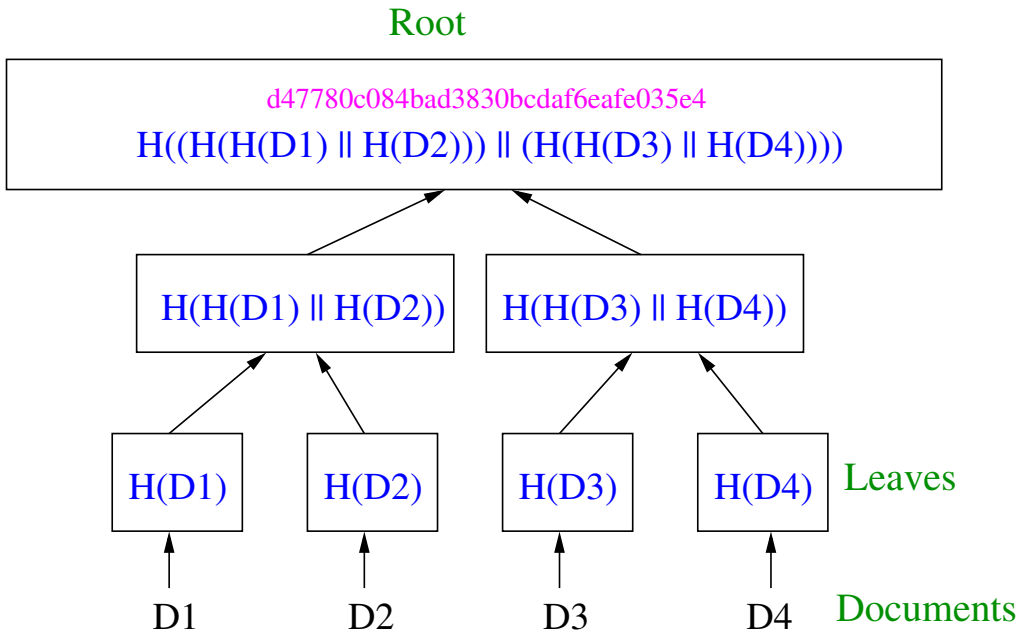
Usually H is the result of one or two words from applying SHA-256

It is a static DS: Given documents, $D_1, \ldots, D_m$, with $m = 2^k$, the Merkle tree will hash a large quantity of data into a single hash.



Construction Merkle's tree

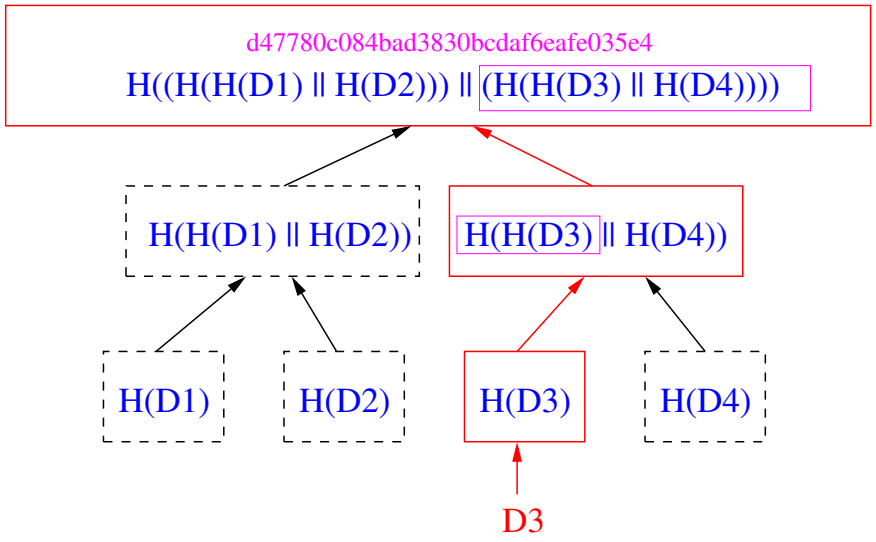
1. Give an ordered list $D_1, \ldots, D_m$, compute $h_c(D_1), \ldots, h_c(D_m)$ and store each hash in one of the m leaves
2. In a bottom-up fashion, at level h do the hash of the concatenation of the contains of pairwise blocks and store them in blocks at level $h-1$, until arriving to the root.
3. Constructing the tree from the items is fast.



Markle's trees as a tool for Data Verification

- The root is the hashing of hundreds of data items. From the root nobody can recover a specific D_i, as SHA-256 is one-way.
- To be sure that the data has not been tampered with, we have to check a given path from the fist hash of the data to the root of the tree, which has size $\lg n$.
- The same ideas can be applied for any kind of acyclic graph.

To check D_1 has not changed, we compute $H(D_1)$, compare with the leaf, compute the hash of the contains, compare with the prefix of the cell at level 1, compute the hash of its contains and compare with the suffix of the root.



Hash pointers Data Structures

Blockchain and Merkle trees are a new form of information technology that will have the relevance and importance comparable to the development of the TCP/IP internet protocol in 1974.

Blockchain could create a trustworthy and secure distributed ledger, without needs of a trusted third party.

Hash pointer technology has many applications:
Bitcoin,Smart contracts,Smart properties ...

Bitcoin: ฿

Satoshi Nakamoto, 2008. (Craig Wright?)

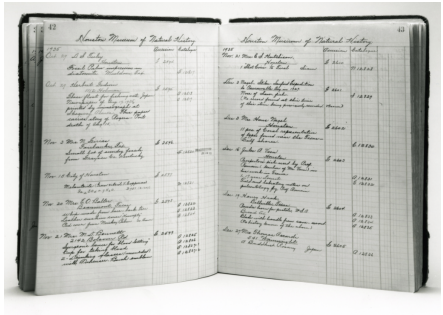
฿ is a entirely digital currency that exists only in electronic form, i.e. money created, controlled and stored in a trusted way form by a network of computers.

Why bitcoin has any value? because people want it and use it !

A bitcoin network is a P2P network of users, besides all users have access to a common blockchain that is a trustworthy distributed ledger of of who own what (at any time in history).

Bitcoin

1. Bitcoin network: P2P network, where nodes are the clients, together with a Bitcoin software. Each node identified by a wallet.
2. A Public ledger:
   The blockchain contains records of every single transaction of bitcoins. It replaces the bank's role (trusted third party).



3. A decentralized and distributed validation of transactions.
4. One way to generate ฿.

Distributed consensus in Bitcoin: Highlights

Bitcoin system has two different characteristics with deterministic models know to be have impossibility of achieving consensus:

1. Nodes have incentives to be truthful (get ฿),
2. it uses randomized protocols.

Bitcoin nodes have not a persistent, long-term identities.

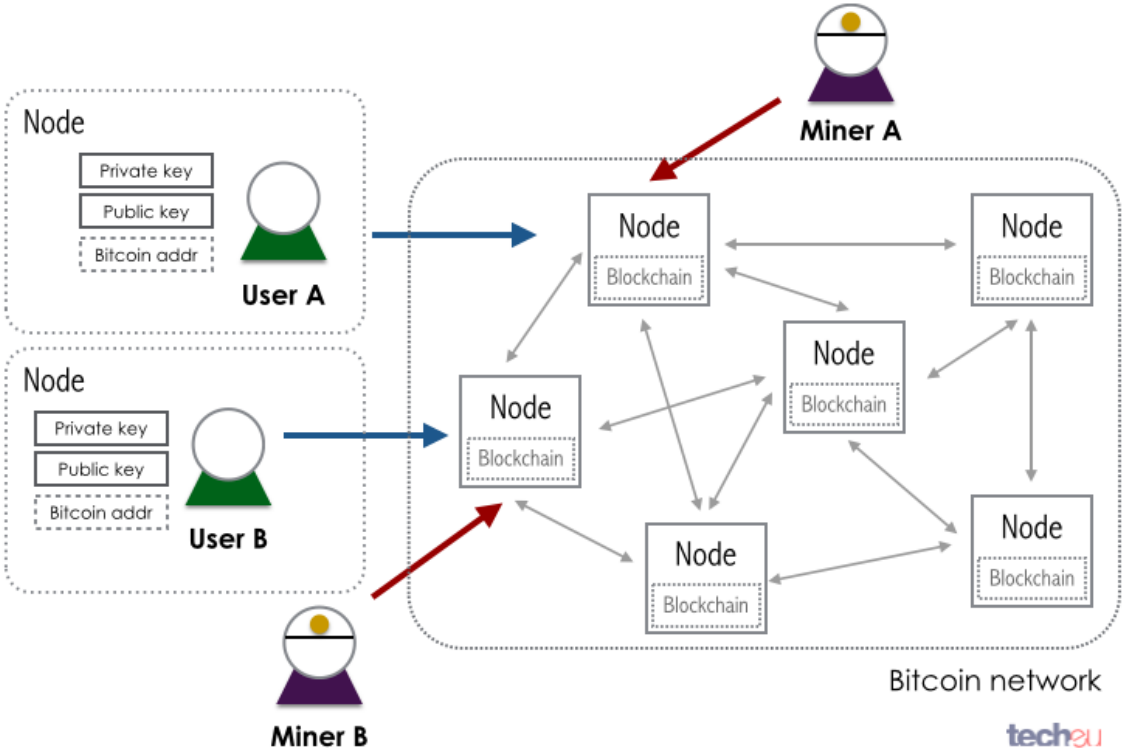
Outline of Bitcoin consensus algorithm:

- New transactions are broadcasted to all active nodes.
- Nodes validate transactions and gather them in a local transition pool.
- Approximately each 10 minutes (a round), one node constructs and broadcast to the network a new block containing hundreds of transactions. If verified by peers, the block becomes part of the blockchain.
- Other nodes accept a new block once they verify the whole block.

The Bitcoin network

- A P2P network of at most 10000 (active) nodes.
- Each node can implement functions among: routing, keeping the blockchain, verifying transactions, mining. They are defined by their wallets.
- The Bitcoin network runs over TCP. It is quite dynamical, at each instant only on-line nodes are considered.
- No geographic topology, the peers of a given node are not selected by proximity.
- The main purpose of the network is to maintain and verify the ledger of transactions (the blockchain)

The Bitcoin network

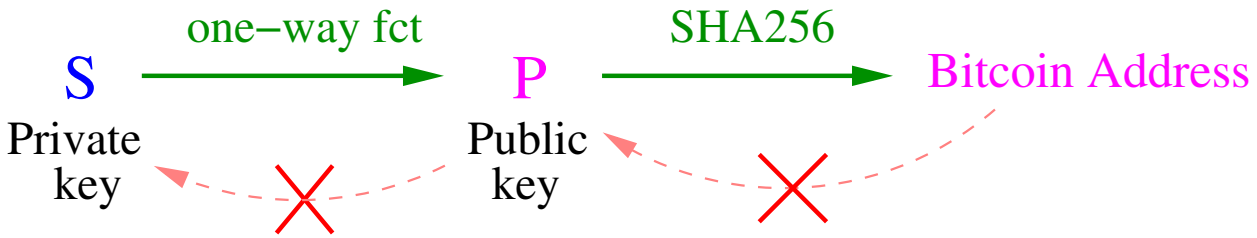


The network: Nodes

1. Full nodes, maintain a complete up-to-date copy of the blockchain (about 50Gb, and constant updates). Need to be active, most the time. Can maintain and verify the blockchain without using external information. Use Bitcoin Core system to build and manage their system (written in C++, Python, Java, etc.)
2. Lightweight nodes (SPV). Store an small part of the blockchain. Use Bit Core system. To verify transactions of blocks need to use information in full nodes. Use a system of verification denoted the *simplified payment verification* (SPV), which asks peers for the needed information.
3. Web clients store their wallets in a third party server. They have little control even in their own transactions.

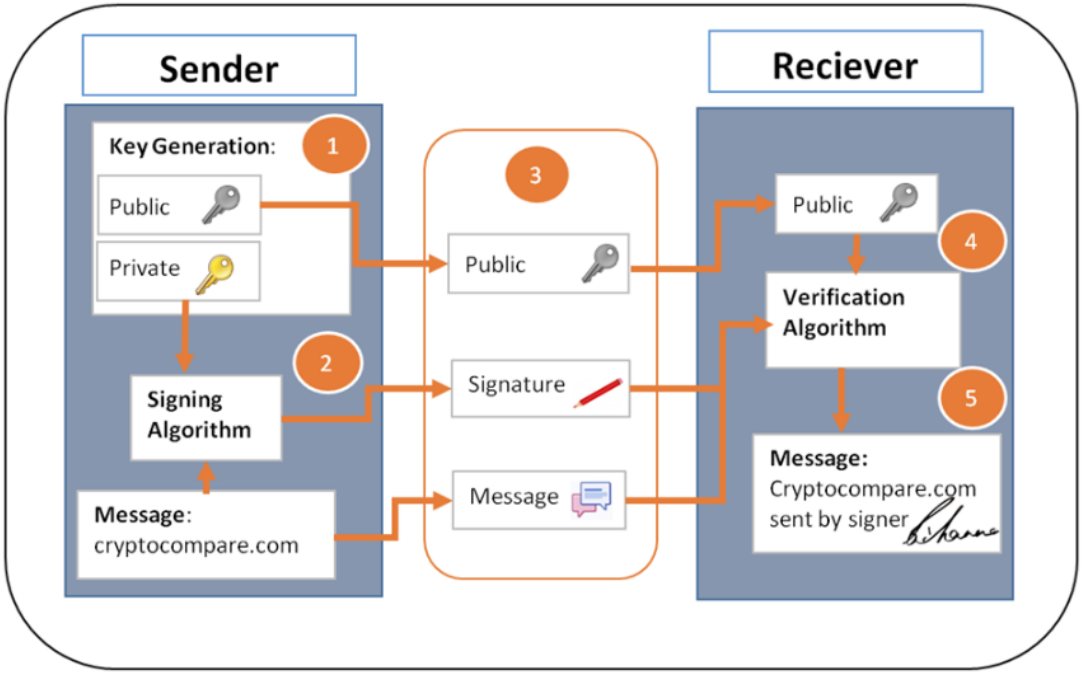
The wallet

- Bitcoin wallets store digital keys, not ฿
- The wallet has associated software to deal with transactions: wallet software, which allow people to manage bitcoins.
- Keys: private (a random integer), and from there we generate the public key and the bitcoin address.
- In the payment of a transaction the recipient identifier is represented by the bitcoin address, which is a digital fingerprint of the public key. Addresses are of single use.
- The keys use in Bitcoin is a system of decentralized identity management: no central register of users.



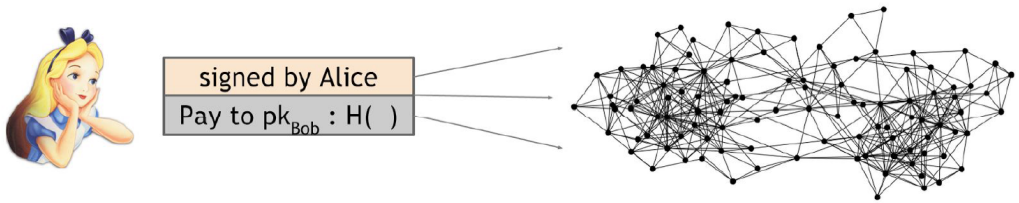
The wallet

- Wallet software can also generate a signature from the private key. The signature can be validated agains the public key.
- Digital keys, signature, address, change often to avoid detecting the real identity of the user



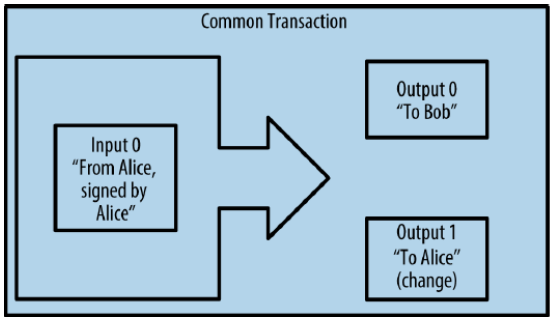
Transactions

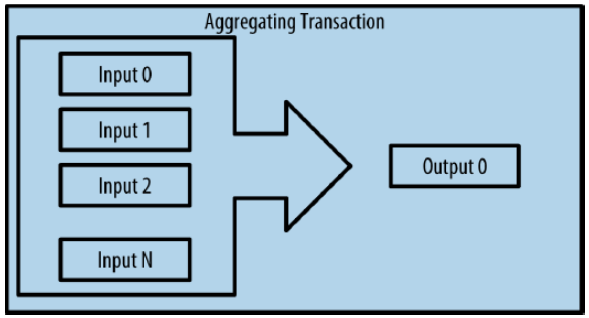
- Bitcoin fundamental building block. The whole idea of ฿ is to avoid account-based interchange of money, as it is algorithmically to costly to keep track of the book-keeping without a central authority.
- If $\mathcal{A}$ want to transfer ฿ to $\mathcal{B}$, a transaction is a data structure that authorizes $\mathcal{A}$ to transfer ฿ to $\mathcal{B}$'s public key, and verify that $\mathcal{A}$ got the amount of ฿.
- $\mathcal{A}$ sent transactions to the net and at some point arrives to $\mathcal{B}$.
- A transition consist in 300 to 400 bytes of data, that does not contain any confidential information.



Fundamental building block of transactions: UTXO

- Unexpended Transaction Output (UTXO). indivisible chucks of ฿ lock to an owner.
- In Bitcoin there are not accounts or balances, only unexpended UTXO scattered in the blockchain.
- In a transaction the input are pointers to existing UTXOs and the output are creating UTXO's.





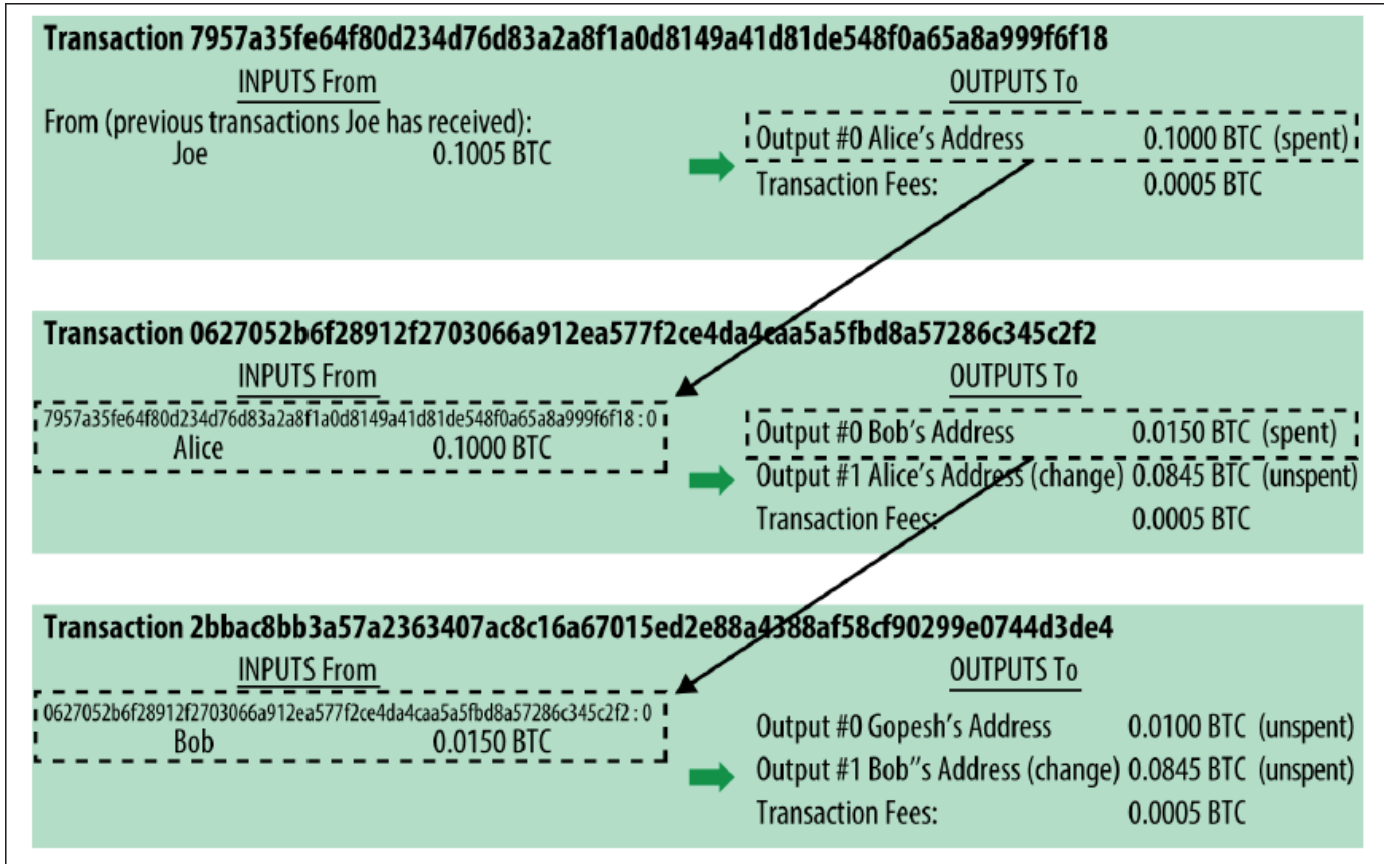
Transaction and UTXO

- Bitcoin uses a very limited script language to lock and unlock for inputs and outputs of a transaction.
- Locking scrip for output: indicates the conditions to spend the output, including the public key or address of the node receiving ฿.
  This can be redeemed by a public key hashing to X, along with the signature from the owner of that script
- Unlocking scrip for input: indicates the conditions to spend the output, including the public key or address of the node receiving ฿.

Script is a stack-based language without loops of complex flow-control why?. All information needed to execute an script is contained within the script.

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

Transaction and UTXO



Validation of transactions

- Every Bitcoin node will validate each transaction going through it, by executing its locking and unlocking scripts. Using the UTXO's pointed in the input, will go to the block where the transaction was stored, and much check that the ฿ have not being spent.
- Until a node does not validate a transaction, it would not continue the broadcasting of the transaction.
- Most nodes will maintain a temporary pool of verified transactions that have not been conformed by the inclusion of a new block in the blockchain (it could be quite large).
- Only a transaction validated by a majority of nodes results in the UTXO being marked as spent.

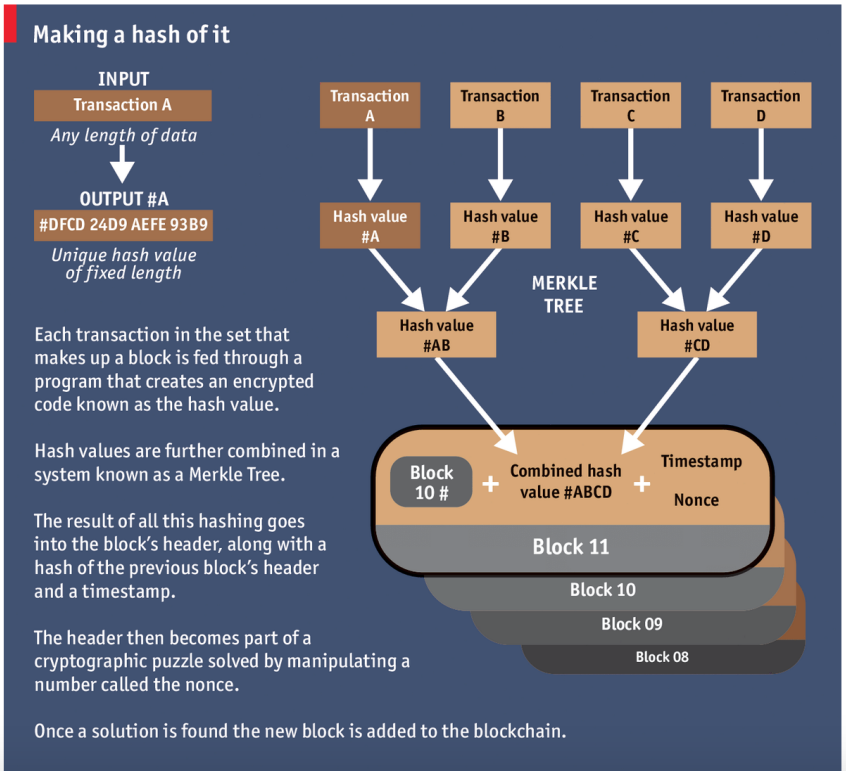
The Blockchain

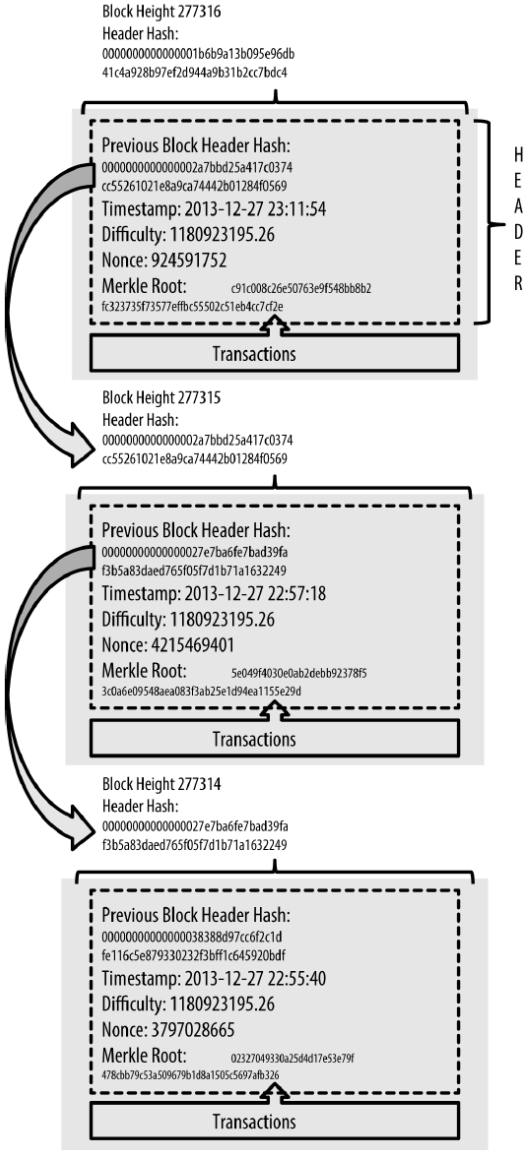
- The data structure implementing a distributed public ledger of ฿ transactions. Up to 4000 transactions can be included in a block.
- The block chain can be visualized as a vertical stack. From the genesis block 0, up to the last block introduced.
- Each block is composed of the block header and the body. The body contain the Merkle's tree of all the transactions being validated since the last block was added.
- Therefore, the Merkle's tree is used to verify the integrity of a given transaction in a block. Transitions are no part of the tree. The root is the output of a SHA256.
- The block header contains: hash-pointer to the previous block (the father); metadata as time of creation and the nonce (solution to the proof of work) and the root of the Merkle's tree.

The Blockchain

- Every block is uniquely identified by the hash of its block header
- The hash of the block header is not stored in the block (but nodes keep it in a data structure to speed up searches.
- Every full node contains the whole Blockchains
- SPV nodes maintain hashes of the block headers, which represent 1/1000 of the total size of the blockchain.
- There is a limitation of the size of a block to 1 Mbyte.
- The size of the blockchain was 448.000 blocks at the end of Jan. 2017.

Block in a Bitcoin blockchain





Mining

- The Bitcoin system of trust is bases on computation. This trust is done through mining.
- Mining is a secure mechanism forming the basis for the P2P digital cash. Mining together with the blockchain are the great inventions in the Bitcoin system.
- The mining process has 2 purposes:
  1. Create trust by validating and recording the transactions. A transaction is not valid until it becomes part of the blockchain.
  2. Creating new ฿.
- Any node using *Bitcoin Core Client* could be a miner. In practice, full nodes have the advantage.
- A block is confirmed every 10 minutes, it records hundreds of transactions.

Mining: Proof of work

- Miners compete to create the new block recording the transactions to be added to the blockchain. The one to win is the one that solves faster a puzzle called the proof of work.
- The one that does it earns and creates 25 ฿ (the miner also gets a few satoshis from each recorded transcription).
- To succeed, the minor has to solve a puzzle, the work of proof: Once the miner has constructed the block, the system provides a integer target t on every node, (the difficulty increases with the size of blockchain). The puzzle is to modify the block header until the hexadecimal result of its hash starts by more at least t 0's.
- The only way to do it is to use exhaustive search, in 2017 on average we need 150×10^{15} hashes to get the solution to the proof of work, which "certifies" the block.

Toy example of proof of work

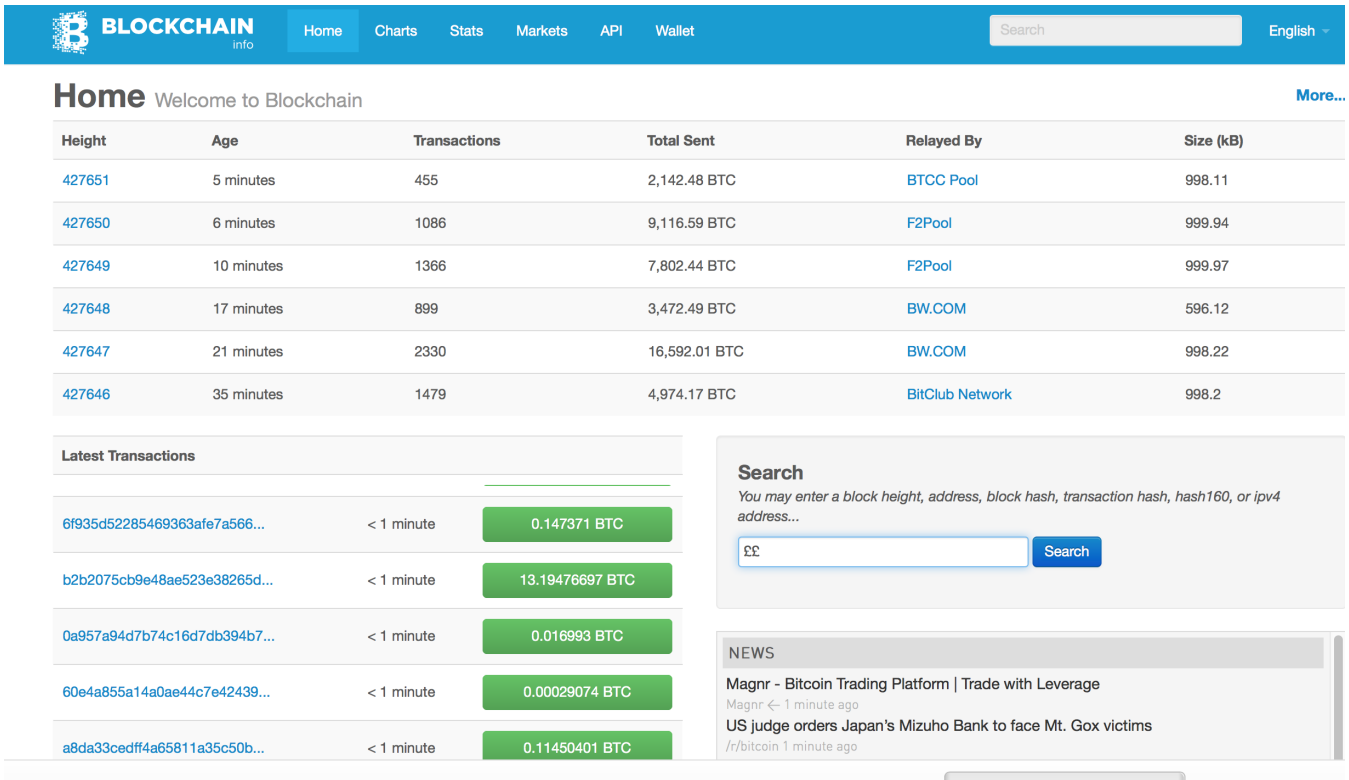
Assume the target value is $t = 1$,
we have a SHA-256 hash of I am Satoshi Nakamoto=
5d7c7ba21cbbcd75d14800b100252d5b428e5b1213d27c385bc141ca6b47989e
In this case the the nonce is 15:

```
I am Satoshi Nakamoto0 => a80a81401765c8eddee25df36728d732...
I am Satoshi Nakamoto1 => f7bc9a6304a4647bb41241a677b5345f...
I am Satoshi Nakamoto2 => ea758a8134b115298a1583ffb80ae629...
I am Satoshi Nakamoto3 => bfa9779618ff072c903d773de30c99bd...
I am Satoshi Nakamoto4 => bce8564de9a83c18c31944a66bde992f...
I am Satoshi Nakamoto5 => eb362c3cf3479be0a97a20163589038e...
I am Satoshi Nakamoto6 => 4a2fd48e3be420d0d28e202360cfbaba...
I am Satoshi Nakamoto7 => 790b5a1349a5f2b909bf74d0d166b17a...
I am Satoshi Nakamoto8 => 702c45e5b15aa54b625d68dd947f1597...
I am Satoshi Nakamoto9 => 7007cf7dd40f5e933cd89fff5b791ff0...
I am Satoshi Nakamoto10 => c2f38c81992f4614206a21537bd634a...
I am Satoshi Nakamoto11 => 7045da6ed8a914690f087690e1e8d66...
I am Satoshi Nakamoto12 => 60f01db30c1a0d4cbce2b4b22e88b9b...
I am Satoshi Nakamoto13 => 0ebc56d59a34f5082aaef3d66b37a66...
```

Consensus validation of a block

- As a new block is broadcast by the net, every full note validates the new block, before propagate it to its peers.
- The miner does not get its reward until a majority of nodes agree it is valid. This distributed validation ensures miners can cheat.
- At each full nodes the validation is done by the function `CheckBlock` of the Bitcoin Core client, and it consists in checking:
  1. Block is syntactically valid.
  2. The proof of work is correct.
  3. The time stamp is < 2 hours into the future.
  4. It has an acceptable size.
  5. The 1st. transaction is the reward to the miner.
- As blocks are created each approx. 10 minuter, the Bitcoin processes 7 transactions/minute. (VISA can handle 10000 transactions/second).

Farms and Blockchain Information web page



BLOCKCHAIN info | Home | Charts | Stats | Markets | API | Wallet | Search | English

Home Welcome to Blockchain

More...

Height	Age	Transactions	Total Sent	Relayed By	Size (kB)
427651	5 minutes	455	2,142.48 BTC	BTCC Pool	998.11
427650	6 minutes	1086	9,116.59 BTC	F2Pool	999.94
427649	10 minutes	1366	7,802.44 BTC	F2Pool	999.97
427648	17 minutes	899	3,472.49 BTC	BW.COM	596.12
427647	21 minutes	2330	16,592.01 BTC	BW.COM	998.22
427646	35 minutes	1479	4,974.17 BTC	BitClub Network	998.2

Latest Transactions

6f935d52285469363afe7a566...	< 1 minute	0.147371 BTC
b2b2075cb9e48ae523e38265d...	< 1 minute	13.19476697 BTC
0a957a94d7b74c16d7db394b7...	< 1 minute	0.016993 BTC
60e4a855a14a0ae44c7e42439...	< 1 minute	0.00029074 BTC
a8da33cedff4a65811a35c50b...	< 1 minute	0.11450401 BTC

Search

You may enter a block height, address, block hash, transaction hash, hash160, or ipv4 address...

££ Search

NEWS

Magnr - Bitcoin Trading Platform | Trade with Leverage

Magnr ← 1 minute ago

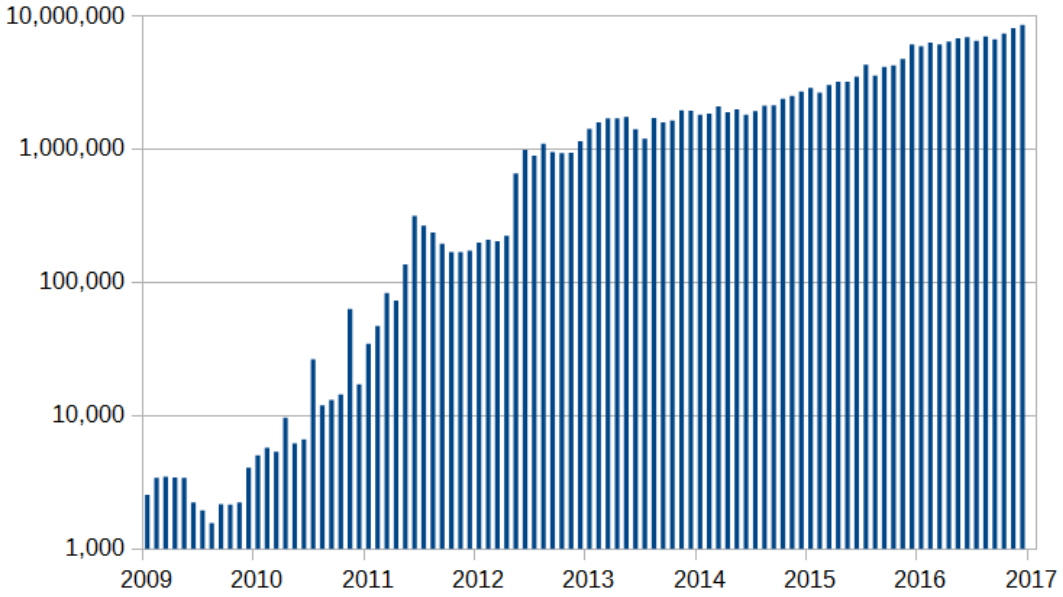
US judge orders Japan's Mizuho Bank to face Mt. Gox victims

/r/bitcoin 1 minute ago

Bitcoin: Summing up

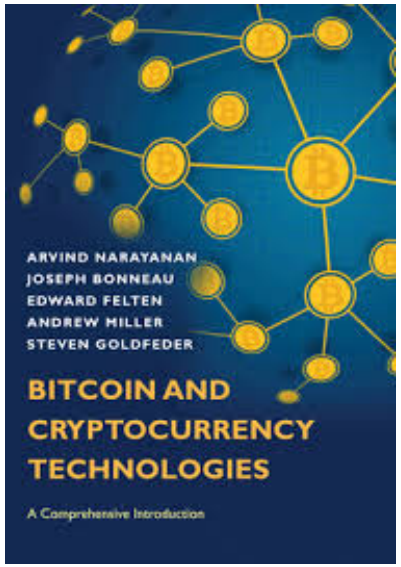
- Implements a digital currency, i.e. money created, validated and stored by a network of computers.
- Why bitcoin has any value? because people use it !
- Bitcoin is not just a currency, for illegal transactions, it is also a way to make payments without intermediators.
- Out control by banks and governments or any central authority.
- Useful for small payments (1 satoshi$= 1/10^8$ ฿). In many countries it is accepted in coffee shops
- The bitcoin has a great fluctuation (1฿= 516 €) (Sept 2016), (1฿= 1166.71 €) (March 2017).
- The total amount of currency in the system is bounded by $\leq 21 \times 10^9$ ฿(2040) . That may damage the ฿
- New alternatives exists; Ether: https://www.ethereum.org
  but actually the fact is:

Evolution of the number of transactions in Bitcoin



References

- 



- https://bitcoin.org/en/
- *The great chain of being sure about things* The Economist, Oct, 31, 2015
- *Bitcoin: A Peer-to-Peer Electronic Cash System* Satoshi Nakamoto.