

Funcions hash

Carlos Borrego

`Carlos.Borrego@uab.cat`

Departament d'Enginyeria de la Informació i de les Comunicacions
Universitat Autònoma de Barcelona

Criptografia i Seguretat

Material propi i adaptat de:

Material de classe de Criptografia i Seguretat

Dr. Guillermo Navarro

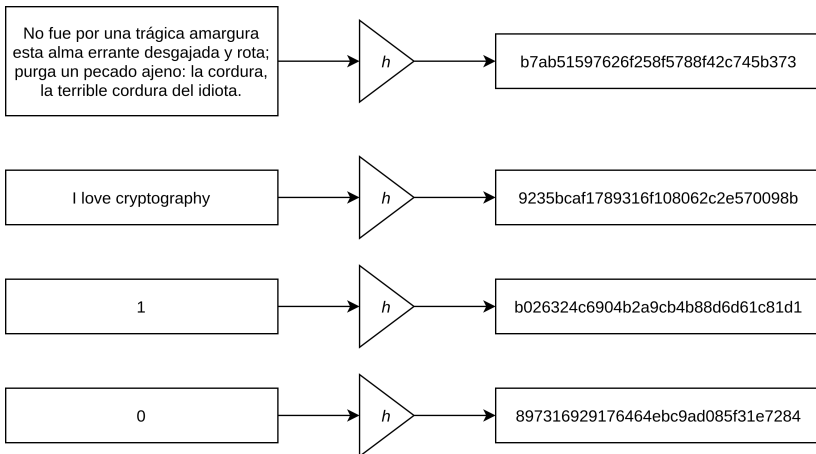
Universitat Autònoma de Barcelona

<http://www.deic.uab.cat/>

Contingut

- 1 Hash functions
- 2 Example application 1: Passwords
- 3 Example application 2: Bloom Filter
- 4 Example application 3: Blockchain
- 5 Example application 4: Merkle Trees
- 6 Example application 5: Codis d'autenticació de missatges

Hash function



Cryptographic hash function

- ① **Preimage resistance (or one-wayness)** Given a hash value z we cannot find a value x such that $z = h(x)$.
- ② **Second preimage resistance (or weak collision resistance)** Given a value x_1 we cannot find another value x_2 such that $x_2 \neq x_1$ and $h(x_1) = h(x_2)$
- ③ **Collision resistance (or strong collision resistance)** We cannot find two values x_1, x_2 ($x_1 \neq x_2$) such that $h(x_1) = h(x_2)$

Note: by *cannot find* we really mean *it is computationally unfeasible to find*.

Birthday paradox

How many students should be in a class in order for at least two of them to have the same birthday?

- We assume a year has (always) 365 days and birthday probability is equally distributed in all of them.

⇒ how many messages do we need to find a strong collision?

Birthday paradox

What is the probability of another person having the same birthday as you ?

Probability $p = 1/365$

How many people must be in a room so that the probability of at least another person having the same birthday as you is greater than 0.5 ?

$$\frac{364}{365}^n < 0.5 \Rightarrow n = 253 \text{ people}$$

How many people must be in a room so that the probability of at least two of them having the same birthday is greater than 0.5?

$$\frac{364}{365}^{n \cdot (n-1) / 2} < 0.5 \Rightarrow n = 23 \text{ people}$$

Cryptographic hash functions

Function	Versions	Output size (bits)	Construction	Year	Author	Broken
MD4		128	Merkle-Damgård Davies-Meyer	1990	Ronald Rivest	1995
MD5		128		1993	Ronald Rivest	2004
SHA-0		160		1993	NIST / NSA	1998
SHA-1		160		1995		2005
SHA-2	SHA-224	224		2004		
	SHA-256	256				
SHA-2	SHA-384	384		2001		
	SHA-512	512				
RIPEMD		128	Merkle-Damgård	1992	RIPE consortium	2004
	RIPEMD-128	128		1996	Hans Dobbertin Antoon Bosselaers Bart Preneel	
	RIPEMD-160	160				
	RIPEMD-256	256				
	RIPEMD-320	320				
SHA-3	SHA3-224	224	Sponge	2015	Guido Bertoni Joan Daemen Michaël Peeters Gilles Van Assche	
	SHA3-256	256				
	SHA3-384	384				
	SHA3-512	512				
Whirlpool		512	Merkle-Damgård Miyaguchi-Preneel	2000	Vincent Rijmen Paulo Barreto	

Contingut

- 1 Hash functions
- 2 Example application 1: Passwords**
- 3 Example application 2: Bloom Filter
- 4 Example application 3: Blockchain
- 5 Example application 4: Merkle Trees
- 6 Example application 5: Codis d'autenticació de missatges

Password-based authentication

Password

A secret data value, usually a character string, that is used as authentication information.



Stored password

- Passwords are NOT (normally) stored in clear text.
- Some schemes use **salt**, random data that is used as an additional input to the hash function.
- Normally using a one-way hash based function.
 - Not the actual hash! usually lots of iterations of the hash function.

Example 1: Linux passwords: format

/etc/passwd or /etc/shadow

\$<id>\$<salt>\$<hashed-pass>

- **id:** password hashing function:

ID	Method ¹
1	MD5
2a	Blowfish (not in mainline glibc; added in some Linux distributions)
5	SHA-256 (since glibc 2.7)
6	SHA-512 (since glibc 2.7)

- **salt**: up to 16 characters.
- **hashed-pass**: base64 encoding of the final hash.

¹ Recall: this is not a single hash, involves several iterations of the algorithm.

Example 1: Linux passwords: length I

Salt and hashed pass sizes:

id	salt	hash size
MD5	8 chars.	22 characters
SHA-256	16 chars.	43 characters
SHA-512	16 chars.	86 characters

Taula: Salt and hashed password sizes.

Examples

```
$1$12345678$t5YhsZQFmL.AAsc7PlYAd1
```

```
$5$123456789abcde$JIRjYzYmG7KX2qYFt1jGN390Nod2AzQN3A5/wJN2li9
```

```
$6$123456789abcde$kiMRFc4LhW/6RL8dzMi7FNj/C8B7oguQAHL/fEHG1v6j  
C46S4rPwBOzCVr192cbxaDa67.xN59Fta9oHOE4Pg1
```

Example 1: Linux passwords: sha512-crypt

sha512-crypt:

- 1 Initial hash of salt and password
- 2 Loop (Num of rounds from 1000 to 999999999, 5000 is the default.): new hash as the concatenation of the previous hash with the hash of the password and salt in several alternate ways.
- 3 Present the final hash string as special base64 encoding.

```
> mkpasswd -m sha-512 letmein 123456789abcde  
$6$123456789abcde$kiMRFc4LhW/6RL8dzMi7FNj/C8B7oguQAHL/fEHG1v6j  
C46S4rPwBOzCVr192cbxaDa67.xN59Fta9oHOE4Pg1
```

```
> mkpasswd -m sha-512 -R 1234 letmein 123456789abcde  
$6$rounds=1234$123456789abcde$Ch6wSMysLSENiPg/uTTwQWmaYbLc/Bt  
zef2JTMO.HaOAFdoKgiI77QZzoApwle0mOuRfTXDpjLU8Bw.MkjQJP1
```

Example 2: Windows passwords

SAM (Security Accounts Manager) file:

→ C:\Windows\System32\config\SAM or

→ HKEY_LOCAL_MACHINE\SAM



- Password hashing function
 - LM (LAN Manager) hash: based on DES
 - NTLM (NT LAN Manager) hash: based on MD4 (> Win Vista)
- No salt is used.

Dictionary attack

Dictionary attack

Guessing a password by repeated trial and error.

- Dictionary: list of words used for guesses.
- Ordered by (estimated) probability of success.
- Types:
 - with known hashed password (know ciphertext) → with a pre-computed dictionary.
 - without the hashed password → with cleartext dictionary.

Pre-computed Dictionaries

- Dictionary attacks → problem: **time!**

⇒ **Pre-computed dictionary:**

password	md5(password)
aaa	47bce5c74f589f4867dbd57e9ca9f808
bbb	08f8e0260c64418510cefb2b06eee5cd
ccc	9df62e693988eb4e1e1444ece0578579
...	...

→ Sort by hash.

→ Password guessing ⇒ table lookup.

Problem: **space!**

Problems and defenses

- Problem: There are publicly available rainbow tables: e.g. Ophcrack
- Possible Defenses:
 - **Salt**: large salt prevents pre-computation. Rainbow tables are unfeasible for large salt (48 ~ 128 bits).
 - **Key stretching**: increase time required to compute hashed password.

Password selection I

- Random selection.
- Pronounceable nonsense.
- Mnemonics.

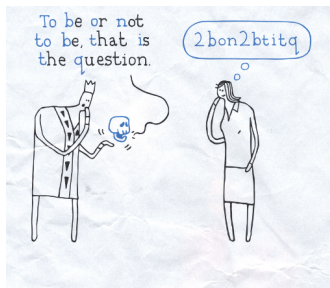
Password best practices I

Best practices (?)

- use common sense!
- consider usability in the recommendation:
“National survey reveals 58 percent of adults need to remember five or more unique online passwords; many people think solving world peace might be easier”

- it is not a bad idea to write down passwords!
- Mnemonics might not be that good!

Password best practices II



To be or not to be, that is the question.


2bon2btitq

Want to create a really strong password? Ask Hamlet.

Or Macbeth. Or Othello. Or even take a lyric from your favourite song. The more unusual the better. Try thinking of a memorable line like, 'To be, or not to be, that is the question' and then use numbers, symbols and mixed letters to recreate it: 2bon2btitq is a password with quadrillions of variations. Which is a lot.

In short, strong passwords can keep you safe online, which is good to know.

To find out more on how to be safer on the Internet go to google.co.uk/goodtoknow



2bon2btitq is actually NOT a good password

Password cracking software

Widely available:

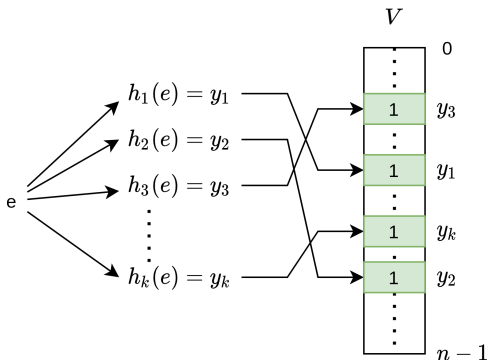
- John the Ripper (UNIXes, Kerberos, Windows, others ...)
<http://www.openwall.com/john/>
- Crack / cracklib (UNIX - Dictionaries)
<http://www.crypticide.com/alecm/software/crack/c50-faq.html>
- Cain and Abel (Windows - Rainbow tables)
<http://www.oxid.it/cain.html>
- L0phtCrack (Windows - not free)
<http://www.l0phtcrack.com/>
- Ophcrack (Windows - Rainbow tables)
<http://ophcrack.sourceforge.net/>
- Rainbow Crack (Generate Rainbow tables)
<http://project-rainbowcrack.com/>

Contingut

- 1 Hash functions
- 2 Example application 1: Passwords
- 3 Example application 2: Bloom Filter**
- 4 Example application 3: Blockchain
- 5 Example application 4: Merkle Trees
- 6 Example application 5: Codis d'autenticació de missatges

Bloom Filter I

- Vector binari V de mida n inicialitzat a 0.
- k funcions hash amb rang $[0, n - 1]$. $h_i : x \rightarrow [0, n - 1]$, $\forall i \in [1, k]$



Bloom Filter II

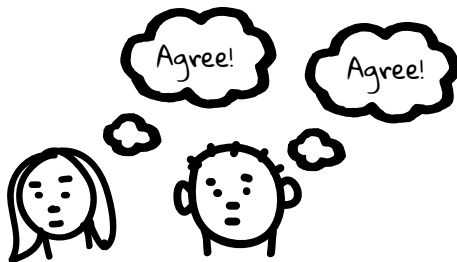
- És probabilístic:
 - Fals negatiu: NO és possible
 - Fals positiu: SÍ és possible, amb probabilitat que depèn de n i k .
- Tipus de funcions hash:
 - Específiques (famílies de funcions hash)
 - Particionar sortida d'una funció hash.
 - Ús d'una llavor.
- Existeixen diverses variants com els filtres de Bloom amb comptadors, que permet esborrar elements..

Contingut

- 1 Hash functions
- 2 Example application 1: Passwords
- 3 Example application 2: Bloom Filter
- 4 Example application 3: Blockchain**
- 5 Example application 4: Merkle Trees
- 6 Example application 5: Codis d'autenticació de missatges

Consensus

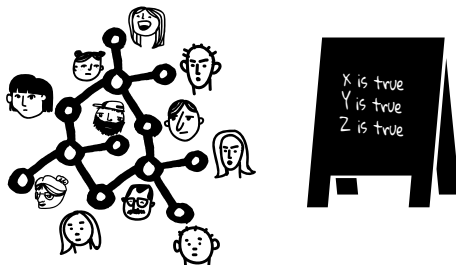
Consensus decision-making is a group decision-making process in which group members develop, and **agree** to support a decision in the best interest of the whole.



In distributed systems, a consensus mechanism is a fault-tolerant mechanism that is used to achieve the necessary **agreement** on a single data value or a single state of the network among distributed processes or multi-agent systems.

Distributed ledger as consensus tool

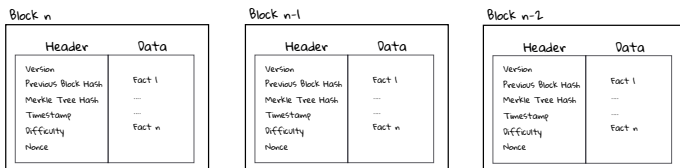
A distributed ledger is a consensus of replicated, shared, and synchronized **digital data** geographically spread across multiples entities without central administrator.



Read and write **operation rules** must be properly defined.

Blockchains as a distributed Ledger implementation

A blockchain is a growing list of records, called blocks, that are **linked** together using cryptography. The blockchain data structure is very different to the world state because once written, it cannot be modified; it is immutable.

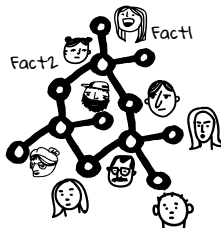
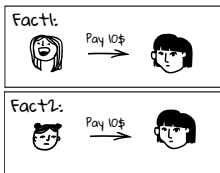


- **Previous Block Hash:** hash of the previous block that forms the chain.
- **Timestamp:** time the block is closed.
- **Difficulty:** keeps the average time between closed blocks steady as the network's hash power changes.
- **Nonce:** value is adjusted by miners so that the hash of the block will be less than or equal to the current target of the network.
- **Merkle Tree Root:** A binary hash tree that helps to encode blockchain data more efficiently and securely (see following slide).

Blockchain mining

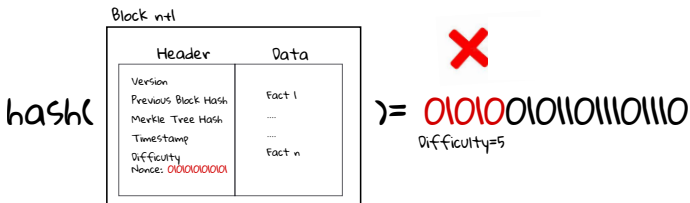
Blockchain mining is used to secure and verify **blockchain facts**. Mining involves Blockchain miners who add facts data to the blockchain global public ledger of past transactions.

Facts candidates:



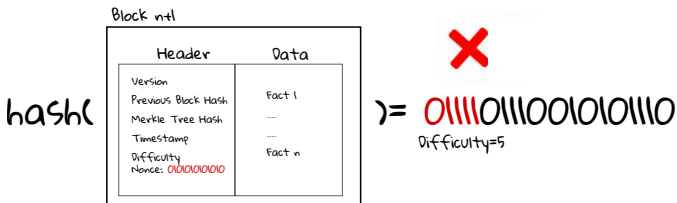
Blockchain hash

Proof of work is the godfather of blockchain consensus algorithms where validators (referred to as miners) hash the data they want to add until they produce a specific solution.



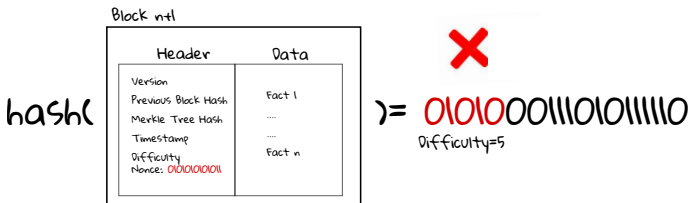
Blockchain hash

Proof of work is the godfather of blockchain consensus algorithms where validators (referred to as miners) hash the data they want to add until they produce a specific solution.



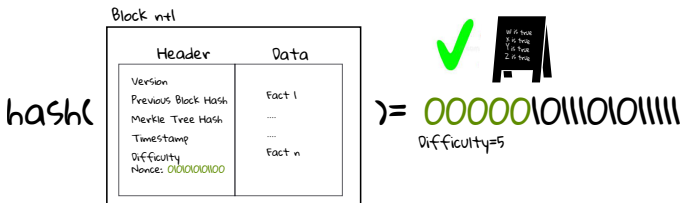
Blockchain hash

Proof of work is the godfather of blockchain consensus algorithms where validators (referred to as miners) hash the data they want to add until they produce a specific solution.



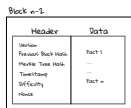
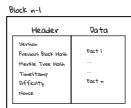
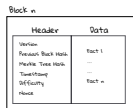
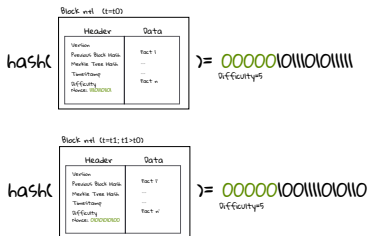
Blockchain hash

Proof of work is the godfather of blockchain consensus algorithms where validators (referred to as miners) hash the data they want to add until they produce a specific solution.



Blockchain forks

Blockchains are typically managed by a P2P network for use as a publicly distributed ledger, where nodes collectively adhere to a protocol to communicate and validate new blocks. Although blockchain records are not unalterable as forks are possible, blockchains may be considered secure by design and exemplify a distributed computing system with high **Byzantine** fault tolerance.



Reading:

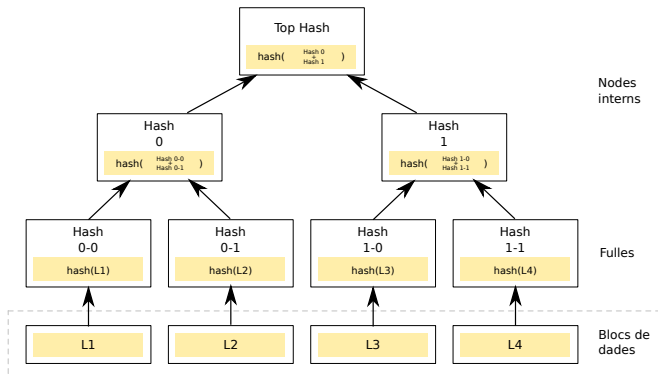
Yesterday, the Bitcoin network experienced one of the most serious hiccups that we have seen in the past four years. Starting from block 225430, the blockchain literally split into two, with one half of the network adding blocks to one version of the chain, and the other half adding to the other.

<https://bitcoinmagazine.com/technical/bitcoin-network-shaken-by-blockchain-fork-1363144448>

Contingut

- 1 Hash functions
- 2 Example application 1: Passwords
- 3 Example application 2: Bloom Filter
- 4 Example application 3: Blockchain
- 5 Example application 4: Merkle Trees**
- 6 Example application 5: Codis d'autenticació de missatges

Merkle Tree

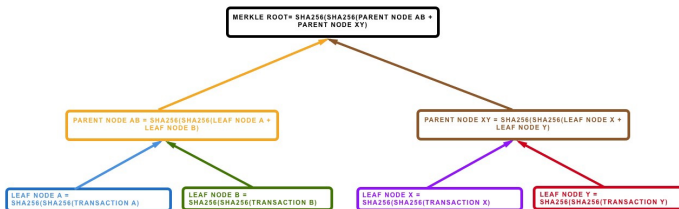


Caraterístiques I

- Arbre amb $\mathcal{L} = \{L_1, \dots, L_n\}$ elements i hash arrel h_r
 - Prova de pertinença per L_3 : $\Pi = (L_3, h_{11}, h_0)$
 - La prova ha d'incloure l'ordre o posició dels valors.
- + La prova conté $\log_2 n$ elements i la verificació requereix $\log_2 n + 1$
- + Resum del conjunt de dades h_r es petit i constant.
- + No es pot falsificar una prova de pertinença (si h es criptogràfica).
- No es pot provar la NO pertinença \Rightarrow arbres de Merkel ordenats.
 - Els elements estan ordenats: $L_1 < L_2 < \dots < L_n$.
 - Prova de que L_j NO hi és: Proves de pertinença de L_i i L_{i+1} , tal que $L_i < L_j < L_{i+1}$

Merkle Tree

A binary hash tree that helps to encode blockchain data more efficiently and securely. When mining, you only need to hash the block header, instead of the whole block.



Additionally, it speeds up the process of facts verification. Example: To verify $h(Y)$, if $h(X)$ is provided to us, we can work out $h(XY)$. Then, we need $h(AB)$ to calculate $h(ABXY)$, the Merkle Root. If it does, it is a proof that the transaction was included in the block.

Contingut

- 1 Hash functions
- 2 Example application 1: Passwords
- 3 Example application 2: Bloom Filter
- 4 Example application 3: Blockchain
- 5 Example application 4: Merkle Trees
- 6 Example application 5: Codis d'autenticació de missatges**

Codis d'autenticació de missatges

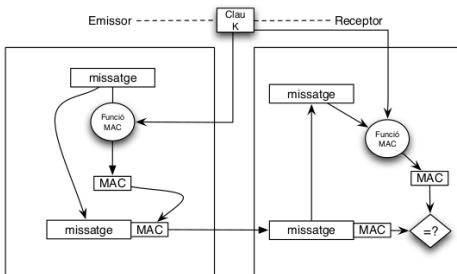
Un codi d'autenticació de missatge (MAC) és una cadena curta d'informació relacionada amb el propi missatge a través d'una clau de manera que permet la seva autenticació.

$$HMAC1_k(m) = h(k \parallel m)$$

$$HMAC2_k(m) = h(m \parallel k)$$

on el símbol \parallel representa la concatenació de cadenes. La primera expressió es coneix com a secret prefix HMAC i la segona com a secret suffix HMAC.

Codis d'autenticació de missatges



Funcions hash

Carlos Borrego

`Carlos.Borrego@uab.cat`

Departament d'Enginyeria de la Informació i de les Comunicacions
Universitat Autònoma de Barcelona

Criptografia i Seguretat