II LIVELLO DI ANALISI:
LA STRUTTURA DEL BITCOIN

Che cos'è fisicamente il Bitcoin e quali sono le caratteristiche essenziali?

Transazione senza intermediario

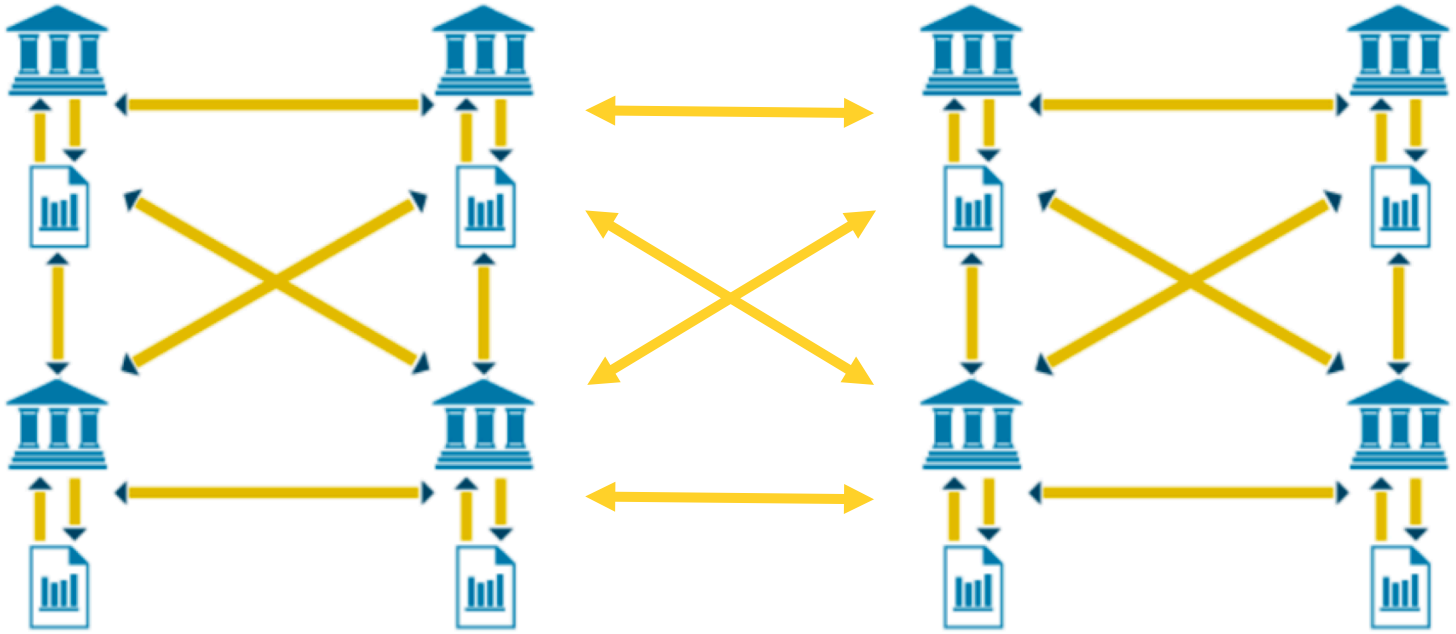
elementi costitutivi:

1. *Distributed ledger* (libro mastro distribuito)
2. *Blockchain* (catena dei blocchi)
3. nodi della rete o *miners* (i minatori) e la loro attività di *mining* (estrazione)

1 – Distributed ledger (libro mastro distribuito)

E’ una rete

che connette dei server che contengono ciascuno un libro mastro



Transazione senza intermediario:

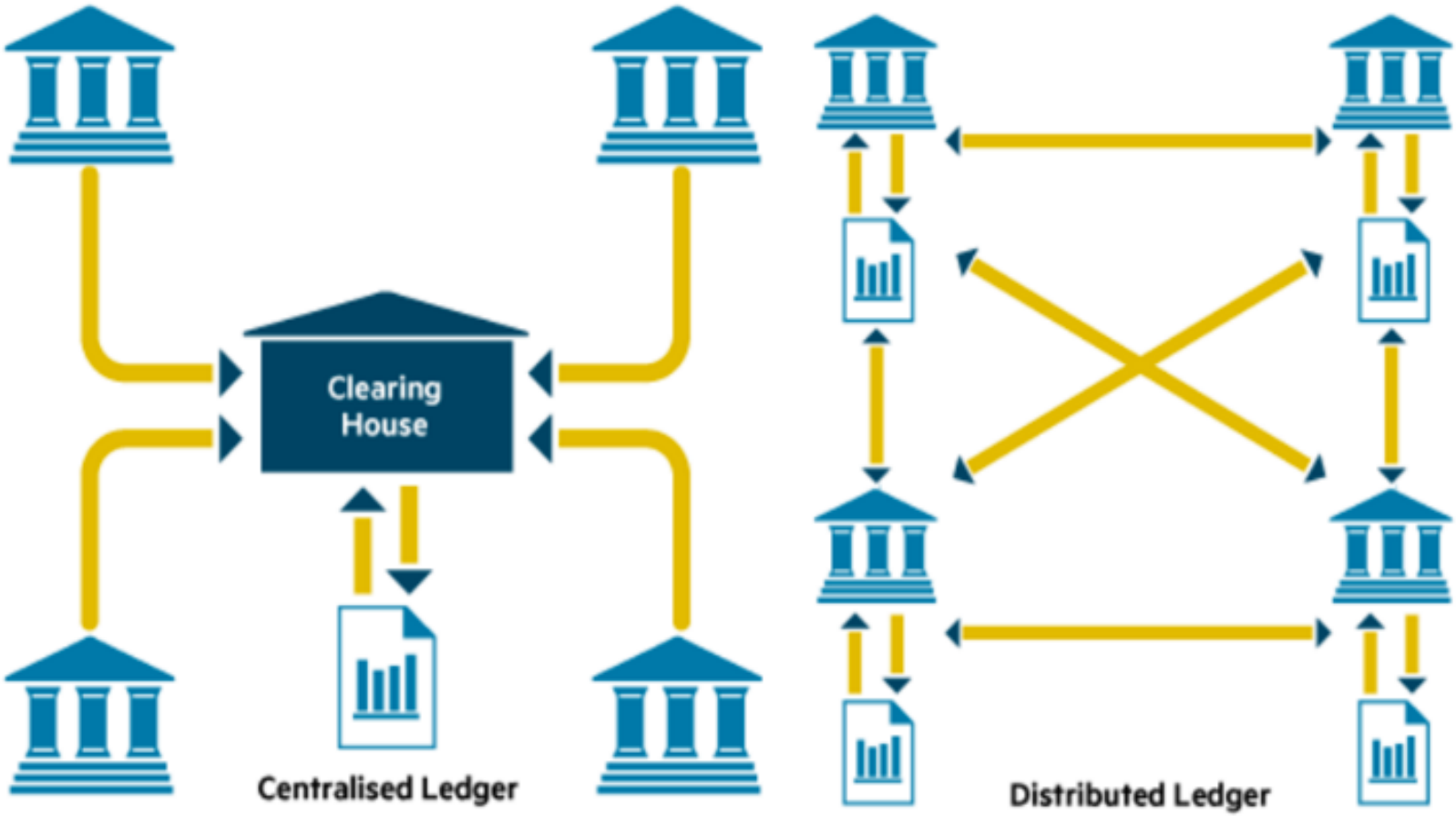
peer-to-peer (P2P) network

Server-based

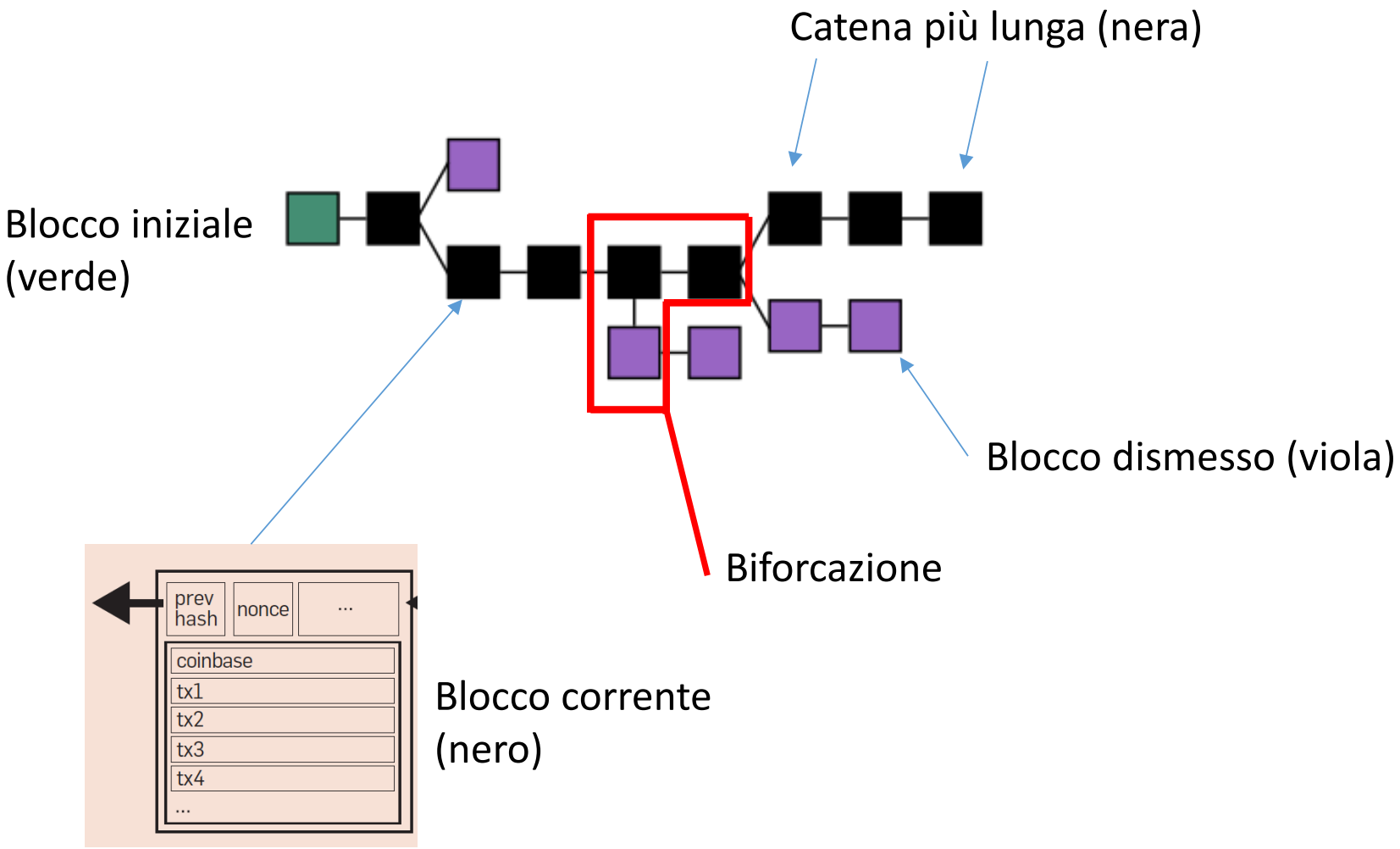
P2P-network

1 – Libro-mastro distribuito (*distributed ledger*)

Transazione senza intermediario: *data-base* distribuito

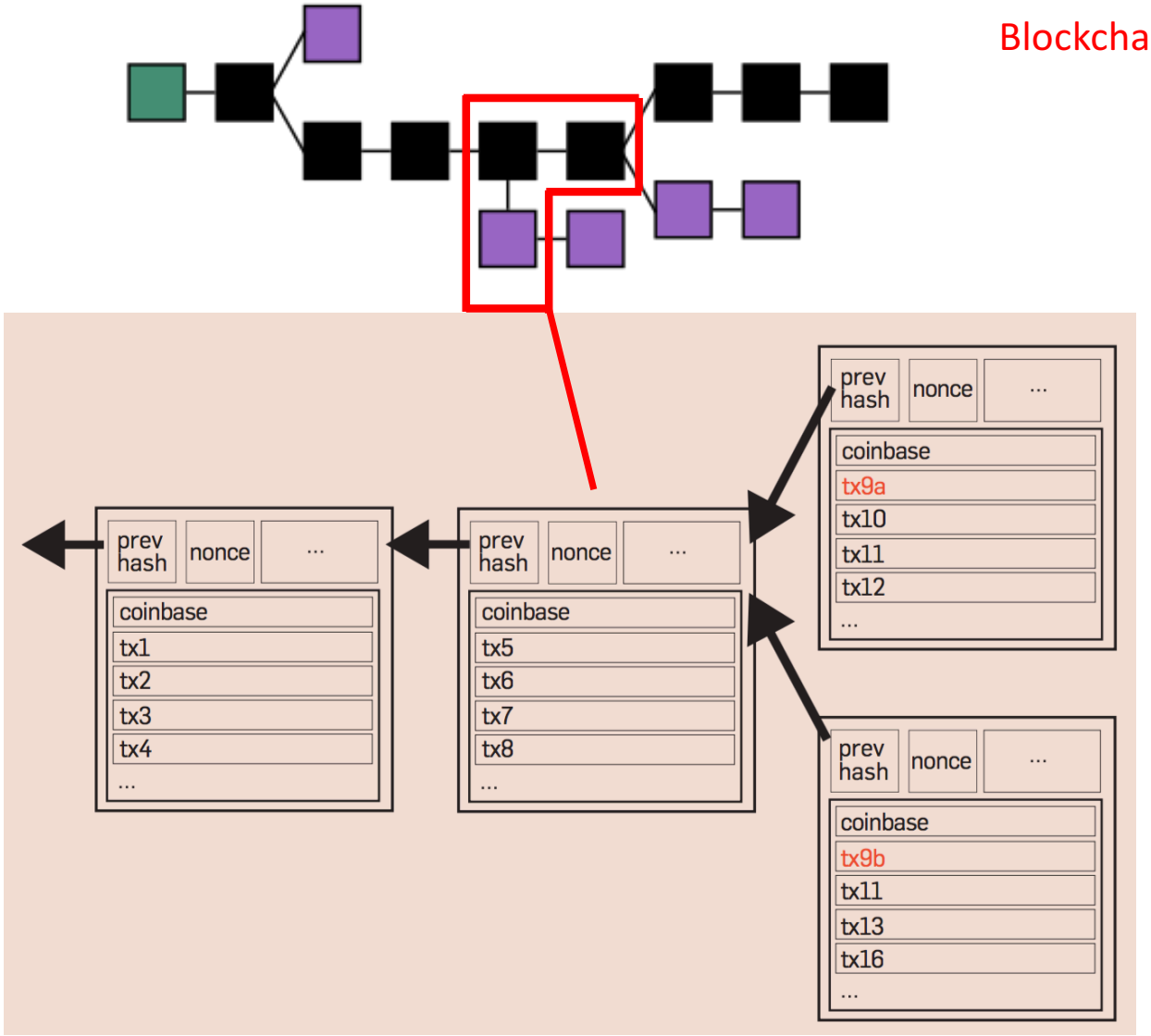


2 – Blockchain (catena dei blocchi)



LA STRUTTURA DEL BITCOIN

Blockchain

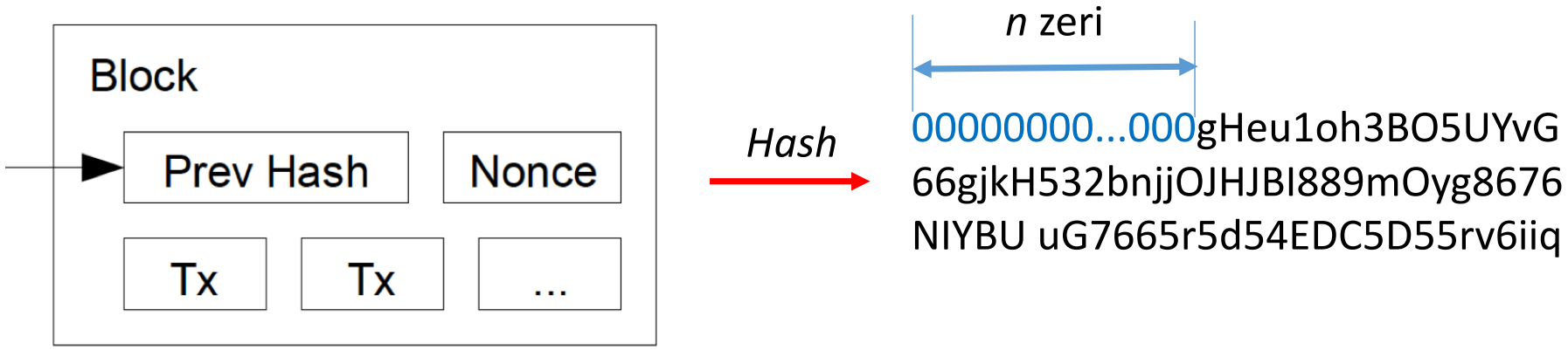


3 – Nodi della rete o *Miners* (minatori) e la loro attività di *mining*

Costituiscono i nodi della rete e la tengono in funzione
$\sim$ 10000 nodi attualmente per BTC (Bitcoin)
$\sim$ 2000 nodi attualmente per BCH (Bitcoin Cash)

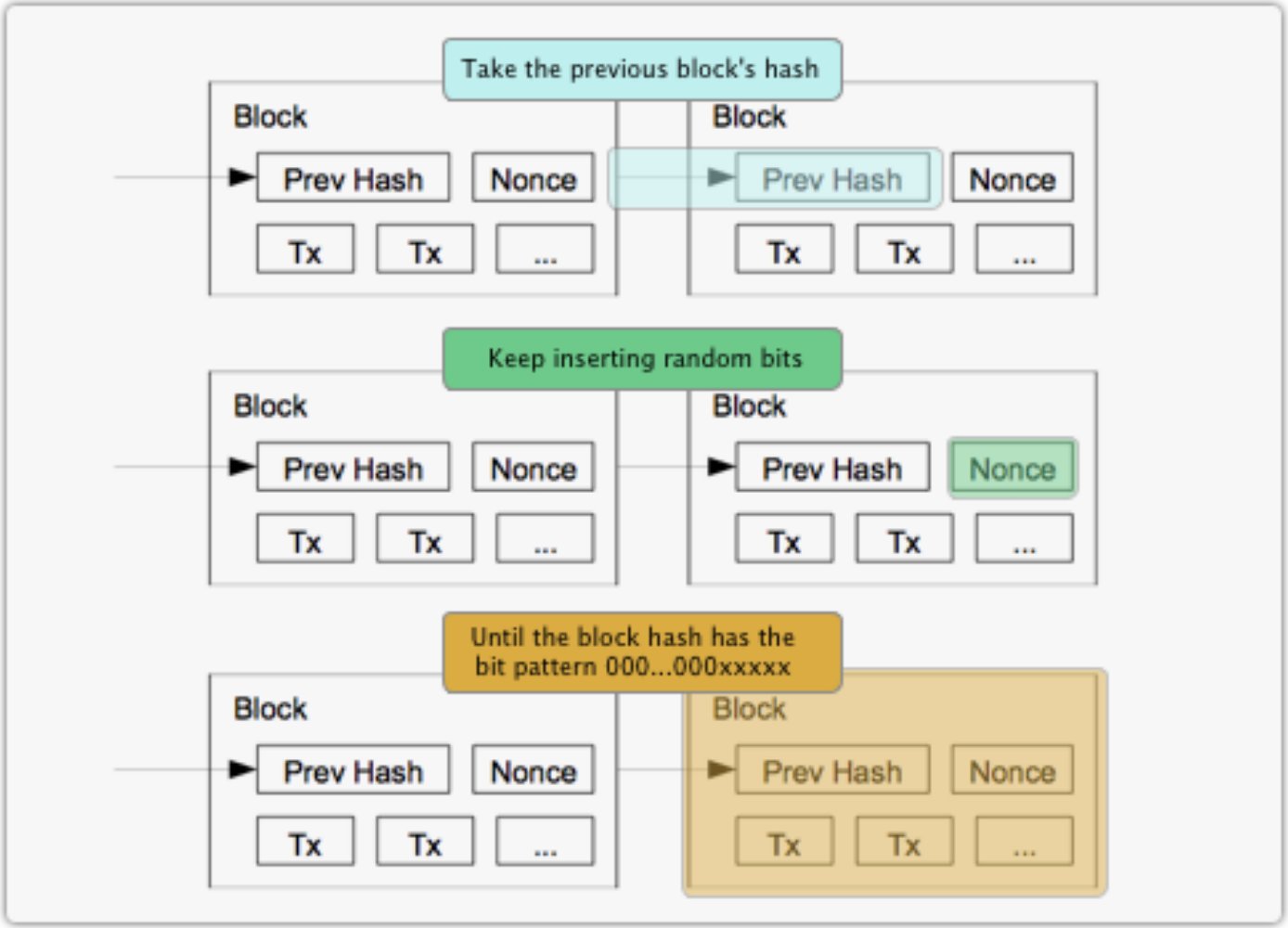
3 – Nodi della rete o *Miners* (“minatori”) e la loro attività di *mining*

Proof of Work - PoW



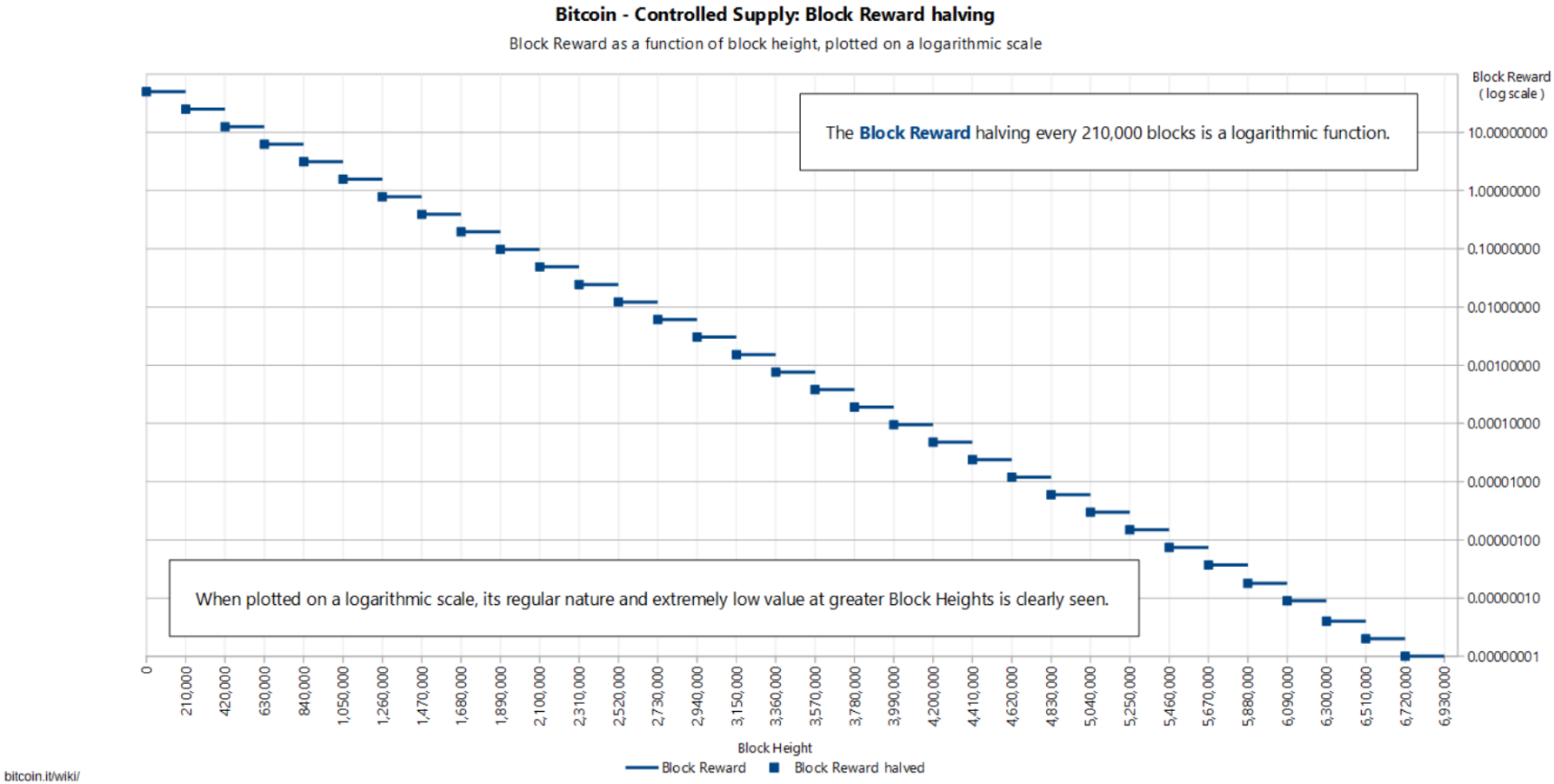
Hash del blocco

Proof of Work - PoW



Proof of Work - PoW

Ciò crea una competizione tra i miners; il primo che riesce a trovare la stringa corretta da inserire nel *nonce* viene ricompensato con un numero di BTC che viene dimezzato ogni 210.000 blocchi (circa 4 anni) a partire da 50 BTC; ora la ricompensa è di 12,5 BTC.



bitcoin.it/wiki/

LA STRUTTURA DEL BITCOIN

Miners

LA STRUTTURA DEL BITCOIN

Miners

mining pool

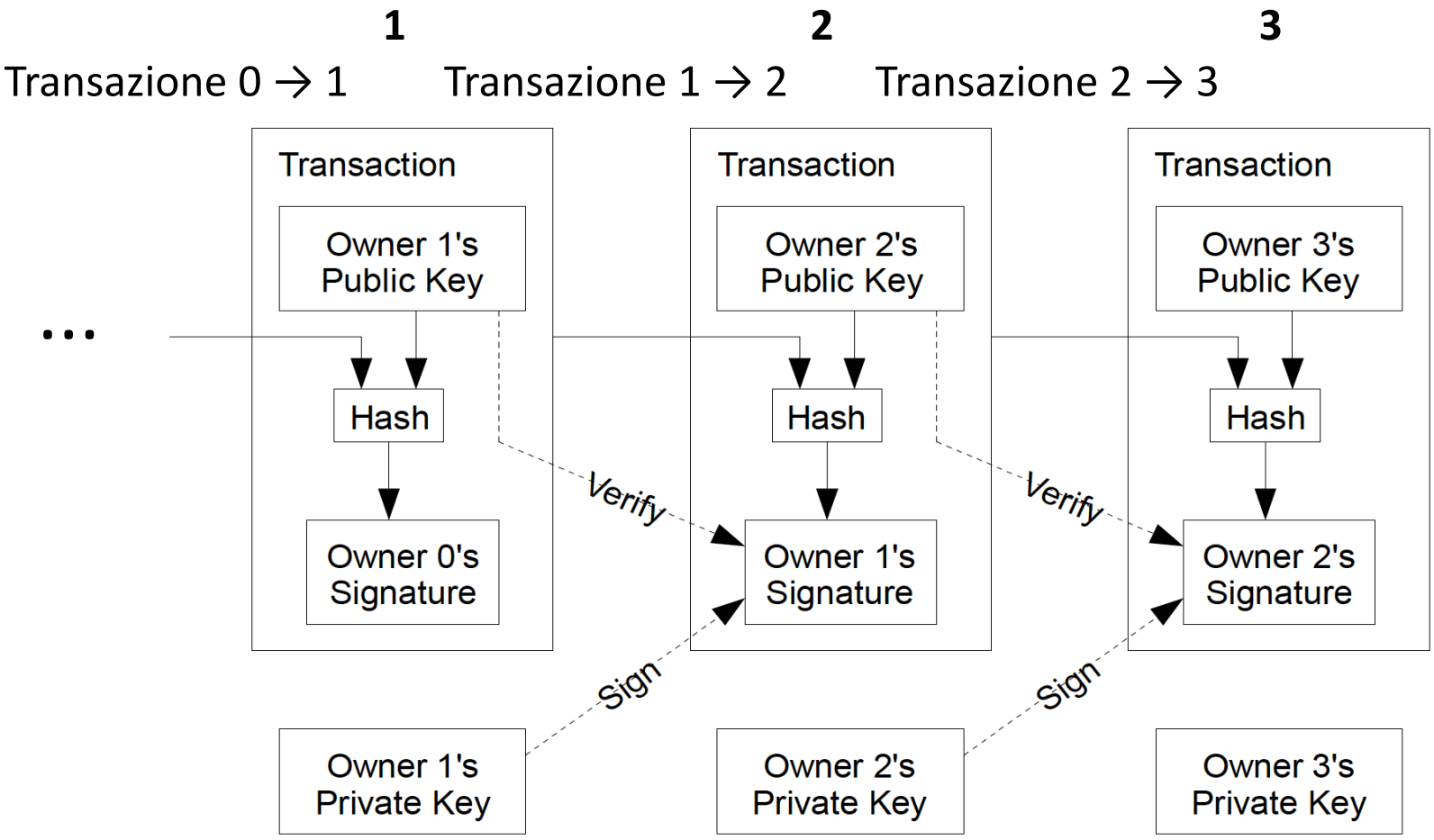
LA STRUTTURA DEL BITCOIN

Miners

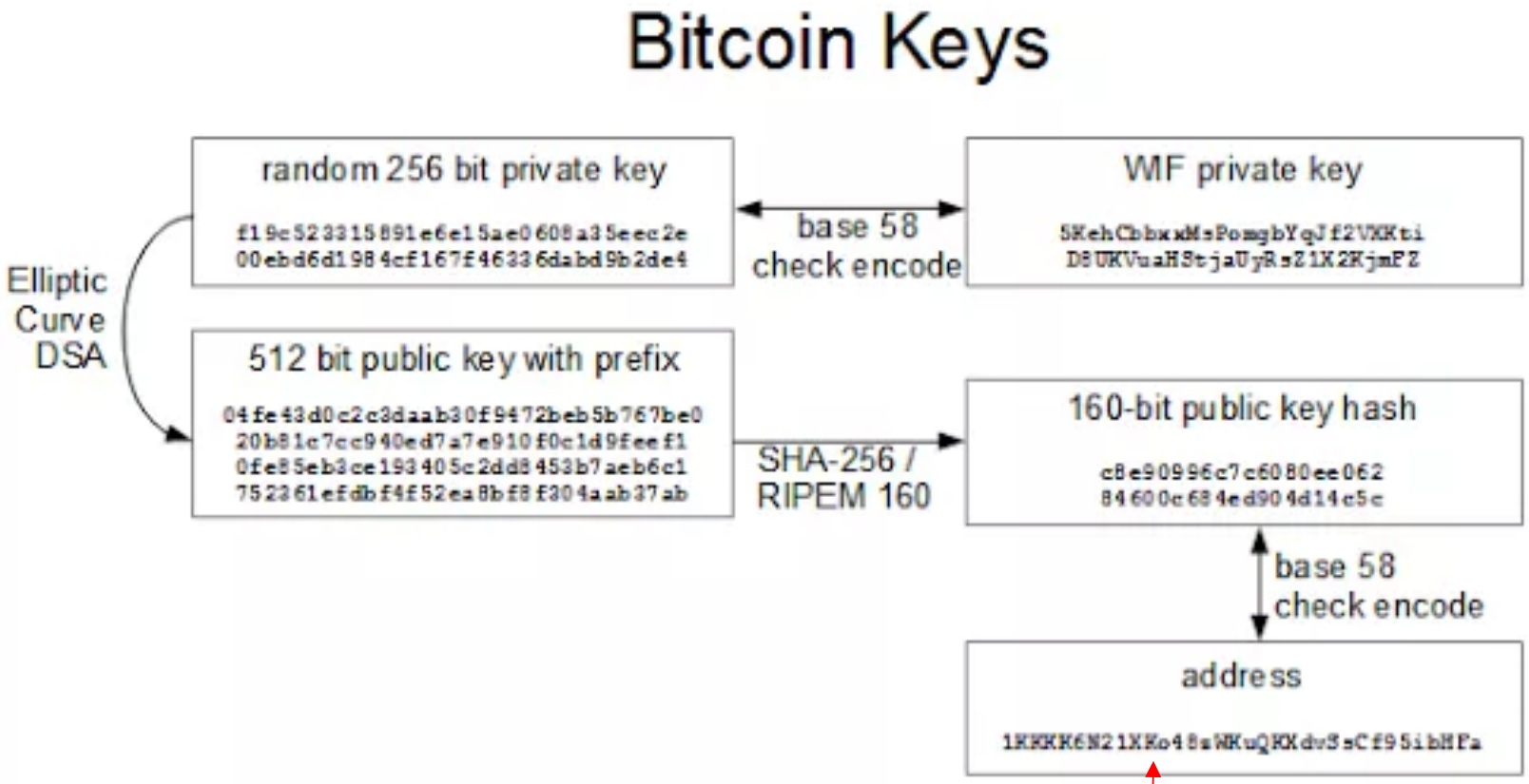
poi, quando
il prezzo
del bitcoin
cala troppo...

Struttura della transazione:

1. L’utente che desidera inviare denaro crea un messaggio con la richiesta del trasferimento
2. il nodo che accetta la richiesta la inoltra su tutti inodi della rete in modo sincrono
3. i trasferimenti vengono effettuati tramite indirizzi Bitcoin (*Bitcoin address*), che sono l’equivalente di un IBAN bancario
4. ogni indirizzo è l'*hash* di una chiave pubblica crittografica
5. ogni utente può generare quanti indirizzi vuole
6. il messaggio del mittente è firmato digitalmente per dimostrare la proprietà del denaro
7. il nodo ricevente verifica la firma e inoltra il messaggio a tutti gli altri nodi della rete
8. tutte le transazioni Bitcoin sono pubbliche


1
2
3
Transazione 0 → 1
Transazione 1 → 2
Transazione 2 → 3
Transaction
Transaction
Transaction
Owner 1's Public Key
Owner 2's Public Key
Owner 3's Public Key
...
Hash
Hash
Hash
Verify
Verify
Owner 0's Signature
Owner 1's Signature
Owner 2's Signature
Sign
Sign
Owner 1's Private Key
Owner 2's Private Key
Owner 3's Private Key

ECDSA – Elliptic Curve Digital Signature Algorithm



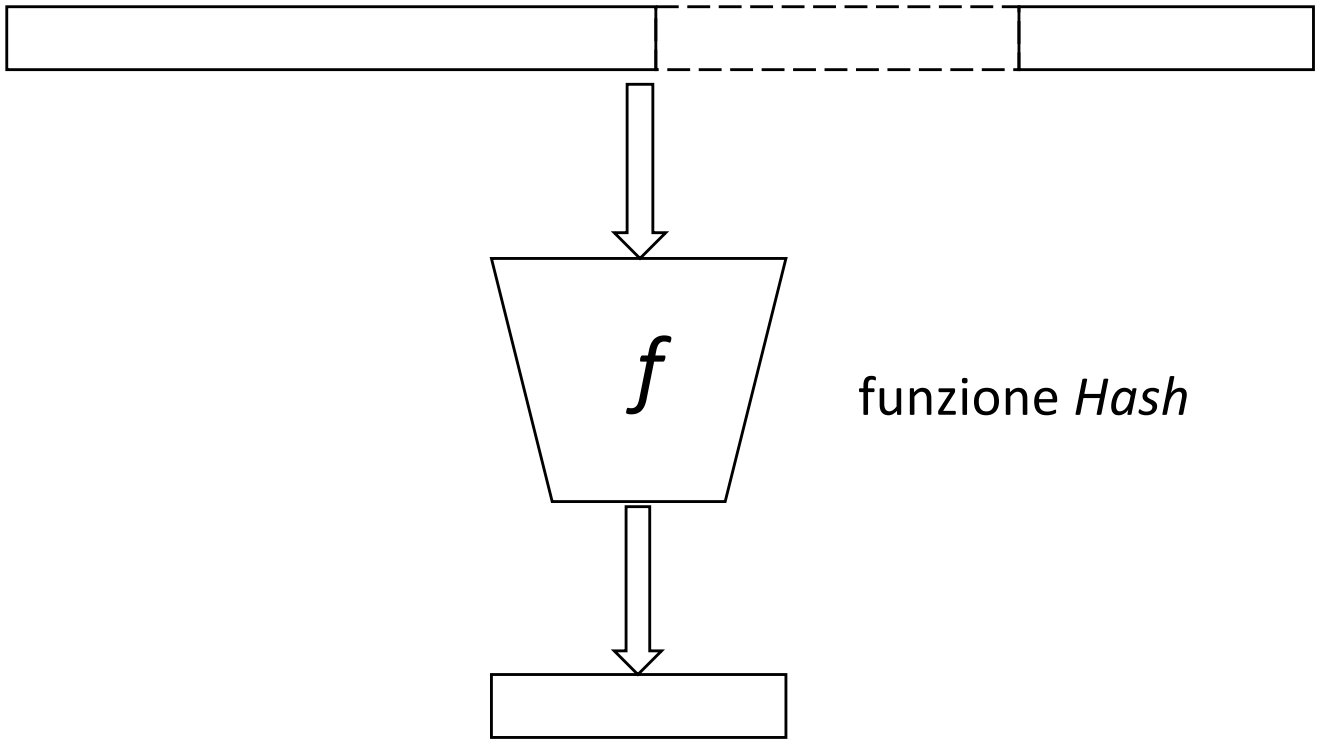
stringa da 34 caratteri in Base58 che costituisce l'indirizzo effettivo

Appendice 1

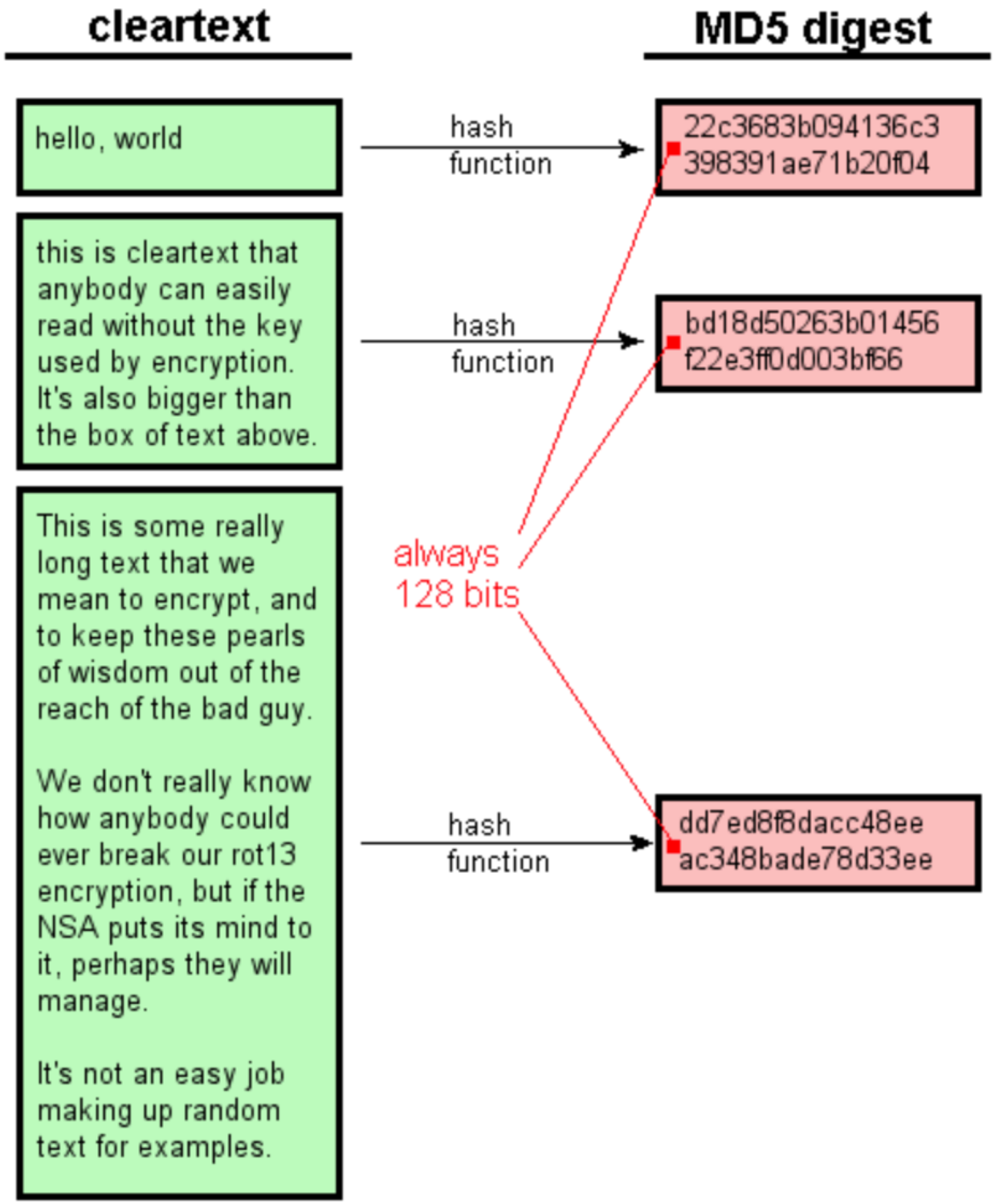
le funzioni Hash

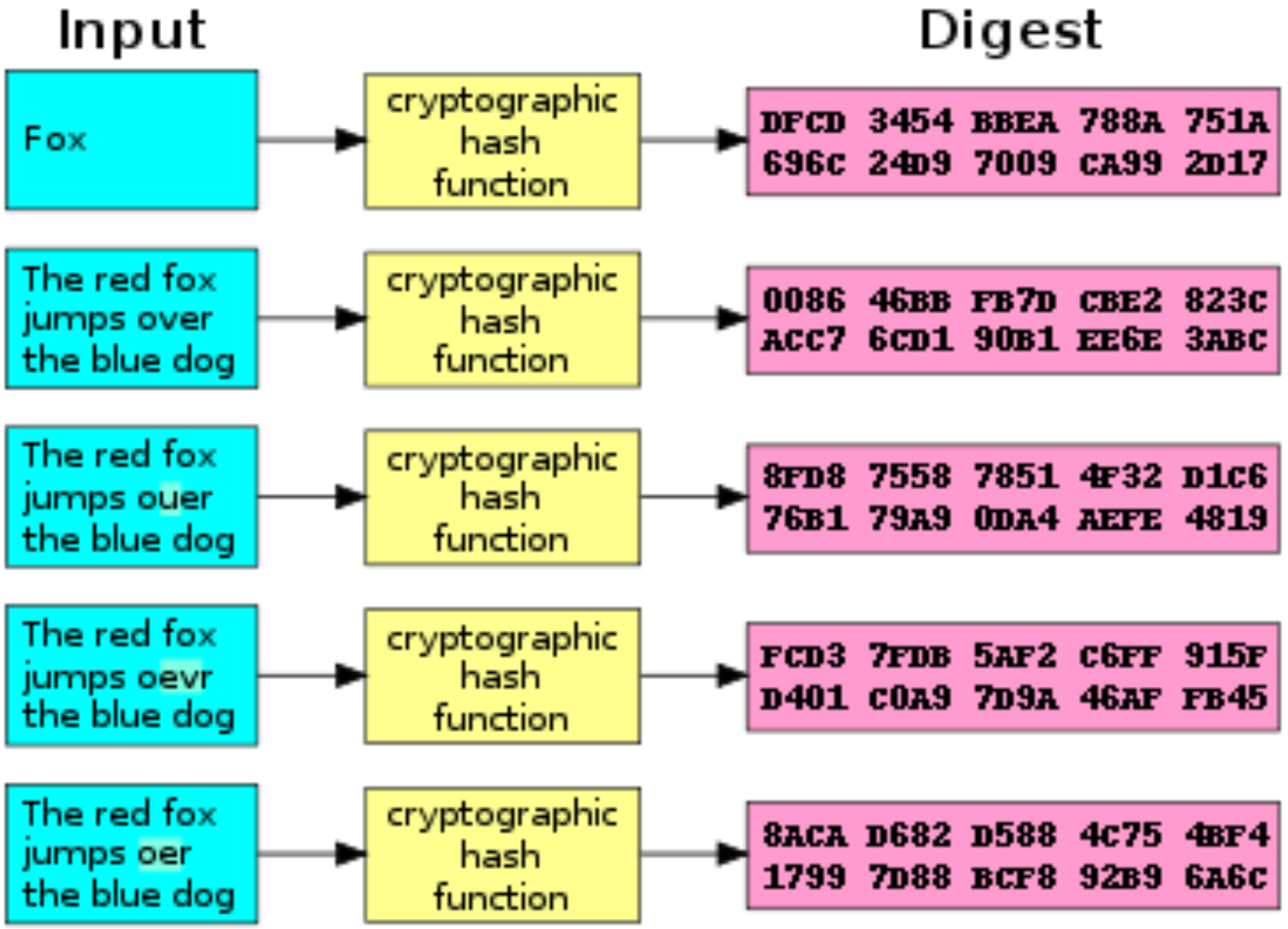
Funzione *Hash*

stringa x di lunghezza arbiraria pari a *n* bit



stringa $H(x)$ di lunghezza costante (estratto)


cleartext
MD5 digest
hello, world
hash function
22c3683b094136c3 398391ae71b20f04
this is cleartext that anybody can easily read without the key used by encryption. It's also bigger than the box of text above.
hash function
bd18d50263b01456 f22e3ff0d003bf66
This is some really long text that we mean to encrypt, and to keep these pearls of wisdom out of the reach of the bad guy.
We don't really know how anybody could ever break our rot13 encryption, but if the NSA puts its mind to it, perhaps they will manage.
It's not an easy job making up random text for examples.
always 128 bits
hash function
dd7ed8f8dacc48ee ac348bade78d33ee


Input
Digest
Fox
cryptographic hash function
DFCD 3454 BBEA 788A 751A 696C 24D9 7009 CA99 2D17
The red fox jumps over the blue dog
cryptographic hash function
0086 46BB FB7D CBE2 823C ACC7 6CD1 90B1 EE6E 3ABC
The red fox jumps ouer the blue dog
cryptographic hash function
8FD8 7558 7851 4F32 D1C6 76B1 79A9 0DA4 AEFE 4819
The red fox jumps oevr the blue dog
cryptographic hash function
FCD3 7FDB 5AF2 C6FF 915F D401 C0A9 7D9A 46AF FB45
The red fox jumps oer the blue dog
cryptographic hash function
8ACA D682 D588 4C75 4BF4 1799 7D88 BCF8 92B9 6A6C

Esempio banale: resto della divisione

Assegnato un numero intero qualunque *x*, l'estratto *H*(*x*) consiste in un numero compreso tra 0 e 255

$$34739 = 135*256 + 179$$

$$H(34739) = 179$$

Funzione *Hash* crittografica

funzione deterministica unidirezionale

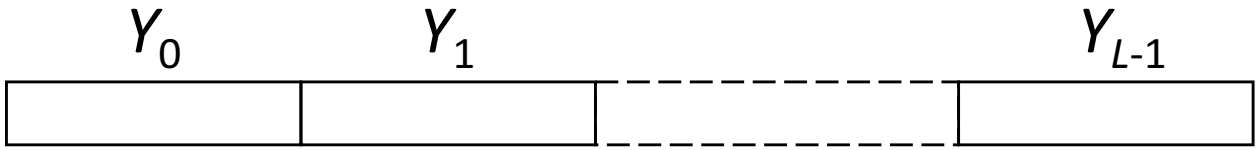
$$H : \{0,1\}^* \rightarrow \{0,1\}^n$$

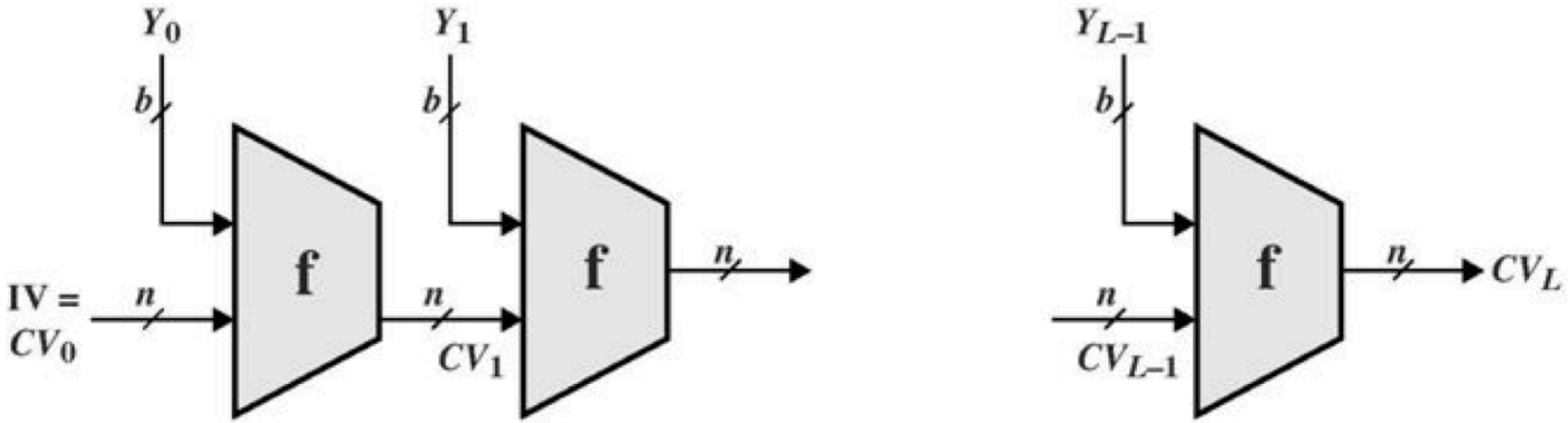
noto $H(x)$ non è possibile ricavare x

noto $H(y)$ non è possibile trovare $x \neq y$ tale che $H(x) = H(y)$

la piccola variazione anche di un solo bit di x porta a una variazione di $H(x)$

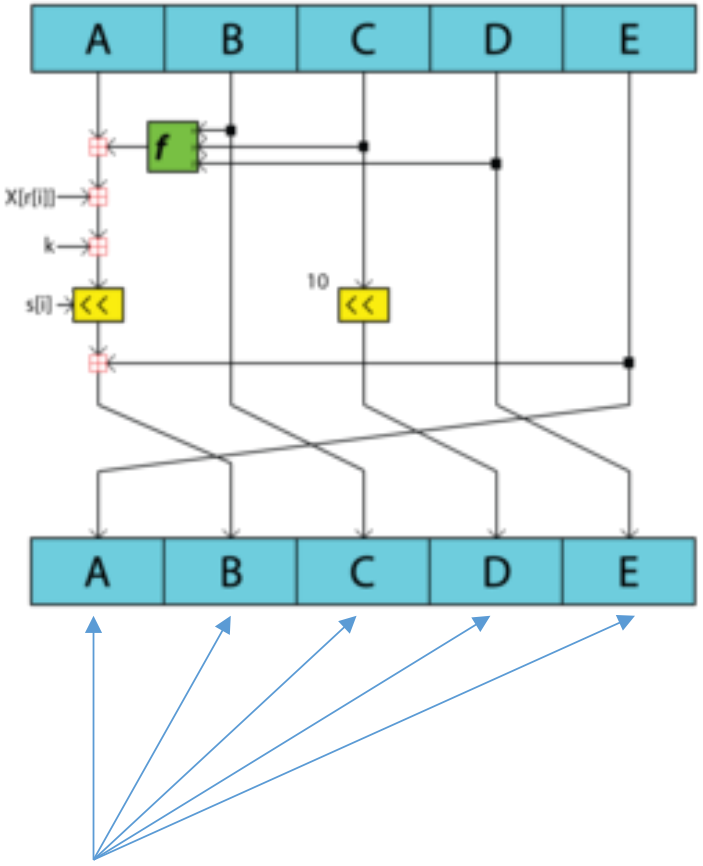
Struttura generale di una funzione *Hash* crittografica





IV = Initial value
CV = chaining variable
Y_i = *i*th input block
f = compression algorithm
L = number of input blocks
n = length of hash code
b = length of input block

RIPEMD160



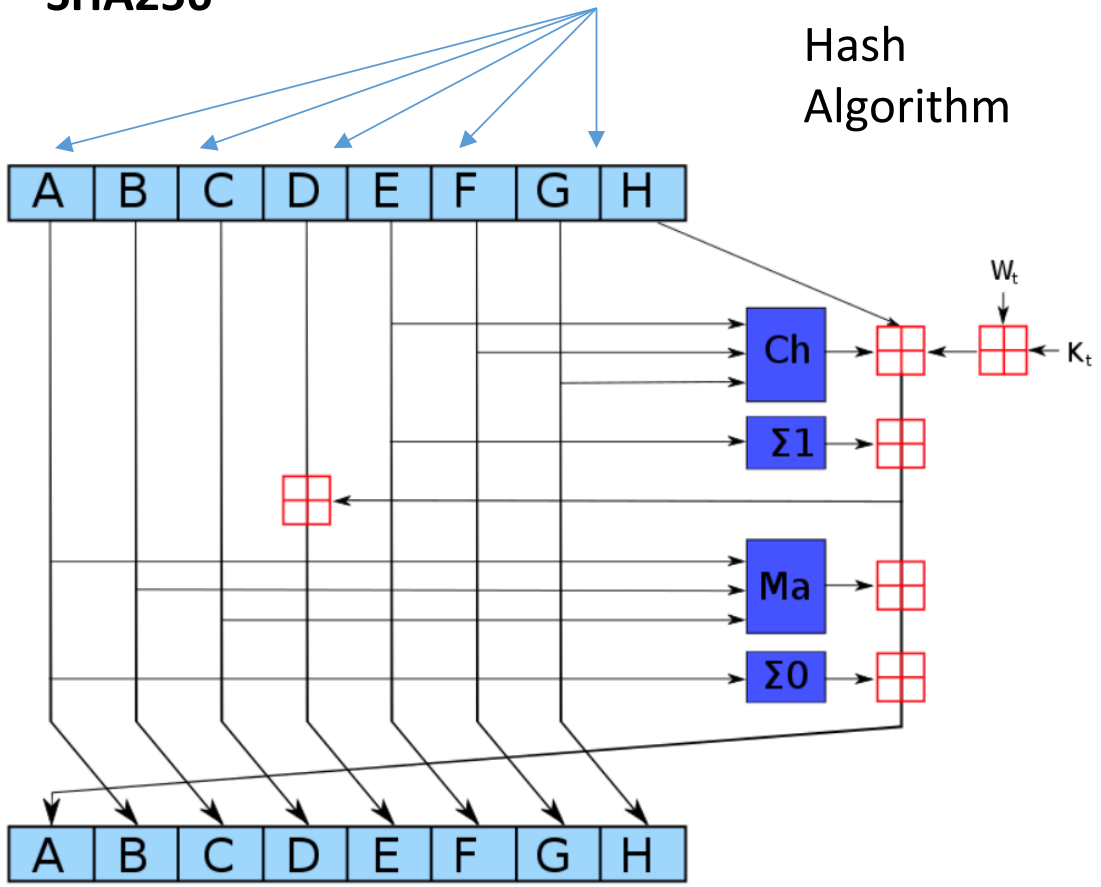
blocchi 32 bit

RACE Integrity Primitives Evaluation Message Digest

SHA256

blocchi 32 bit

Secure Hash Algorithm



One iteration in a SHA-2 family compression function. The blue components perform the following operations:

$$\mathrm{Ch}(E,F,G) = (E \wedge F) \oplus (\neg E \wedge G)$$
$$\mathrm{Ma}(A,B,C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$$
$$\Sigma_0(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$$
$$\Sigma_1(E) = (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)$$

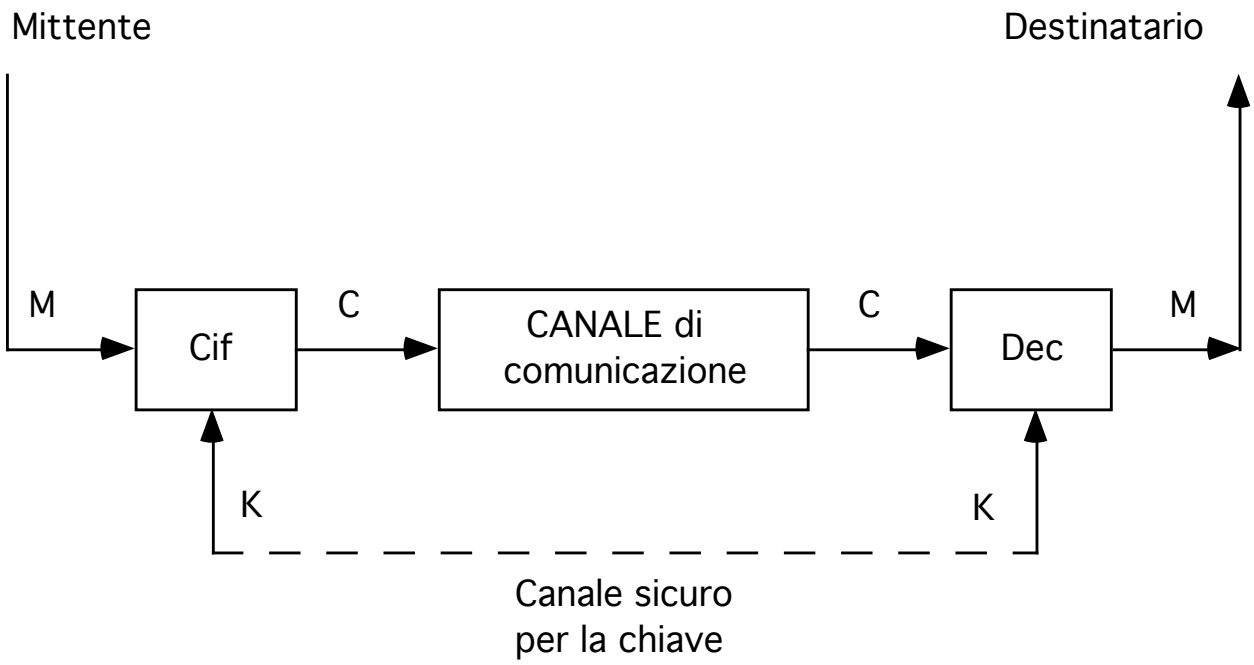
The bitwise rotation uses different constants for SHA-512. The given numbers are for SHA-256.

The red ⊞ is addition modulo 2^{32} for SHA-256, or 2^{64} for SHA-512.

Appendice 2

la Crittografia

Cenni sulla crittografia



Decrittazione

Ricavare M da C senza usare K

C = Cif(M, K)

M = Dec(C, K)

M = Messaggio

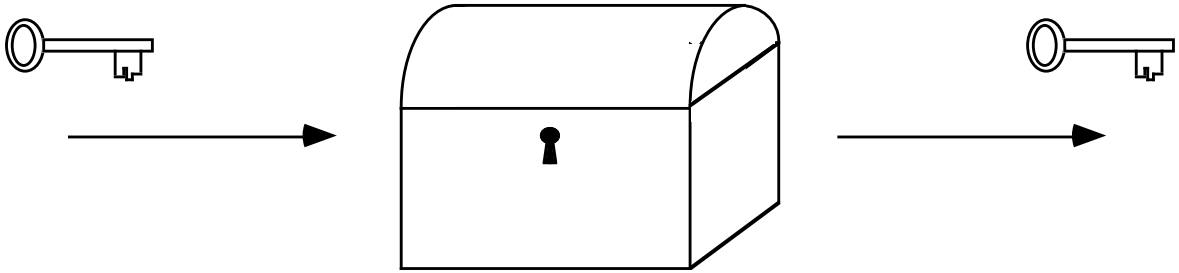
C = Crittogramma

K = Chiave

Cifratura

Decifrazione

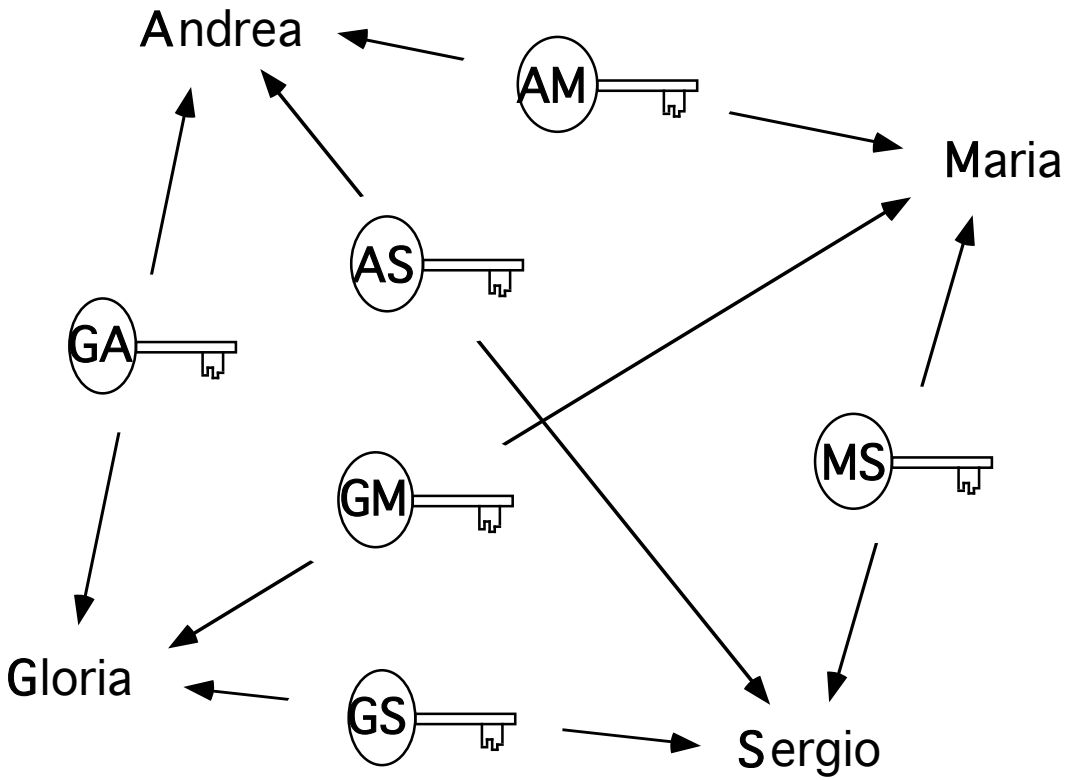
Approccio classico:
crittografia a chiave segreta (simmetrica)



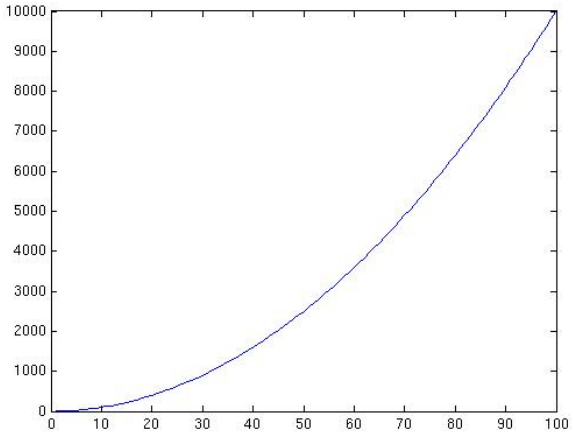
Cifratura :
il mittente chiude
la serratura

Decifrazione :
il ricevente apre
la serratura

Problema della gestione delle chiavi

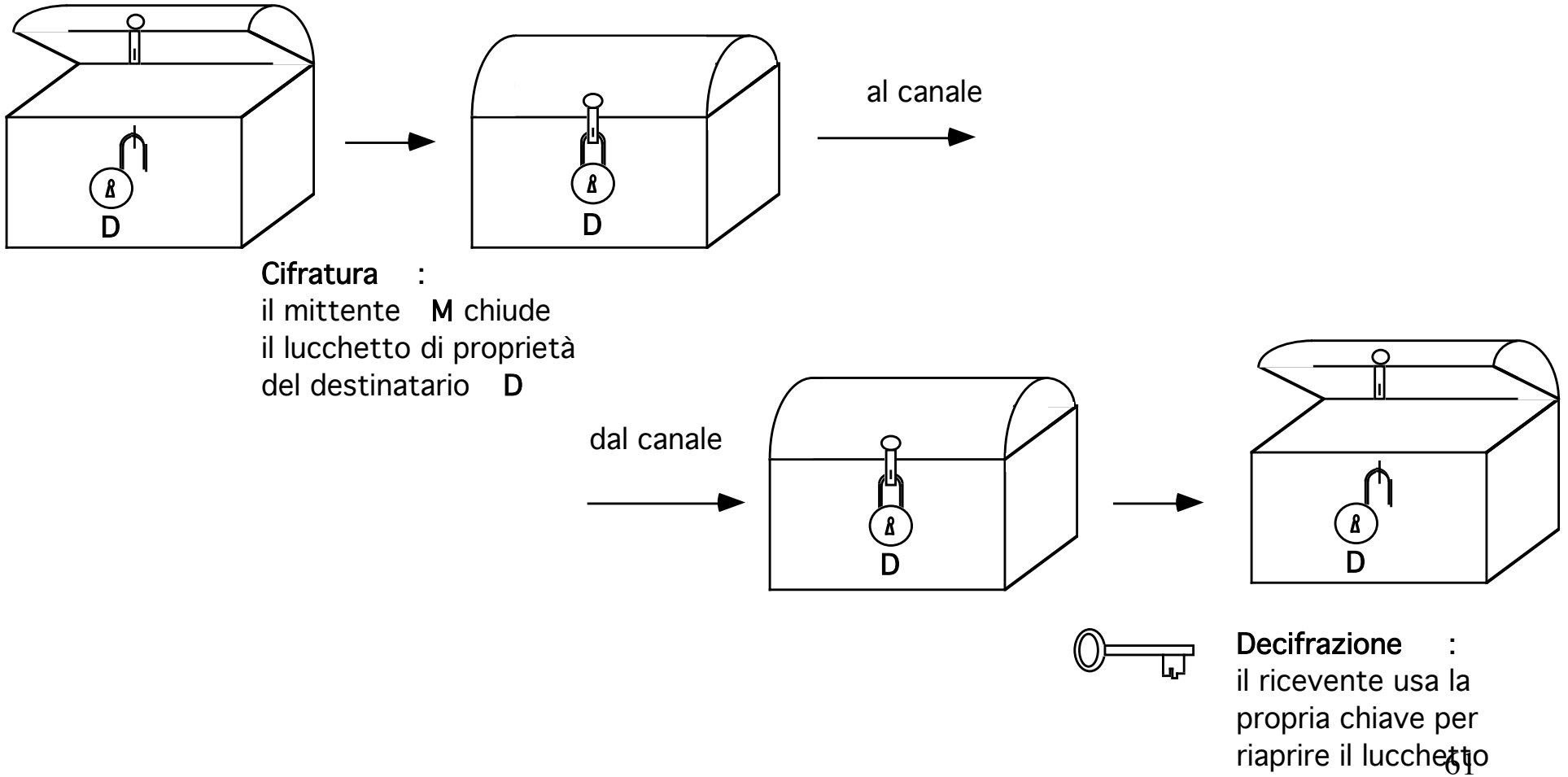


N	Numero Chiavi
10	45
100	4950
1000	499500



N utenti $\longrightarrow \frac{N\,(N-1)}{2}$ chiavi

Approccio moderno:
crittografia a chiave pubblica (asimmetrica)



Cifratura :
il mittente **M** chiude
il lucchetto di proprietà
del destinatario **D**

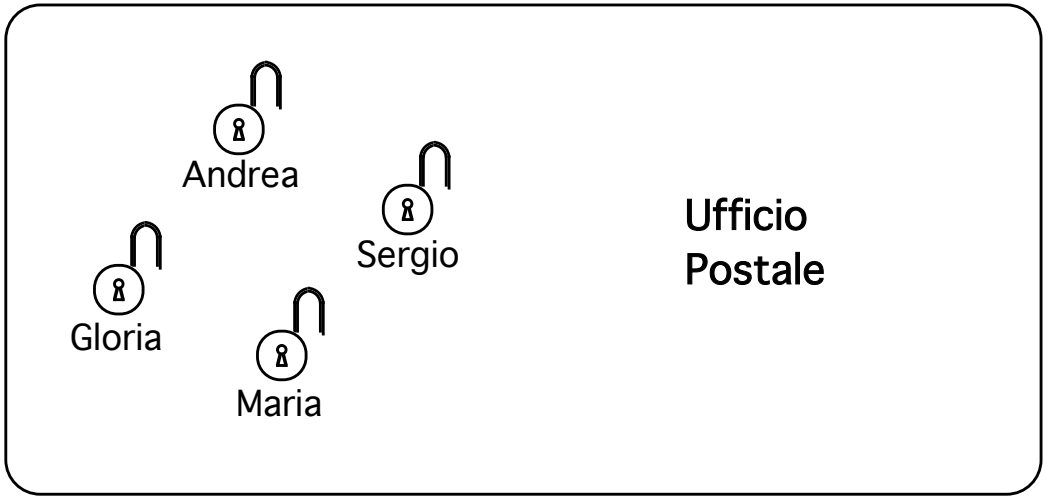
Decifrazione :
il ricevente usa la
propria chiave per
riaprire il lucchetto

Se Maria vuole mandare un messaggio a Sergio

1) si reca all'ufficio postale e prende il lucchetto di Sergio

2) lo chiude sul baule contenente il messaggio e lo spedisce

3) solo Sergio è in grado di riaprire il baule



Vantaggio della crittografia a chiave pubblica:

Le chiavi non devono essere scambiate

(ciascuno conserva la propria in una cassaforte)

La chiave di cifratura (chiusura del lucchetto)
e' (un'operazione) diversa dalla
chiave di decifrazione (apertura del lucchetto)

Conseguenza:

Mittente e ricevente

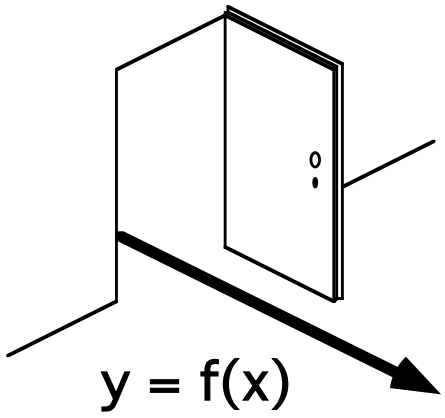
non devono condividere alcuna chiave

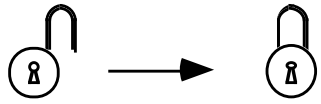
$$X:\ c = C_y(m) \qquad \xrightarrow{canale} \qquad Y:\ m = D_y(c) = D_y C_y(m)$$

Come realizzare il "lucchetto" da un punto di vista matematico?...

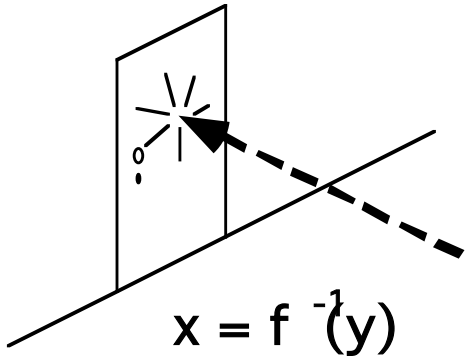
...Impiegando le *Funzioni Unidirezionali*

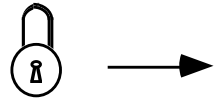
facili da calcolare in **senso diretto**





(chiusura del lucchetto)





(tentativo di riapertura senza la chiave)

praticamente **impossibili** da calcolare **in senso inverso**

Utenti	Chiave di cifratura (pubblica)	Chiave di decifrazione (privata)
X	C_X	C_X^{-1}
Y	C_Y	C_Y^{-1}
Z	C_Z	C_Z^{-1}
:	:	:

Poiché la funzione è unidirezionale:

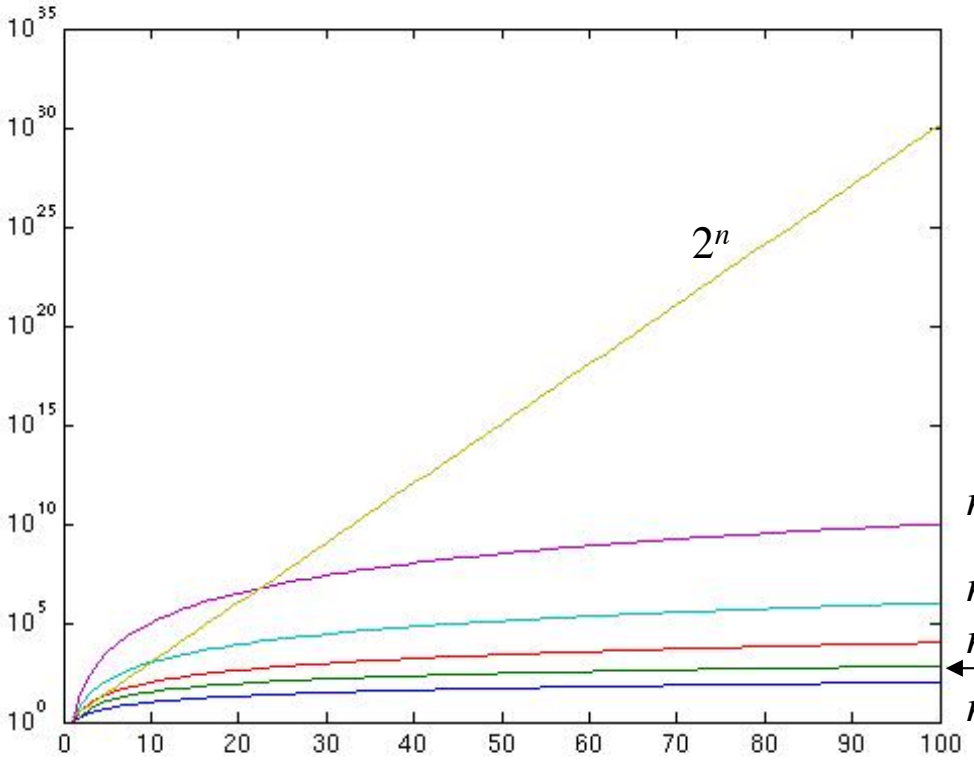
dalla chiave pubblica (funzione) di cifratura deve essere *praticamente impossibile* ricavare la chiave privata (funzione) di decifrazione

Per capire cosa significa "praticamente impossibile"
bisogna fare riferimento alla teoria della

Complessità computazionale

Complicazione rilevante:

mentre alcuni problemi sono "facili" (esistono algoritmi efficienti)
altri problemi sono "strutturalmente" difficili e per essi
NON sono noti algoritmi efficienti.



problemi *intrattabili* = **difficili**

problemi *trattabili* = **facili**

$n = 10^6$

crescita con	n	$n \log n$	n^2	n^3	2^n
tempo, 10^9 op/s	1 ms	6 ms	16.66 m	31.7 a	$3.14\ 10^{301004}$ mld anni
tempo, 10^{12} op/s	1 μs	6 μs	1 s	11.6 g	$3.14\ 10^{301001}$ mld anni

$t = 24\ h$

crescita con	n	$n \log n$	n^2	n^3	2^n
dim. del problema, 10^9 op/s	$8.64\ 10^{13}$	$3.00\ 10^{12}$	$9.29\ 10^6$	44208	46
dim. del problema, 10^{12} op/s	$8.64\ 10^{16}$	$2.44\ 10^{15}$	$294\ 10^6$	442083	56

Esempi di fiunzioni unidirezionali usate in crittografia

0) problema del logaritmo finito

1) problema delle somme parziali

2) fattorizzazione di un intero

3) problema del logaritmo finito su curve ellittiche

0) Problema del logaritmo finito

funzione **diretta** : dato **p** primo, un esponente **x** e una base **a** (primitiva) calcolare

$$y = a^x \mod p$$ **facile**

funzione **inversa** : dato **y** trovare il suo logaritmo **x** in base **a**

$$x = \log_a y \mod p$$ **difficile**

1) Problema delle somme parziali

$$(7, 11, 2, 9, 5) \cdot (1, 0, 1, 0, 1) = 14$$

dati gli interi

e la quintupla binaria

è facile trovare il prodotto scalare (o somma parziale)

viceversa

e la somma parziale

$$(7, 11, 2, 9, 5) \cdot (\ ? \ ) = 14$$

dati gli interi

NON è facile trovare gli elementi costituenti la somma parziale
(se i vettori sono molto grandi)

2) Problema della fattorizzazione di un intero

Dati p e q interi primi (molto grandi) è facile calcolare

$$n = p \cdot q$$

Dato n intero (molto grande)
NON è facile scomporlo nei suoi fattori primi

$$n = p \cdot q$$

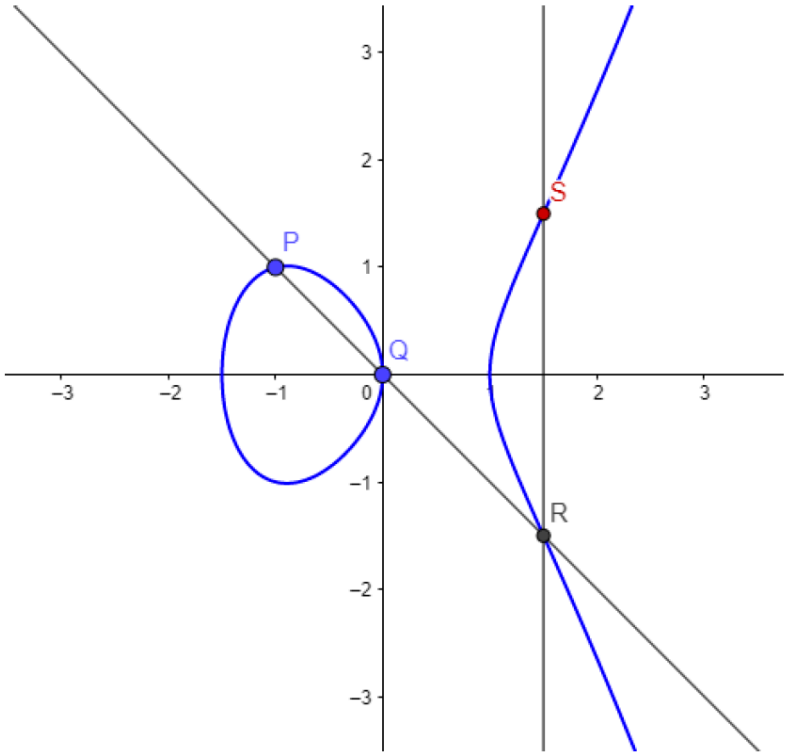
3) Problema ECDLP Elliptic Curve Discrete Logarithm Problem

ECDSA – Elliptic Curve Digital Signature Algorithm

Neal Koblitz Victor S. Miller

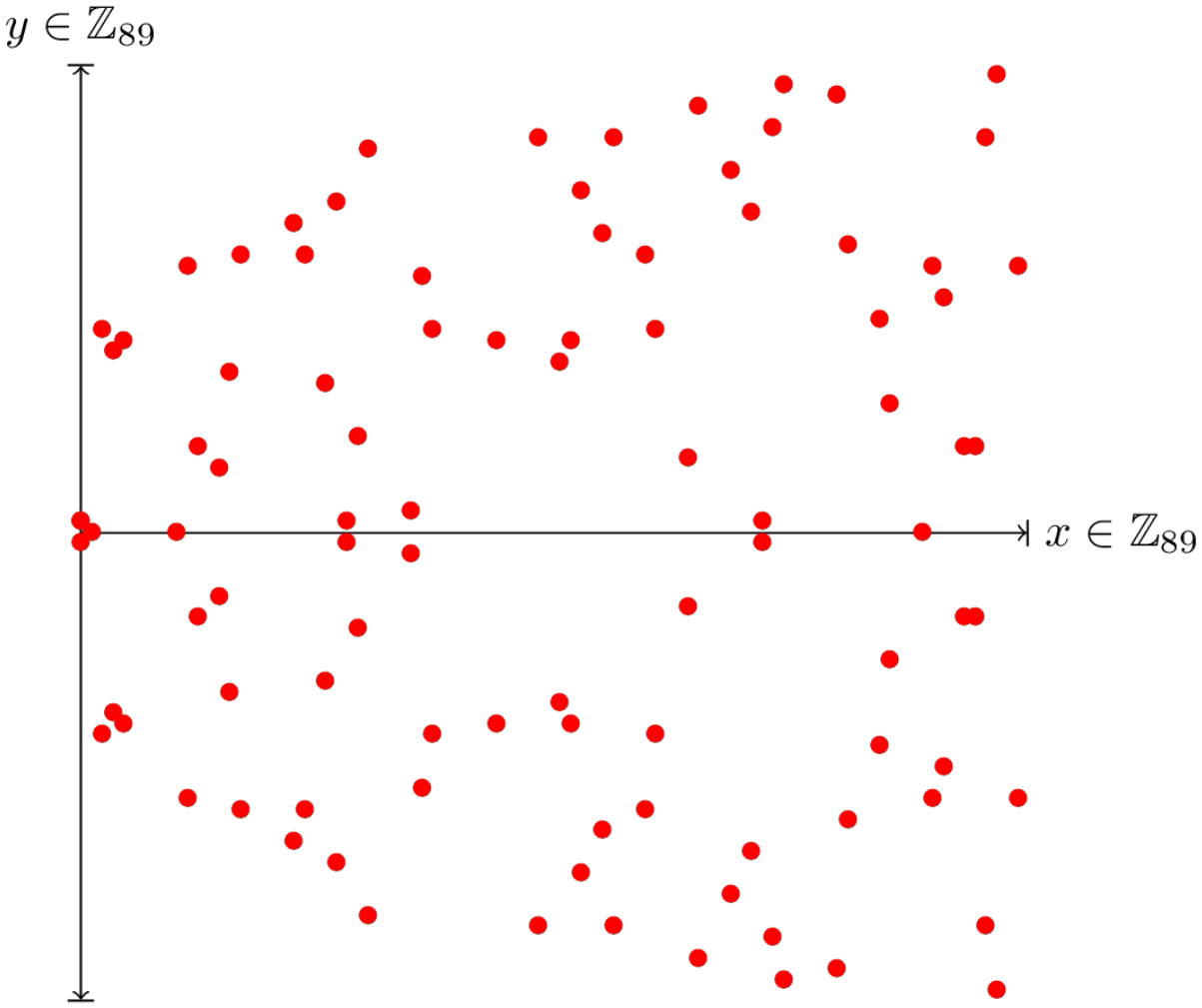
$$y^2 = x^3 + ax + b$$

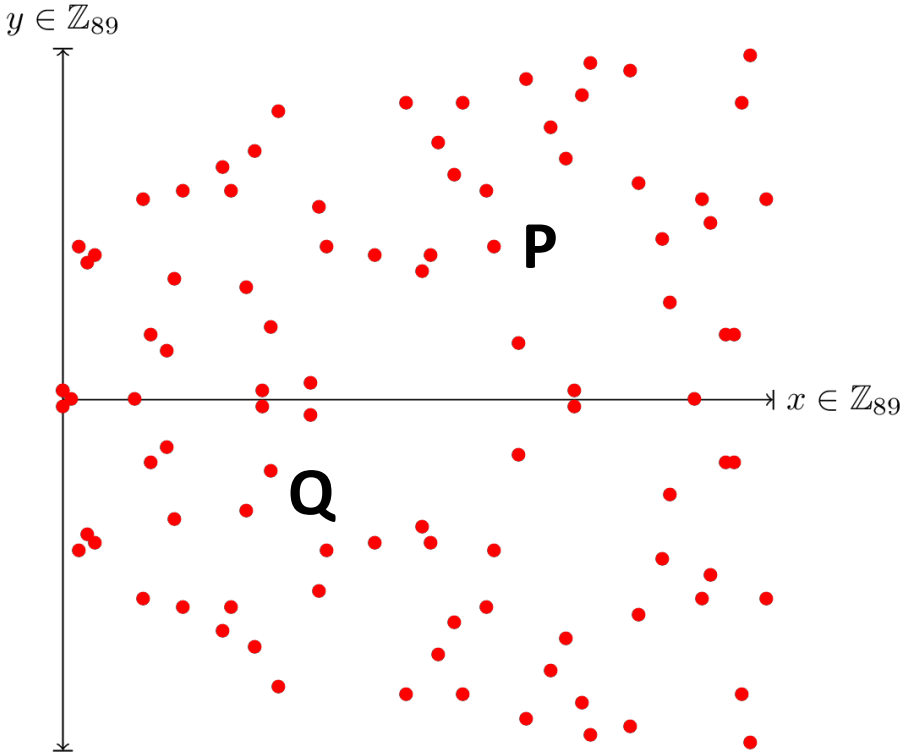
$$y^2 = x(x-1)\left(x - \frac{3}{2}\right) \quad su\ \mathbb{Q}.$$



gli elementi sono su un campo finito GF(p^q)

In realtà si lavora su un campo finito di Galois;
ecco come potrebbe apparire effettivamente la “curva” su un piano cartesiano





assegnato *n* si calcola $Q = n\,P$ **facile**

assegnati Q e P non è possibile risalire a *n* **difficile**

Cifratura a chiave pubblica:

X vuole mandare a Y un messaggio ***m***:

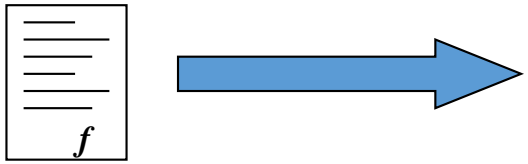
1. legge la chiave pubblica C_Y di Y
2. costruisce il crittogramma $\boldsymbol{c} = C_Y(\boldsymbol{m})$

3. lo invia sul canale
4. Y lo riceve, ed e' l'unico a conoscere C_Y^{-1}
5. esegue l'operazione $C_Y^{-1}(\boldsymbol{c}) = C_Y^{-1}C_Y(\boldsymbol{m}) = \boldsymbol{m}$

Firma numerica

Y vuole firmare un messaggio a X:

1. usa la propria chiave segreta C_Y^{-1} (e' l'unico a conoscerla)
2. costruisce la firma $f = C_Y^{-1}(Y)$



3. invia la firma sul documento che viene trasmesso
4. X la riceve, e usa la chiave pubblica C_Y per verificare la firma
5. esegue l'operazione $C_Y(f) = C_Y C_Y^{-1}(Y) = Y$

Integrità dei documenti