Law, Science and Technology
MSCA ITN EJD n. 814177

Mirko Zichichi

(Sicurezza Informatica e) **Blockchain**

Mirko Zichichi https://mirkozichichi.me

Dottorando in Law, Science, and Technology Joint Doctorate - Rights of Internet of Everything

“Decentralized Systems for the Protection and Portability of Personal Data”

Ontology Engineering Group (OEG), Universidad Politécnica de Madrid

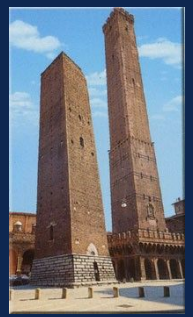
e-mail: mirko.zichichi@upm.es

Dipartimento di Scienze Giuridiche Università di Bologna

*e-mail: **mirko.zichichi2@unibo.it***

Dipartimento di Giurisprudenza Università di Torino

email: mirko.zichichi@unito.it



Outline

- **Introduzione alla Sicurezza Informatica**
- **Crittografia**
  - Simmetrica
  - Asimmetrica
- **Funzioni Hash**
  - Hash pointers
- **Firma Digitale**
- **Introduzione ai Sistemi Distribuiti**
- **Blockchain**
  - Transazioni
  - Blocchi
  - Meccanismi di Consenso

IL REPORT

Data breach: in Italia danni per oltre 3 milioni di euro durante la pandemia

Home > Cyber Security

Condividi questo articolo

Secondo uno studio del Ponemon Institute, promosso da Ibm, il costo per ogni informazione rubata è di 135 euro, valore quasi raddoppiato nell'ultimo decennio. Per identificare e contenere una minaccia informatica servono 250 giorni. Aumentano gli attacchi al settore della sanità

28 Lug 2021

https://www.corrierecomunicazioni.it/cyber-security/data-breach-in-italia-danni-per-oltre-3-milioni-di-euro-durante-la-pandemia/

*“**07/08/2021***

COMUNICAZIONE AGLI INTERESSATI

*In data 30/07/2021 un attacco informatico effettuato da Hacker al **data center** che ospita alcuni dei sistemi informatici della nostra Regione ha compromesso l'utilizzo di alcuni dei servizi e delle applicazioni a disposizione del cittadino.*

*Stiamo provvedendo a fare tutto il necessario per porre rimedio all'accaduto e bloccare questo attacco per evitare ulteriori **conseguenze sulla privacy e la sicurezza dei dati personali dei cittadini** in possesso della Regione.”*

https://www.regione.lazio.it/notizie/attacco-hacker

La Sicurezza Informatica

- La sicurezza informatica è la protezione delle risorse dall'accesso, utilizzo, alterazione o distruzione non autorizzati
- Due tipi di sicurezza
  - Fisica: protezione dei dispositivi fisici tramite allarmi, antifurto, porte blindate, casseforti...
  - Logica: protezione delle informazioni tramite risorse non fisiche (crittografia, firma elettronica...)

Sicurezza Informatica: Terminologia

- Confidenzialità (Segretezza)
  - Impedire la divulgazione non autorizzata di dati, garantire l'autenticità della fonte
- Integrità
  - Impedire le modifiche non autorizzate ai dati
- Autenticazione
  - Verificare l'identità della controparte (*con chi sto comunicando?*)
- Disponibilità
  - Impedire ritardi nella diffusione dei dati, o la loro rimozione
  - Es: Attacchi *Denial of Service* (*DoS*), Ransomware
- Non ripudiabilità
  - Impedire che la controparte possa negare una sua azione

Aspetti fondamentali

1. inesistenza di sistemi sicuri
2. sicurezza = conoscenza
3. entità componenti di un sistema (in)sicuro

1. Sistemi Sicuri

NON ESISTONO SISTEMI SICURI

- Il software può non essere perfetto
  - usualmente non lo è
  - ci possono esserci errori di progettazione nei protocolli che usiamo normalmente
- Il mito del sistema inviolabile deve essere assimilato a quello del caveau non svaligiabile o della nave inaffondabile (es. Titanic)
- Il **grado di sicurezza** è dato dal tempo necessario per violare il sistema, dall'investimento necessario e dalla probabilità di successo

1. Sistemi Sicuri: Complessità

UN SISTEMA PIÙ È COMPLESSO PIÙ È INSICURO

- La crescita di complessità porta alla creazione di sistemi insicuri
- Un buon punto di partenza per la creazione di sistemi sicuri è l'applicazione metodica della KISS rule:

Keep It Simple and Stupid

- Eliminare la complessità non necessaria
  - "Quello che non c'è non si rompe"

2. Sicurezza = Conoscenza

- Nessun sistema sconosciuto può essere considerato sicuro
- Quali implicazioni ha questa affermazione sul tema del software **Open Source vs. Closed Source**, **Sistemi Aperti vs. Sistemi Chiusi**?
  - Atto di fiducia nei confronti di una comunità vs. di un'azienda
- L'educazione degli utenti è una forma di conoscenza fondamentale
  - (es. percezione dell'importanza del tema sicurezza, pratiche comunemente diffuse ma altamente insicure, resistenza ideologiche o per semplice abitudine ecc.)

2. Sicurezza = Conoscenza
Principio di Kerckhoffs

"Un sistema crittografico deve essere sicuro anche se ogni suo aspetto, tranne le chiavi, è noto pubblicamente"

- Il contrario di "*Security by obscurity*"
- "*System security should not depend on the secrecy of the implementation or its components*"
  NIST special publication 800-123, Guide to General Server Security,

Auguste Kerckhoffs (1835 – 1903),
Source:

3. Entità Componenti di un Sistema

- Le entità che compongono un sistema sono:

  1) **HARDWARE**

  2) **SOFTWARE**

  3) **HUMANWARE**

- La sicurezza è un processo:
  - Serve un continuo apporto di lavoro e di educazione
  - Un sistema considerato sicuro oggi può non esserlo domani
  - I sistemi che non vengono aggiornati in modo continuo divengono fragili e insicuri
- La componente umana non deve mai essere sottovalutata

3. Entità Componenti di un Sistema: 3) HUMANWARE - Top Passwords 2018

- 123456
- password
- 123456789
- 12345678
- 12345
- 111111
- 1234567
- sunshine
- qwerty
- iloveyou

https://www.teamsid.com/100-worst-passwords-top-50/

3. Entità Componenti di un Sistema: 3) HUMANWARE - Top Passwords 2018

- 4.7% degli utenti ha come password *password*
- 8.5% ha come passwords *password* o *123456*
- 9.8% ha come passwords *password*, *123456* o *12345678*
- 14% ha una password tra le top 10 passwords
- 40% ha una password tra le top 100 passwords
- 79% ha una password tra le top 500 passwords
- 91% ha una password tra le top 1000 passwords

"Usare Pippo come password era troppo difficile?"

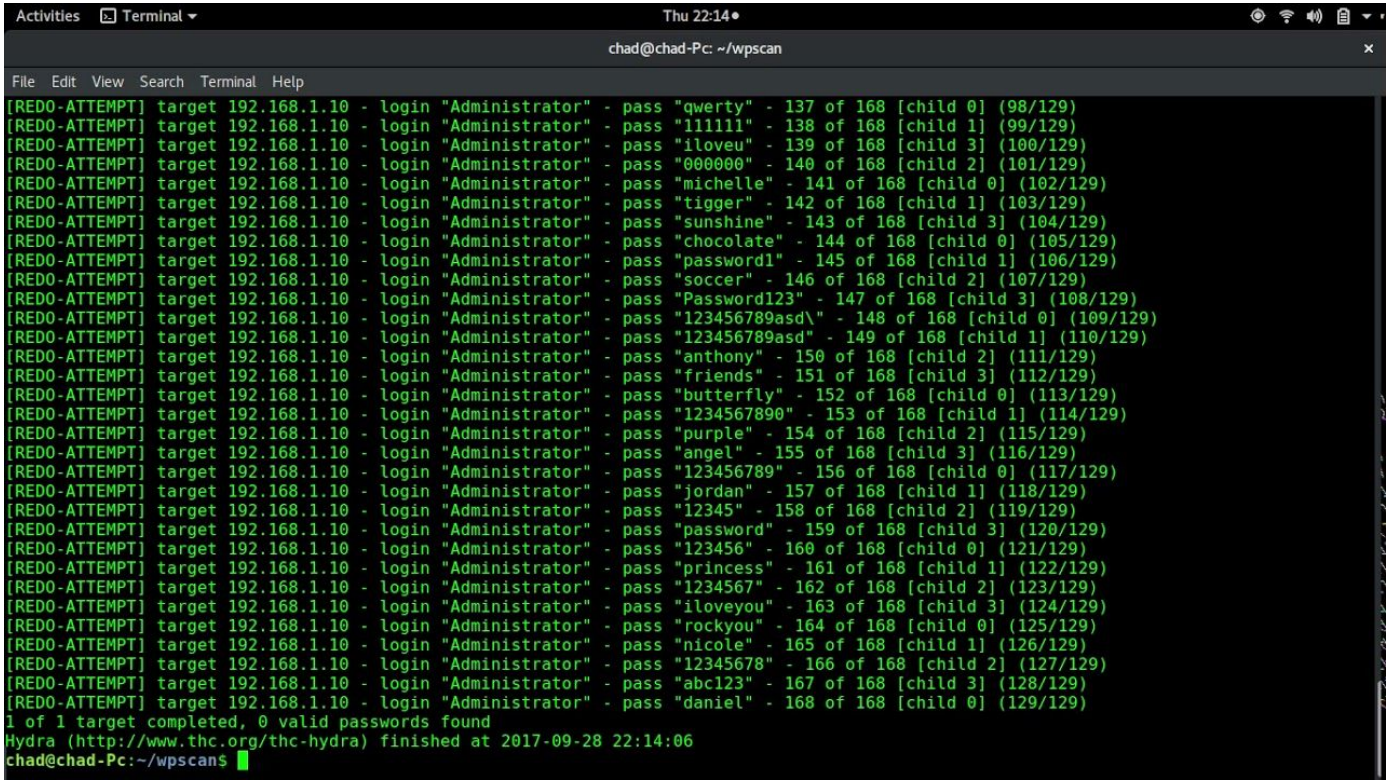
Video:
https://www.sondriotoday.it/attualita/password-pippo-olimpiadi-computer-telecronista-twitter.html

Tu che password usi?

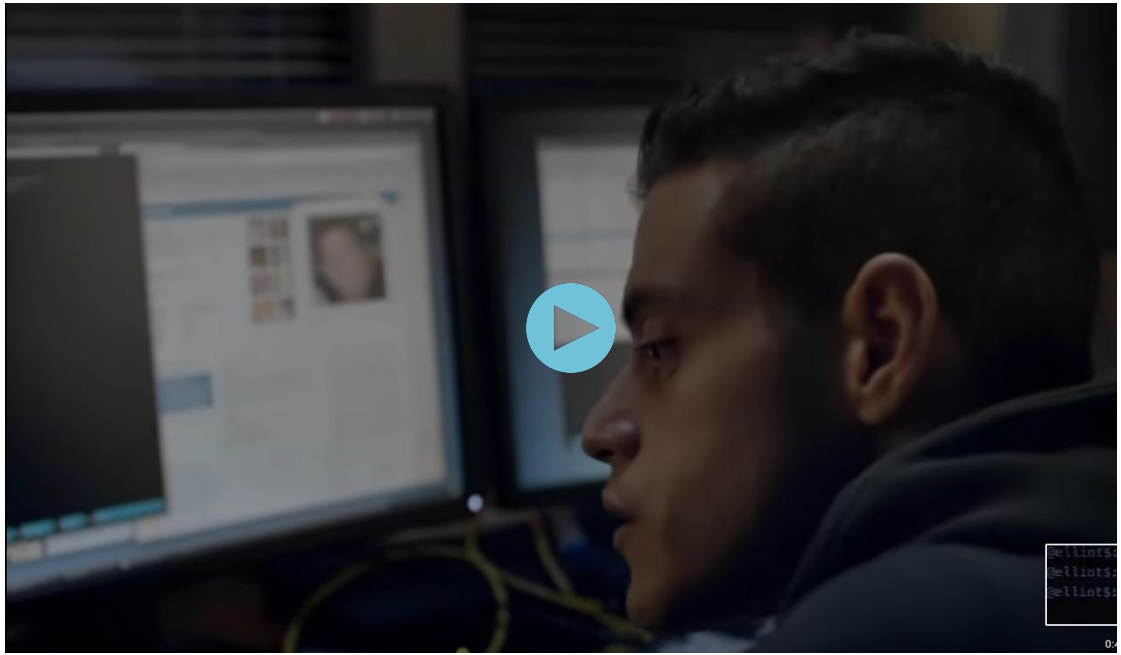
- Quasi tutto il mondo usa le stesse pwd
- Quindi è banale entrare nella maggior parte degli account
- Ecco come si spiegano tanti dei "furti di identità"

- Inoltre, studi hanno verificato che è facile *recuperare password da semplici informazioni pubbliche su facebook*

Dictionary Attack



Phishing + Dictionary Attack



Video: https://www.youtube.com/watch?v=JMYEr4Bgey4

*"**People** often represent the **weakest link in the security chain** and are chronically responsible for the failure of security systems."*

Secrets and Lies: Digital Security in a Networked World, Bruce Schneier, 2000

Cosa ci ha insegnato l'attacco hacker nel Lazio

- Ransomware Lockbit 2.0
- Semplicità di diffusione del virus nella nella rete di sistemi informatici della regione
  +
- Errore umano, ad es. *"clicca qui per scoprire come guadagnare soldi con i bitcoin"*

- *"l'errore di una persona (che può anche starci, siamo tutti esseri umani in fondo) ha molto probabilmente bloccato un intero sistema che avrebbe dovuto reggere invece che piegarsi"*

https://marcodalprato.medium.com/cosa-ci-ha-insegnato-lattacco-hacker-in-lazio-8ae7ae8dedd5

“Are humans really the weakest link?”

Erik Hollnagel: *“**all human activity** --individually and/or collectively-- **is variable** in the sense that it is **adjusted to the conditions**. The variability is therefore a strength, indeed a necessity, rather than a liability."*

La variabilità dell’attività umana è un ->

Compromesso tra **efficienza** e **rigorosità**

Secondo questo principio, le richieste di produttività tendono a *ridurre la rigorosità*, mentre le richieste di *sicurezza riducono l'efficienza*

https://www.techrepublic.com/article/cybersecurity-pros-are-humans-really-the-weakest-link/

Crittografia

Crittografia

- Disciplina che studia le tecniche per **cifrare un messaggio** in modo tale che solo il **legittimo destinatario** sia in grado di leggerlo
- Requisiti:
  - Cifrare/decifrare messaggi deve essere ragionevolmente efficiente
  - Deve essere “difficile” interpretare un messaggio cifrato da parte di chi non è autorizzato

Principio di base

- **Algoritmo di cifratura**: trasforma un messaggio "in chiaro" ***M*** in un messaggio cifrato ***M'***
- **Chiavi**: necessarie per l'algoritmo di cifratura K_{enc} e decifratura K_{dec}
- **Algoritmo di decifratura**: ricavava dal messaggio cifrato ***M'*** il messaggio in chiaro ***M***

Notazione

- $E(K_{enc}, M)=M'$ <- Funzione di cifratura (encryption)
- K_{enc} <- chiave di cifratura (encryption key)
- M <- messaggio in chiaro (plaintext)
- M' <- messaggio cifrato (ciphertext)

- $D(K_{dec}, M')$ $=M$<- Funzione di decifratura (decryption)
- K_{dec} <- chiave di decifratura (decryption key)

Sistemi crittografici

- Si deve sempre avere che:

$$D(K_{dec}, E(K_{enc}, M)) = M$$

- Se cifro un messaggio M usando la chiave K_{enc} e decifro il risultato usando K_{dec} , devo ottenere nuovamente M

Sistemi crittografici

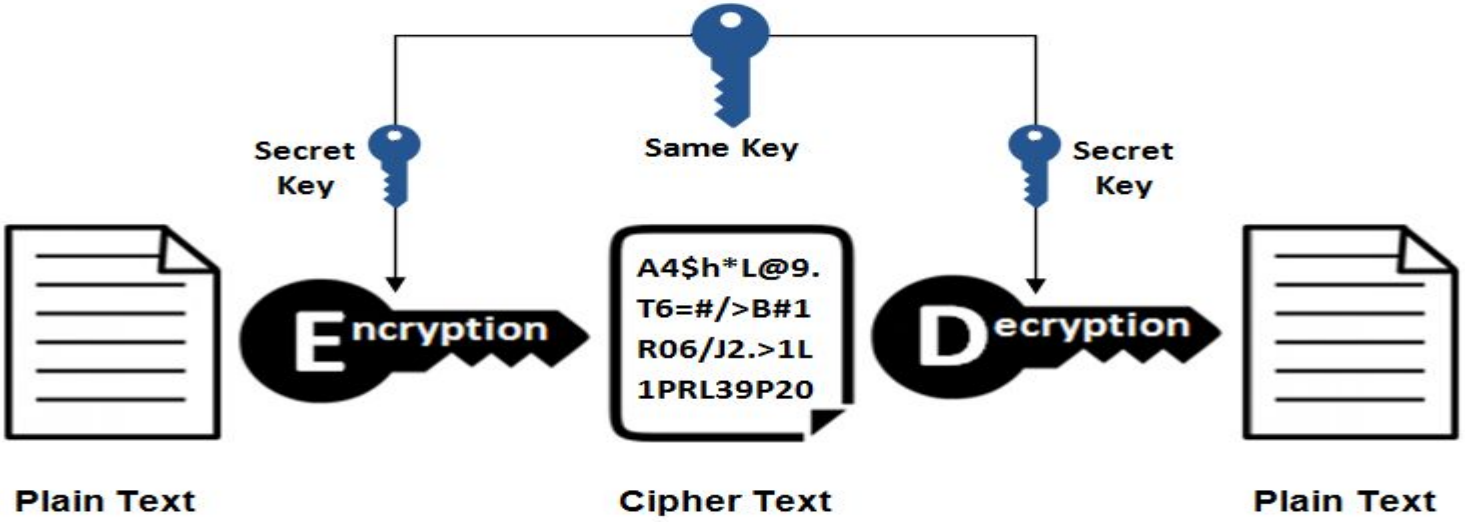
- A chiave segreta (crittografia simmetrica)
  - $K_{enc} = K_{dec}$ <- le chiavi di cifratura e decifratura sono uguali

- A chiave pubblica (crittografia asimmetrica)
  - $K_{enc} \neq K_{dec}$ <- le chiavi di cifratura e decifratura sono diverse

.

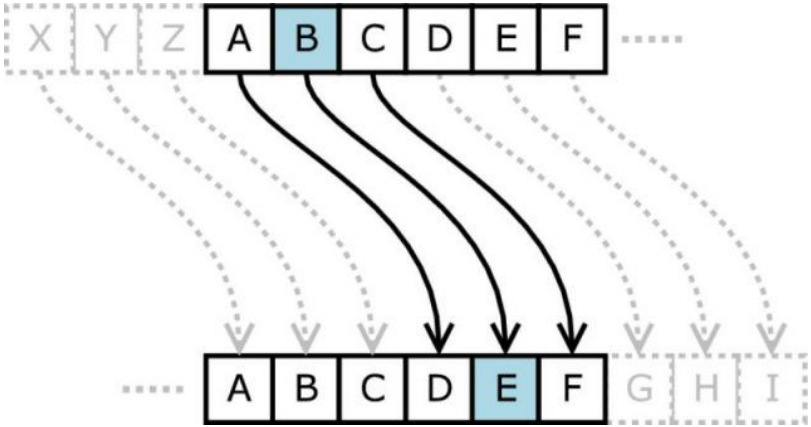
Crittografia Simmetrica

Crittografia Simmetrica



Sistemi crittografici a chiave simmetrica

- Uno dei primi esempi è il “cifrario di Cesare”
  - La chiave *K* è un numero intero
  - Ogni lettera dell'alfabeto viene sostituita da quella che la segue di *K* posizioni



https://www.xarg.org/tools/caesar-cipher

Sistemi crittografici a chiave simmetrica

- Cifrario Polimorfico:
  - Un cifrario che cambia ad ogni cifratura
  - Ogni volta darà risultati diversi
  - Esempio:
    - Usiamo il cifrario di Cesare, ma cambiamo il valore di K secondo una sequenza pre-determinata
    - Ex: {3, -1, 2, 6, …}
    - 1° cifratura di “Hello” ⟶ “Khoor” (shift di 3 lettere)
    - 2° cifratura di “Hello” ⟶ “Gdkkn” (shift di -1)
    - …

DES (*Data Encryption Standard*)

- Progettato da IBM e adottato come standard dal governo USA nel 1977
  - Chiave lunga 56 bit
  - Messaggio diviso in blocchi da 64 bit che vengono cifrati individualmente
- Esistono $2^{56} \approx 7.2 \times 10^{16}$ chiavi
  - Sembra un numero grande, ma un moderno calcolatore può esaminarle tutte in poche ore!
- Una variante (*Triplo DES*) usa chiavi più lunghe e fornisce un livello accettabile di sicurezza

AES (*Advanced Encryption Standard*)

- Adottato come standard nel 2001, sostituisce DES
- Caratteristiche di AES
  - Il messaggio viene scomposto in blocchi da 128 bit che vengono cifrati individualmente
  - Si possono usare chiavi lunghe 128, 192 o 256 bit
  - Esistono $2^{128} \approx 3.4 \times 10^{38}$ chiavi a 128 bit, per cui esaminarle tutte è al momento impraticabile

Riassunto

Key size (bits)	Cipher	Time Required at 10^9 decryptions/μs	Time Required at 10^{13} decryptions/μs
56	DES	$2^{55}\,\mu s = 1.125$ years	1 hour
128	AES	$2^{127}\,\mu s = 5.3 \times 10^{21}$ years	5.3×10^{17} years
168	Triple DES	$2^{167}\,\mu s = 5.8 \times 10^{33}$ years	5.8×10^{29} years
192	AES	$2^{191}\,\mu s = 9.8 \times 10^{40}$ years	9.8×10^{36} years
256	AES	$2^{255}\,\mu s = 1.8 \times 10^{60}$ years	1.8×10^{56} years

Fonte: W. Stallings and L. Brown, *Computer Security: Principles and Practice*, Pearson; 2 edition, 2011, ISBN 978-0132775069

Quanto tempo serve per esplorare tutto lo spazio delle chiavi?

Tipo password	Lunghezza	Numero di password possibili	Tempo brute force a 10^9 pwd/s	Tempo brute force a 10^{13} pwd/s
alfa	8	$26^8 \approx 2.09 \times 10^{11}$	3.48 min	Istantaneo
alfa	10	$26^{10} \approx 1.41 \times 10^{14}$	1.64 giorni	14 sec
alfa	12	$26^{12} \approx 9.54 \times 10^{16}$	3.03 anni	2 ore
alfa+ALFA+num	12	$52^{12} \approx 3.23 \times 10^{21}$	1.0×10^5 anni	10 anni
alfa+ALFA+num	15	$62^{15} \approx 7.69 \times 10^{26}$	2.4×10^{10} anni	2.4×10^6 anni

Crittografia Simmetrica

- Punti a favore
  - Efficienza degli algoritmi di cifratura e decifratura
- Punti deboli
  - Distribuzione delle chiavi – Necessario un meccanismo sicuro per decidere la chiave da usare tra due parti
  - Scalabilità – Per ogni coppia di interlocutori, serve una chiave diversa → il numero di chiavi cresce in maniera esponenziale
  - Sicurezza limitata – Fornisce confidenzialità, ma limitata autenticità e non garantisce non-repudiabilità

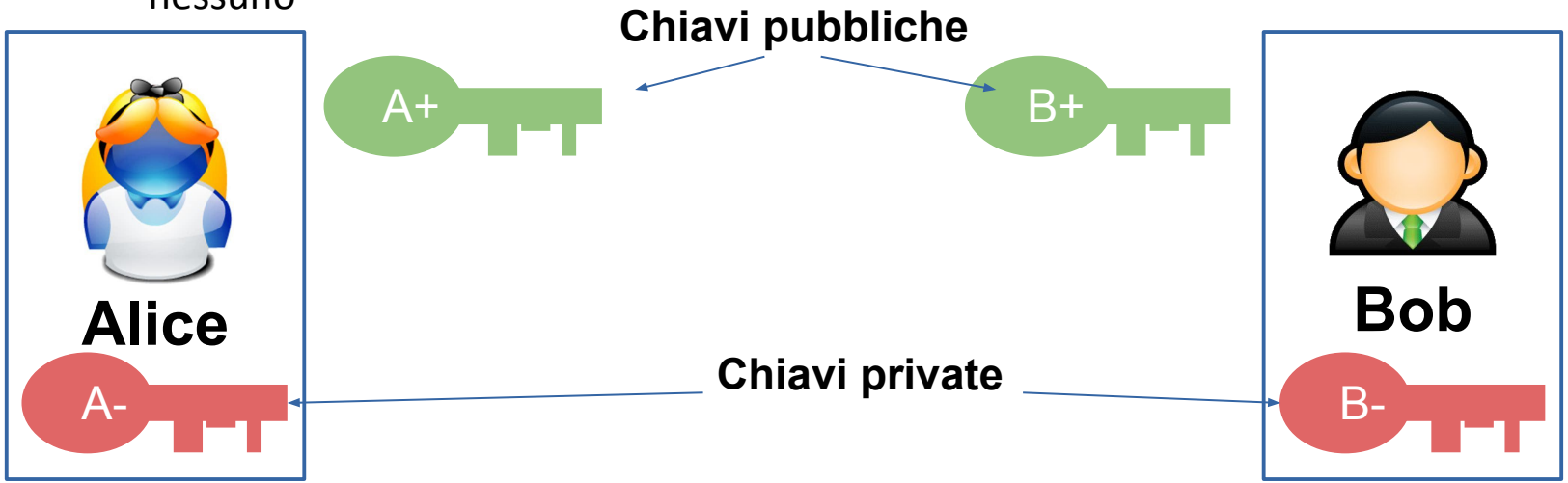
Crittografia Asimmetrica

Crittografia a chiave pubblica (asimmetrica)

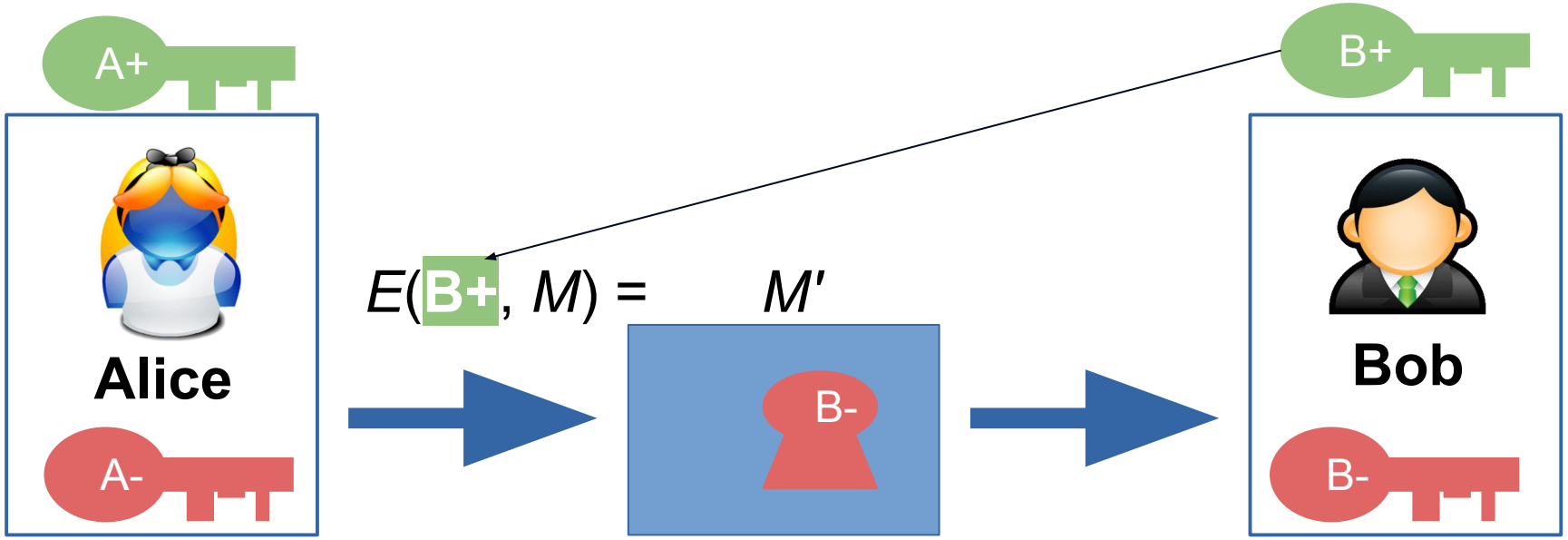
- Introdotta nella seconda metà degli anni '70 da W. Diffie e M. Hellman
- Requisiti
  - Ciascun utente ha due chiavi
  - Se si usa una delle due chiavi per cifrare un messaggio, l'altra (e solo quella) può essere usata per decifrarlo
  - È impossibile derivare una delle due chiavi anche se si conosce l'altra (le due chiavi sono totalmente indipendenti)

Alice e Bob

- Ogni utente possiede due chiavi
  - Una è pubblica, e viene resa disponibile a chiunque
  - L'altra è privata, e l'utente deve custodirla gelosamente e non comunicarla a nessuno



Alice e Bob



1 Alice cifra il messaggio *M* con la chiave pubblica di Bob B+

2 Bob decifra il messaggio *M'* con la propria chiave privata B-

Demo

http://www.narbona.net/uah/Crypto.html

Applicazioni

visti

- **Autenticità**
  - Validare la sorgente di un messaggio per garantire l'autenticità del mittente
- **Confidenzialità**
  - Soggetti non autorizzati non possono accedere al dato

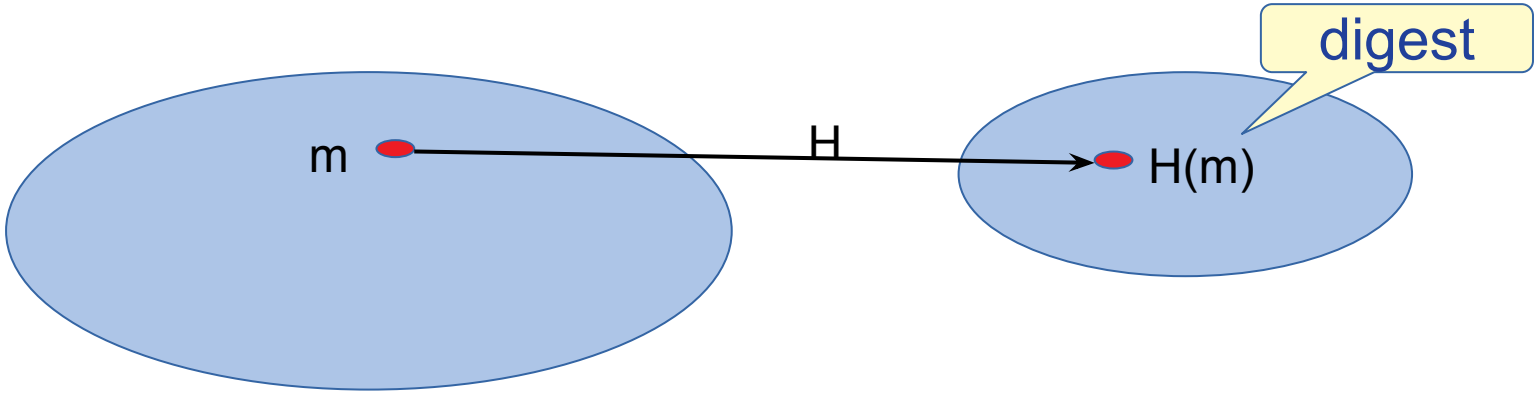
- **Integrità**

vedi più avanti

  - Assicurarsi che il messaggio non sia stato alterato durante la trasmissione, accidentalmente o intenzionalmente
- **Non-repudiabilità**
  - Un mittente non può inviare il messaggio e negare successivamente di averlo fatto

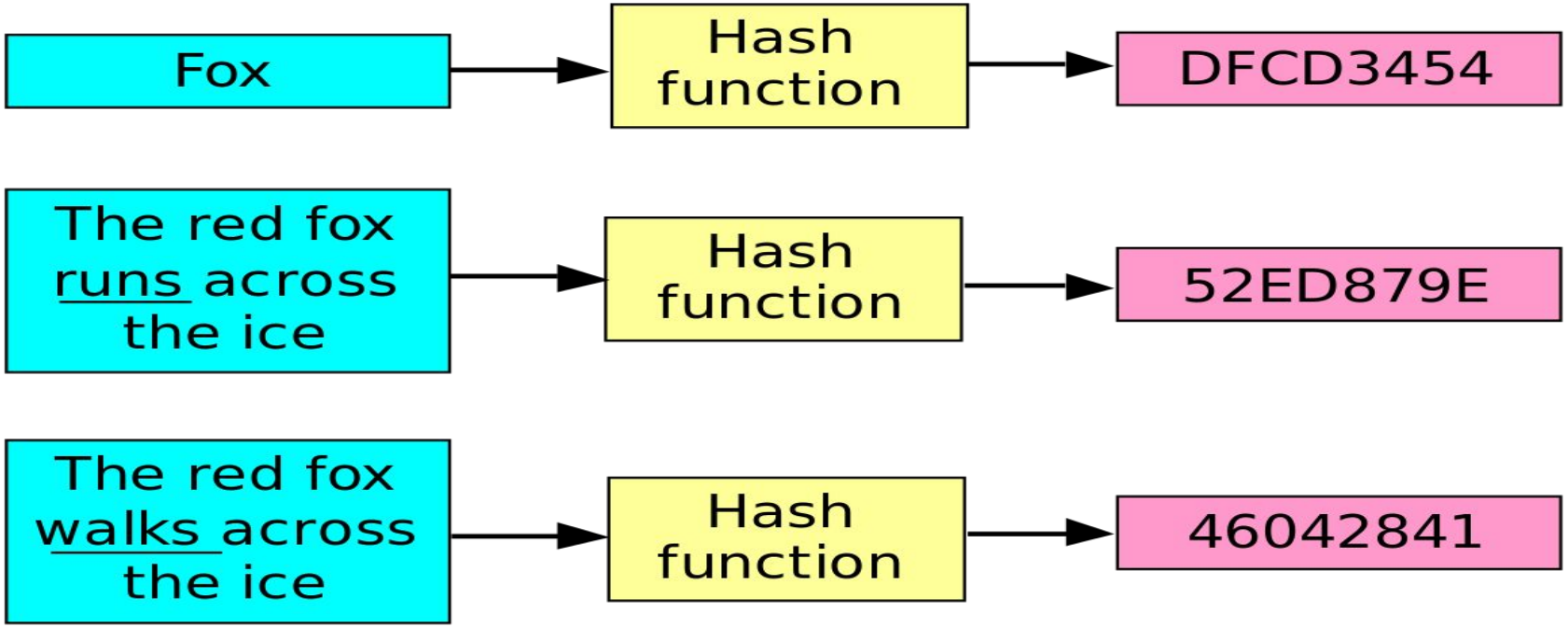
Funzioni Hash

Funzioni Hash



Messaggio *M* ----------> hash(*M*) ----------> *digest*

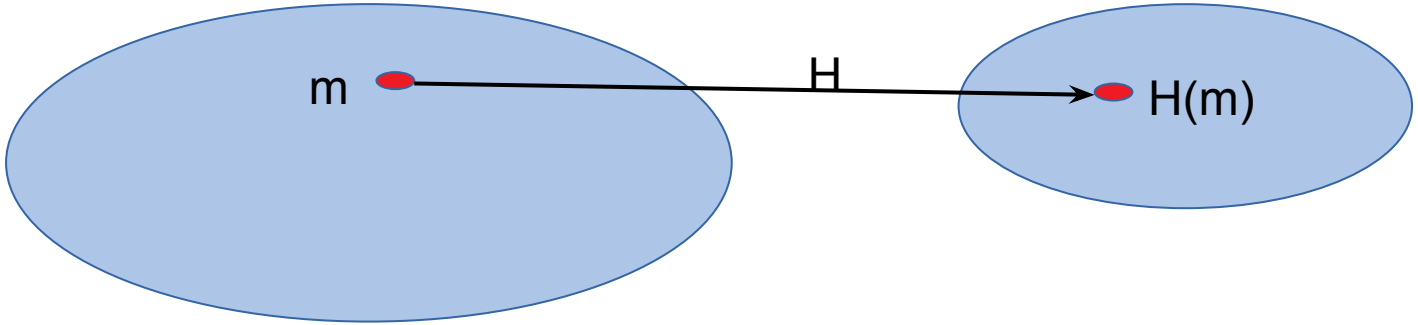
Funzioni Hash



Funzioni Hash Crittografiche

- Classe speciale di funzioni hash
  - In queste diapositive quando ci riferiamo a "funzione hash", intendiamo in realtà una "funzione hash crittografica".
- Applicazioni
  - Verifica dell'integrità di messaggi e file
  - Firma digitale
  - Verifica della password
  - Proof-of-work

Funzioni Hash (Crittografiche)



- Computationally efficient
- Security properties
  - Collision-free
  - Hiding
  - Crypto-puzzle friendly

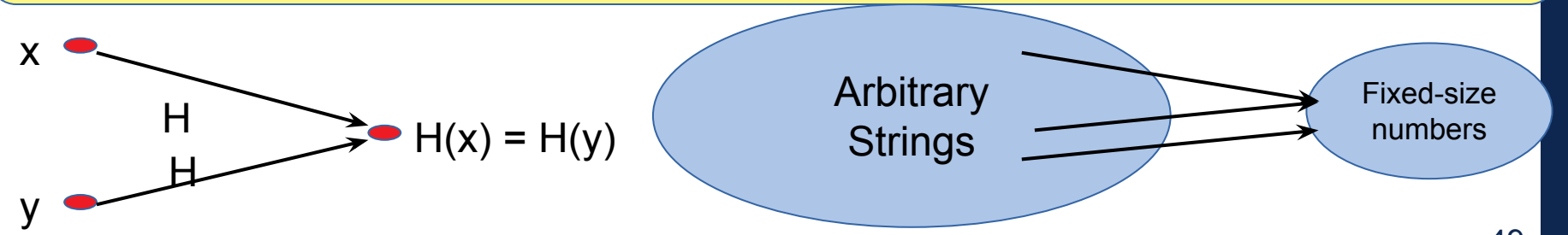
1) Hash Property: Collision-free

Cannot ~~find~~ have collisions

Pigeon principle: se n oggetti sono messi in m contenitori, con n > m, allora almeno un contenitore deve contenere più di un oggetto

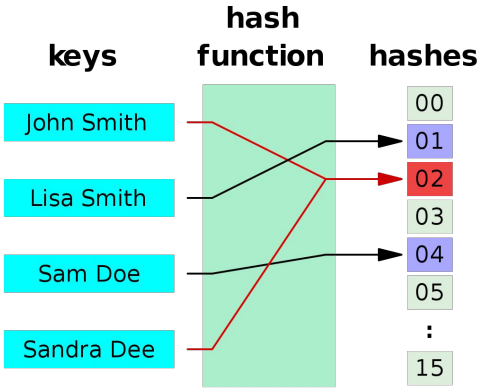
Le collisioni esistono!

Non si possono trovare *x* e *y* tali che *x != y* e *H(x)=H(y)*



1) Trovare una collisione

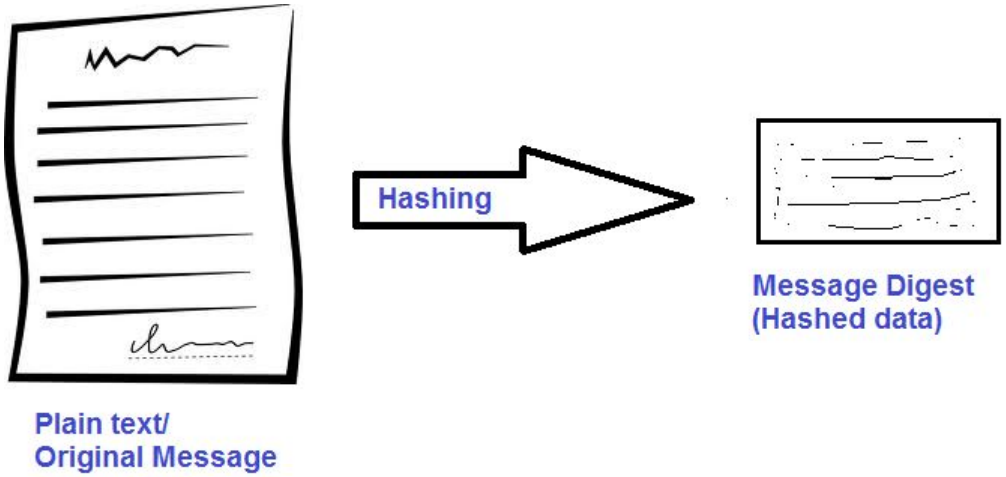
- Se l'output è lungo 256 bits, abbiamo 2^{256} valori possibili
- Possiamo solamente tirare ad indovinare
- Scegliendo 2^{130} valori random distinti, il 99.8% delle volte avverrà una collision



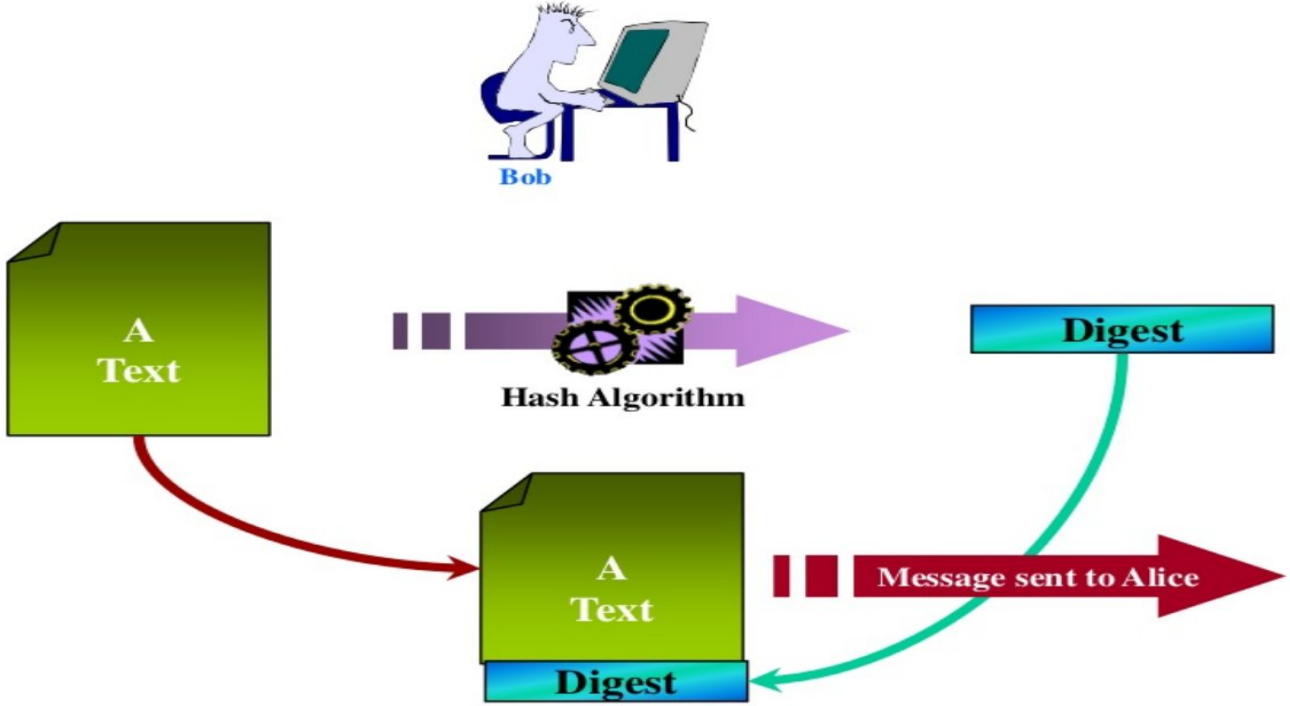
- Numero delle stelle nell'universo = ~ 2^{77} !!

1) Applicazione: Hash Digest Identification

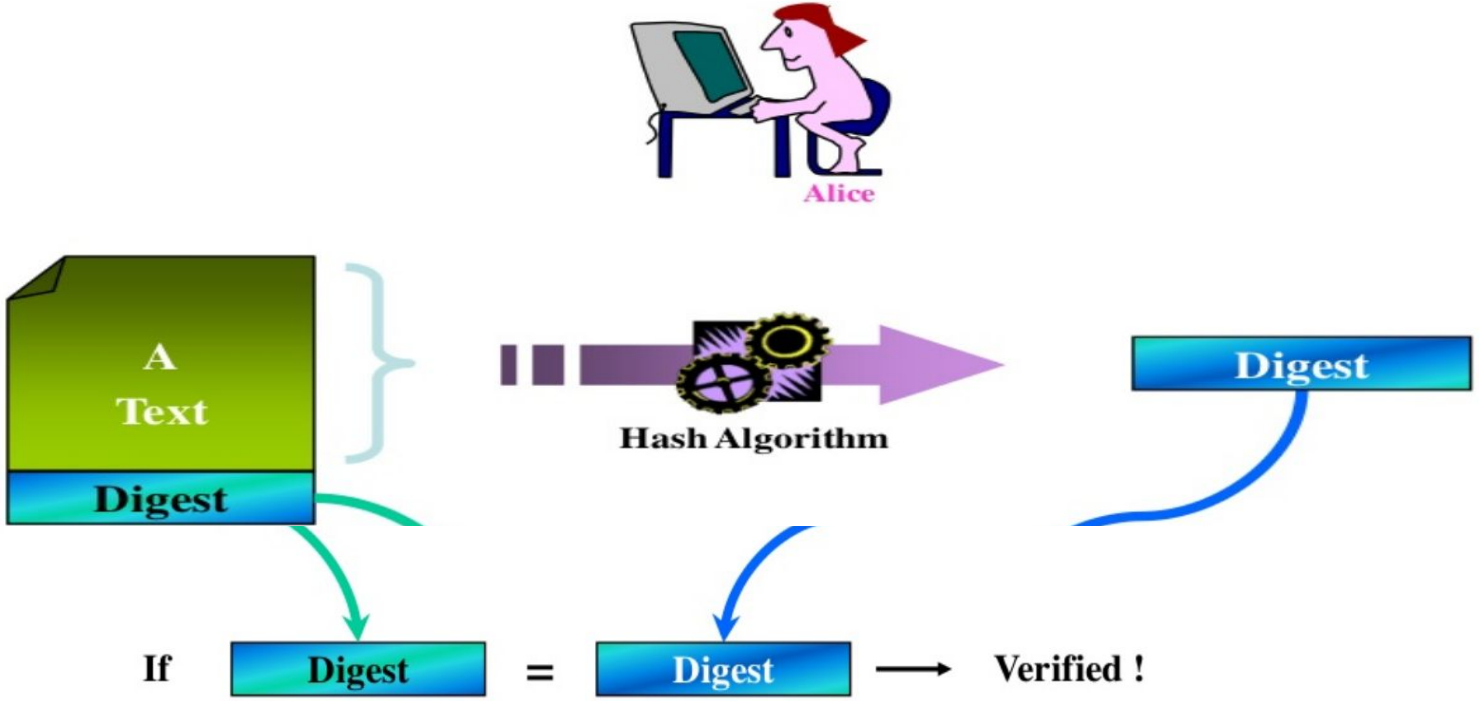
- Se conosciamo $H(x) = H(y) \rightarrow$ possiamo assumere che $x = y$
- Per riconoscere un file, basta ricordare il suo hash
  - Utile perché l'hash è piccolo come dimensione (256 bit)



1) Applicazione: Integrità dei dati



1) Applicazione: Integrità dei dati - Verifica



2) Hash Property: Hiding

- Dato *H(x)* non deve essere facile trovare *x*

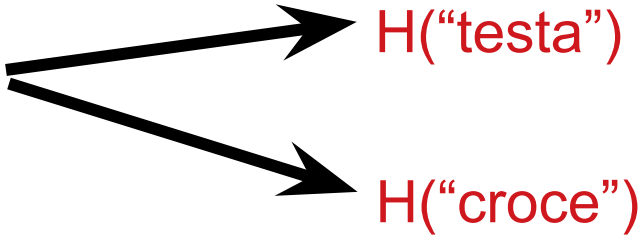
2) Hash Property: Hiding

- Dato *H(x)* non deve essere facile trovare *x*

H(moat) = 98E2W0dja8A...aslOSa216F3

H(m**ao**t) = E6712e3awa4...gz3wle3A9C9

2) Esempio in cui Hiding fallisce



- Guardando i risultati dell'hash, è facile capire se x era una croce o una testa
  - Solo due input!
  - Basta fare l'hash dei due input e guardare il risultato dell'hash

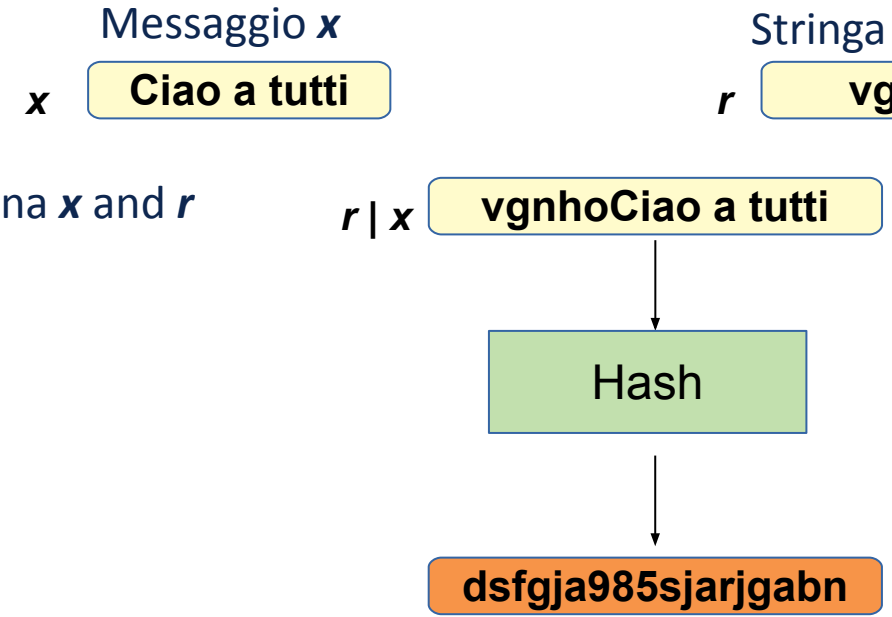
2) Hash Property: Hiding

- Funziona solo quando abbiamo
  - Un **grande insieme** di possibili valori di x
  - Non ci sono particolari x che sono **più probabili**
  - Non ci sono particolari x che sono **più interessanti di altri**

2) Applicazione: Commitment

- Vuoi vincolarti a un valore, rivelarlo più tardi
  - "sigillare un valore in una busta" ora, e
  - "aprire la busta" più tardi

3) Hash Salting

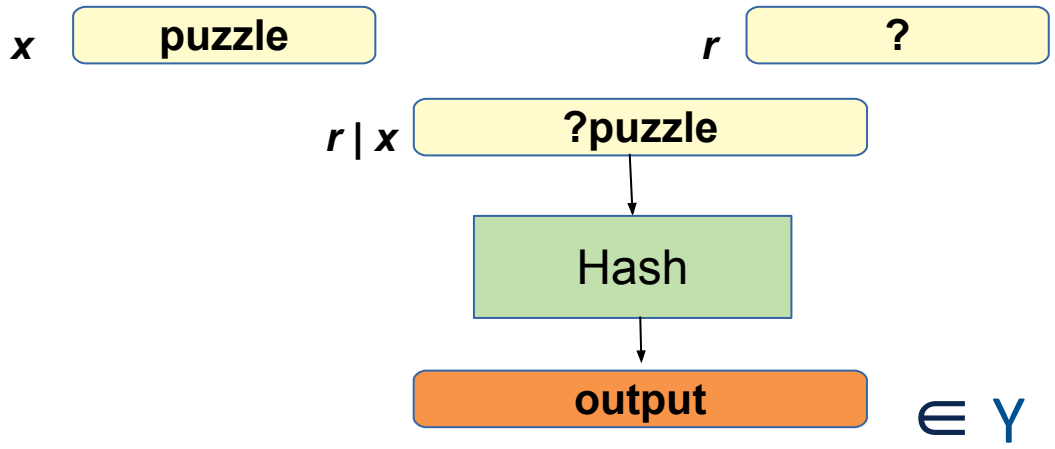


Questo può essere utilizzato per ideare cripto-puzzle

Dato il risultato della funzione hash H(*r* | *x*), **risulta difficile trovare *x***

3) Hash Property: Crypto-puzzle friendly

- Dato un puzzle x ed un insieme **target** Y,
- trova una stringa random r tale che -> **H(r | x) ∈ Y**



3) Hash Property: Crypto-puzzle friendly

- Dato un puzzle **x** ed un insieme **target** Y,
- trova una stringa random **r** tale che -> **H(r | x) ∈ Y**

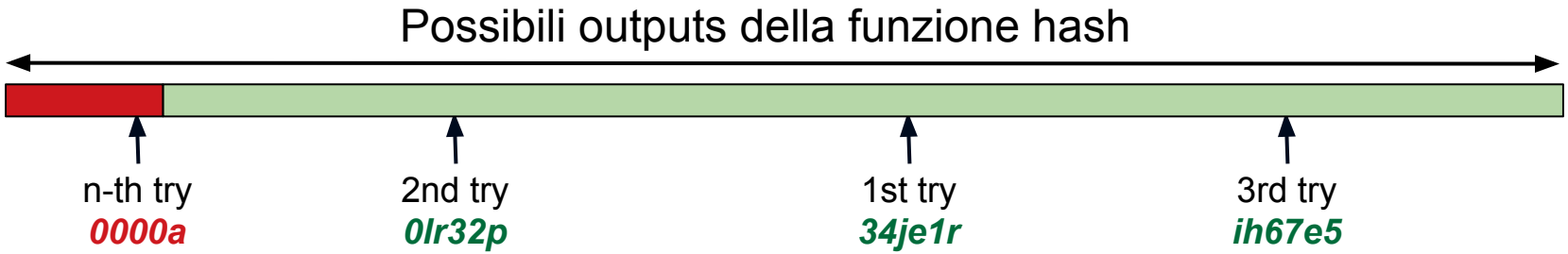
3) Hash Property: Crypto-puzzle friendly

target
Y

Possibili outputs della funzione hash

3) Hash Property: Crypto-puzzle friendly

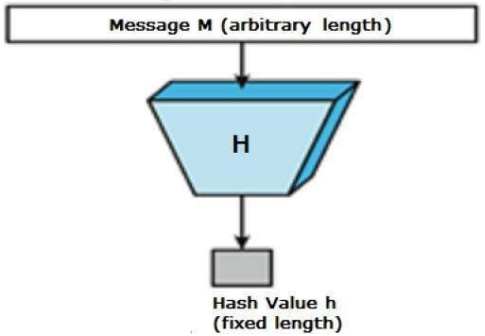
- Dato un puzzle x ed un insieme **target** Y,
- trova una stringa random r tale che -> **H(r | x) ∈ Y**



Più è piccolo **Y**, più è difficile il puzzle

Funzioni Hash: Schemi

MD5 (128 bits)
SHA-1 (160 bits)
SHA-256/384/512 (256/384/512 bits)



Insicuri:

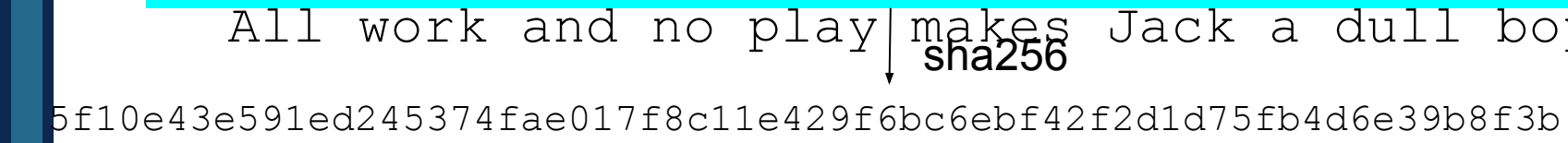
- **MD5:** A 2013 attack by Xie Tao, Fanbao Liu, and Dengguo Feng breaks MD5 collision resistance in 2^{18} time. This attack runs in less than a second on a desktop computer.
- **SHA-1**: Some theoretical attacks - not yet collisions

Viene consigliato l'uso di **SHA-2 (256 or 512) or SHA-3/Keccak**

Esempio (SHA-256)

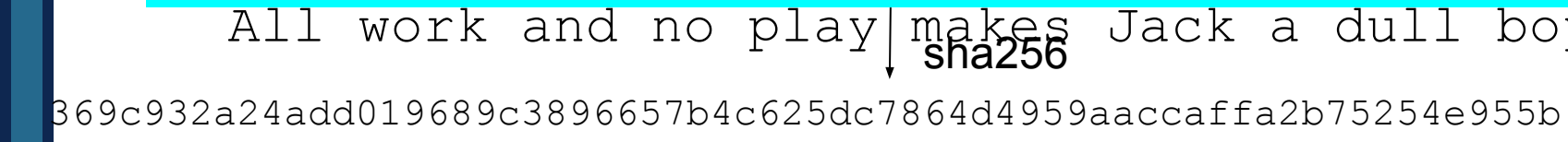
msg1.txt

```
All work and no play makes Jack a dull boy
 All work and no play makes Jack a dull boy
```



msg2.txt

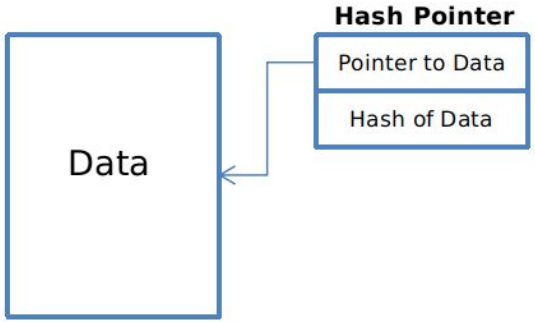
```
All work and no play makes Jack a dull boy
 All work and no play makes jack a dull boy
```



Hash pointers e Strutture Dati

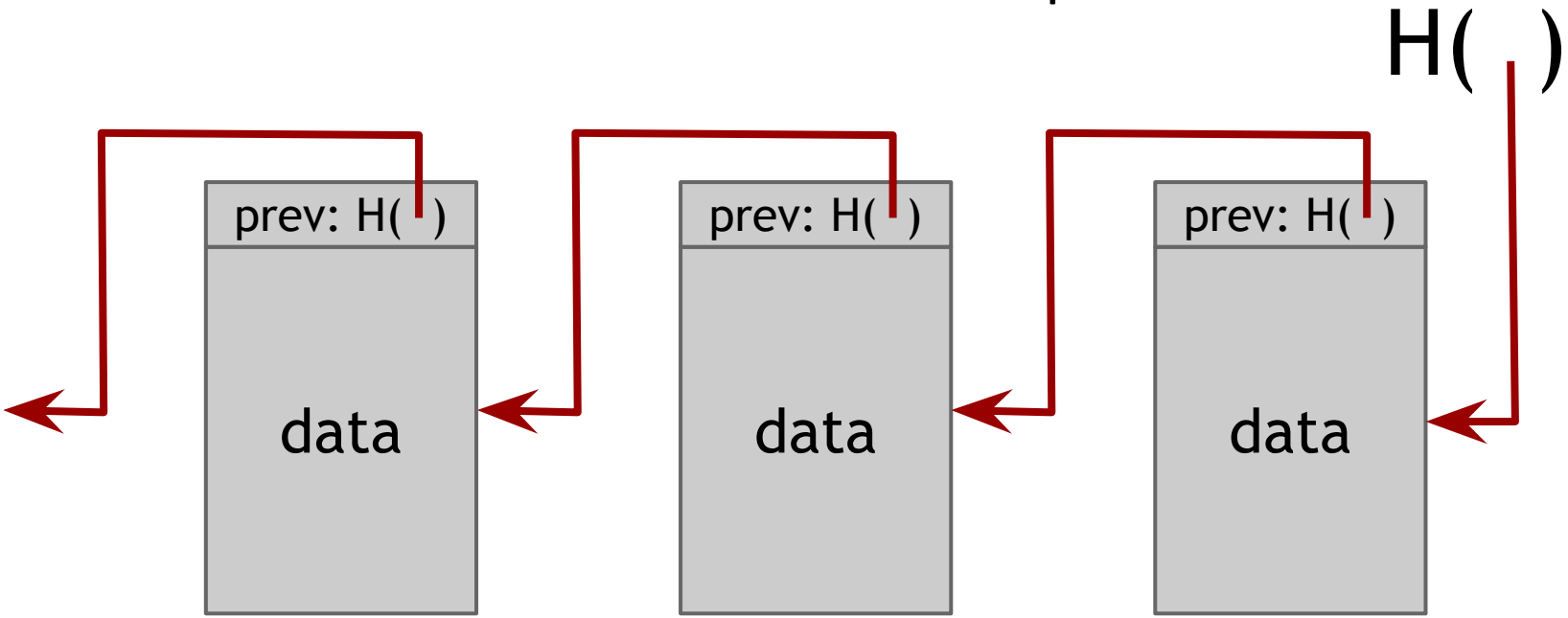
Hash Pointer

- L'Hash Pointer è il risultato di una funzione hash crittografica usato per "puntare" (riferirsi a) dati conservati altrove
  - I dati vengono salvati e viene calcolato il loro digest
  - Il digest viene usato altrove per verificare che i dati salvati non vengano alterati

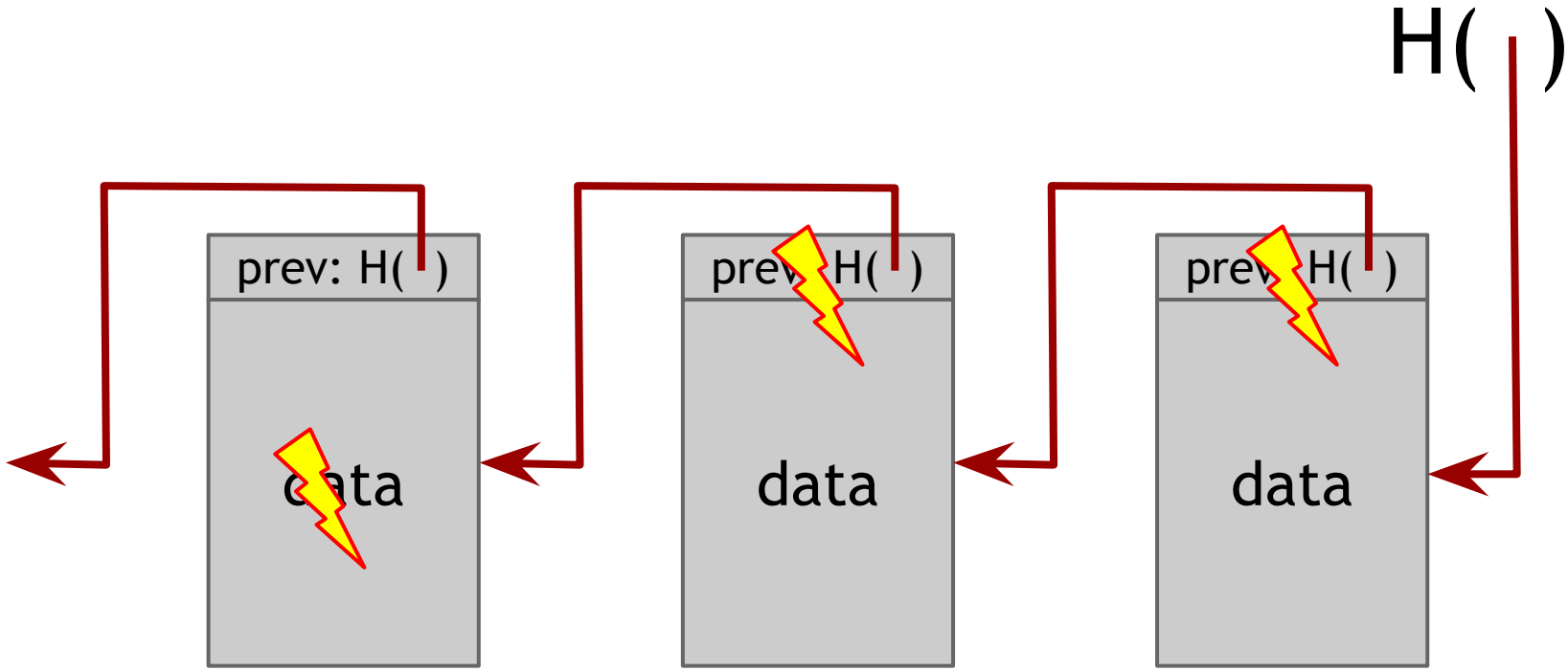


Key Idea

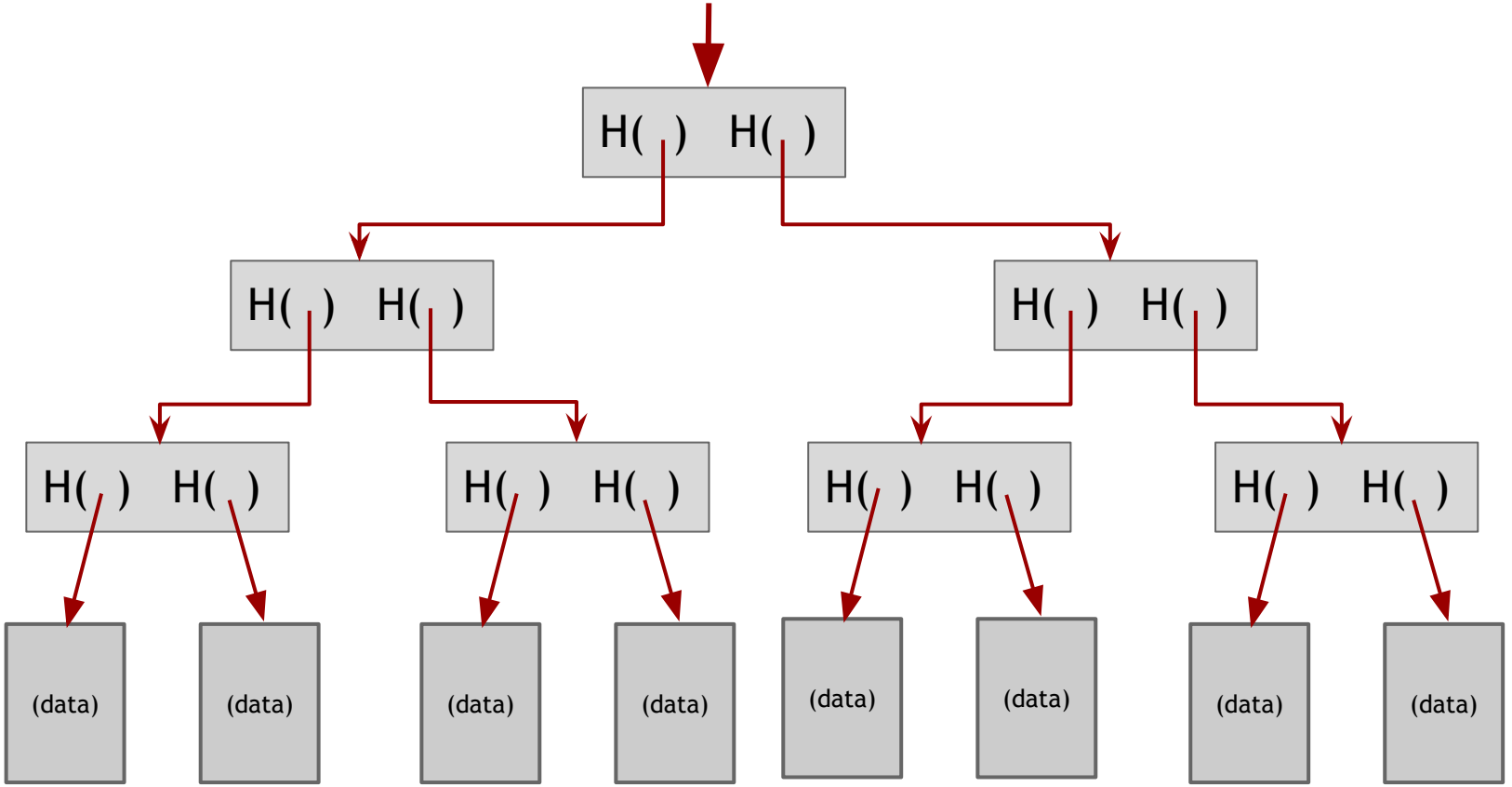
- Costruire strutture di dati con hash pointers



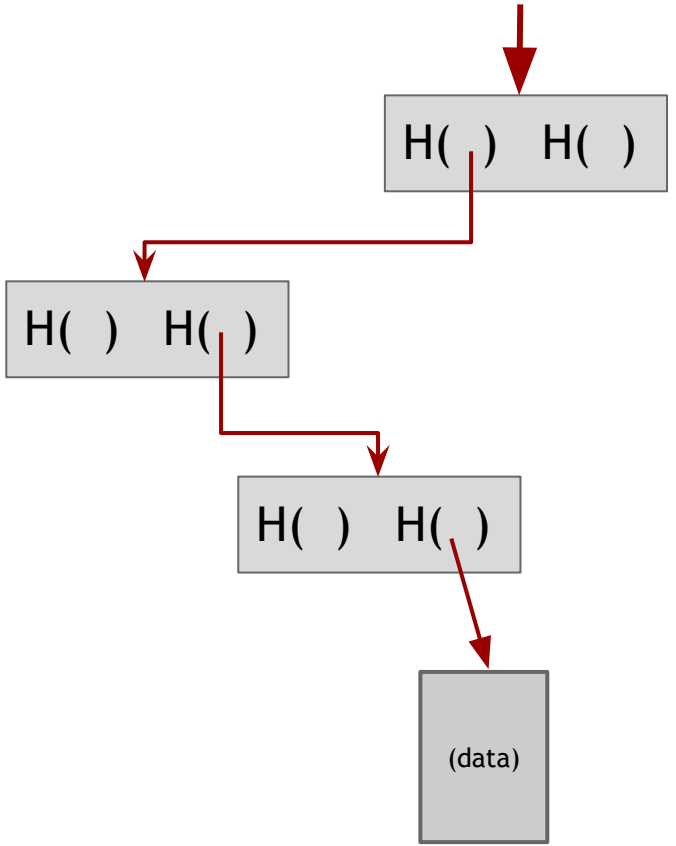
Tampering



“Merkle tree”

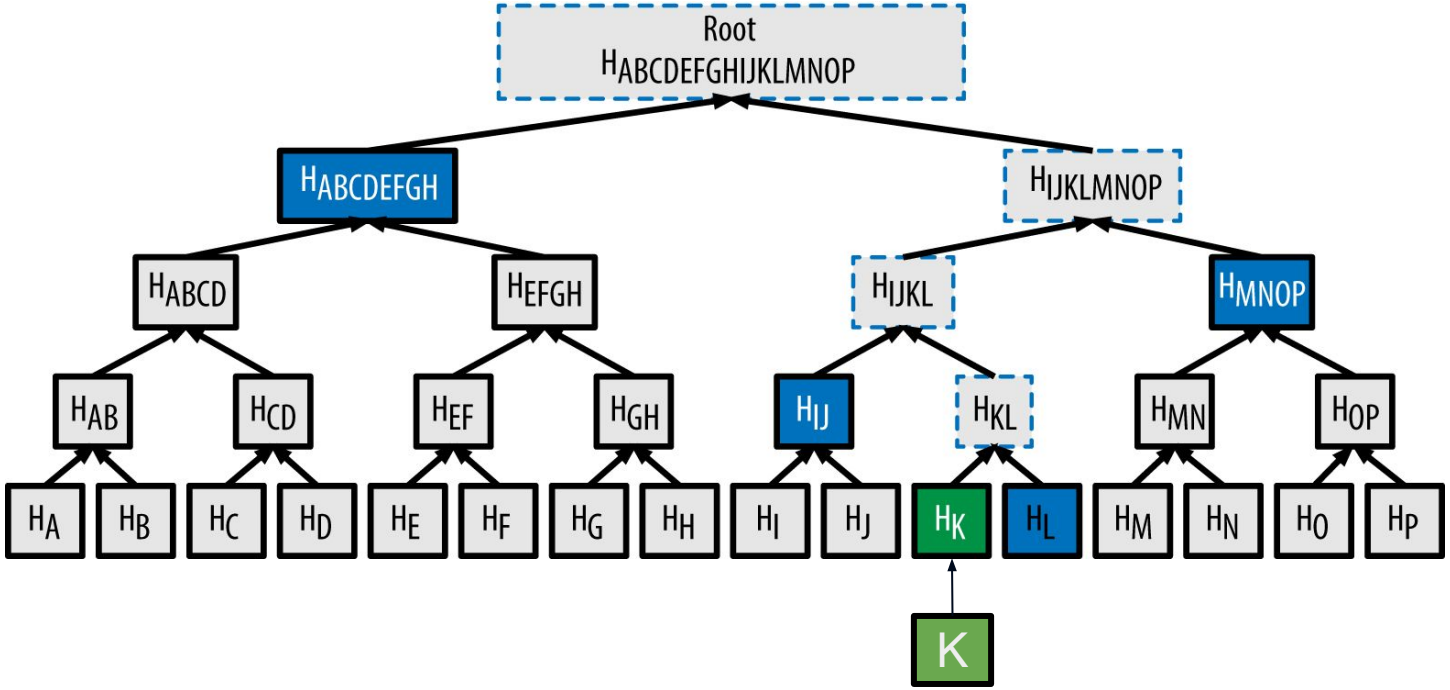


La proof in un Merkle tree



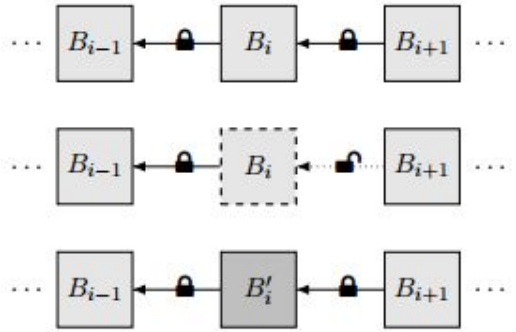
O(log n)

La proof in un Merkle tree



Chameleon hashing

- Il *Chameleon hashing* utilizza funzioni hash abbinate a una ***coppia chiave pubblica e chiave privata***.
- **Chiave pubblica** -> può essere utilizzata da chiunque per verificare la funzione hash.
- **Chiave privata** -> Trovare una collisione di due hash è impossibile per chi non conosce la chiave privata.
- Tuttavia, chi conosce la chiave privata può trovare facilmente una collisione.
- Questo permette di **cambiare l'input originale della funzione hash originale con un altro input**

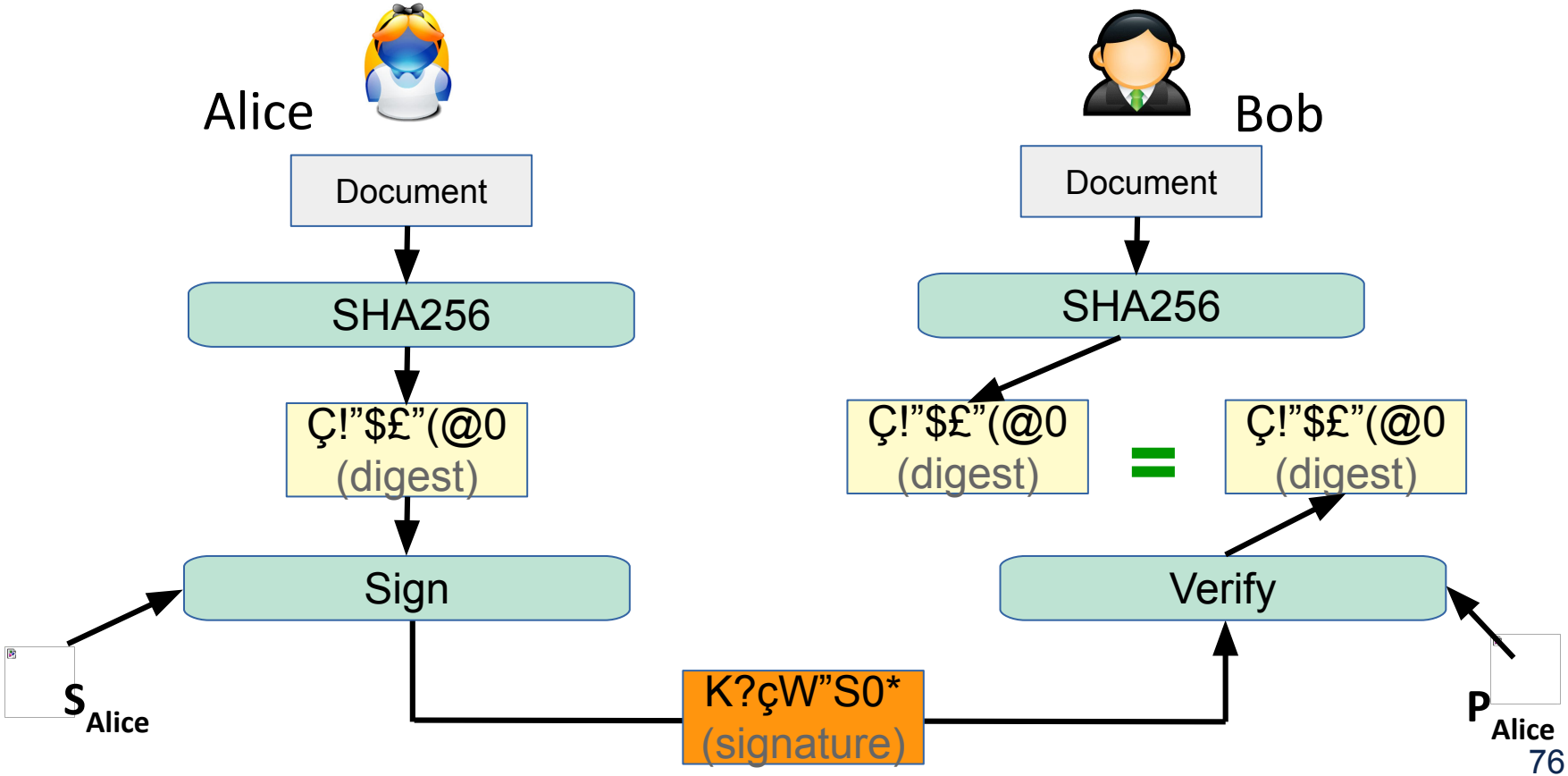


Firma Digitale

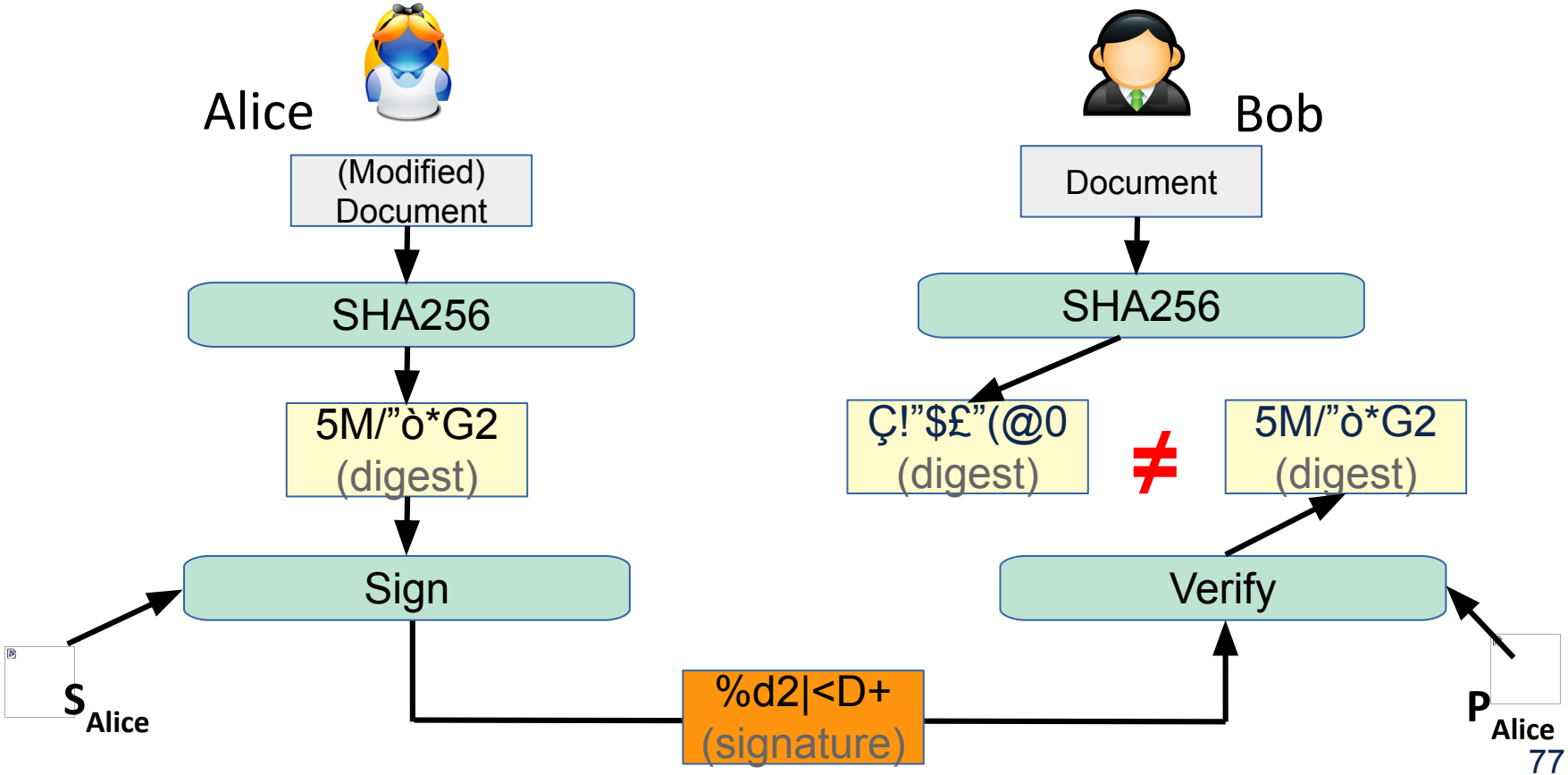
Firma Digitale

- Schema per la verifica dell'**autenticità** dei messaggi digitali (documenti).
- Impiega la crittografia **asimmetrica**
- **Integrità**: garantisce che il messaggio non sia stato alterato durante il trasporto (utilizzando il digest)
- **Autenticazione**: una firma digitale valida dà al destinatario un motivo molto forte per credere che il messaggio sia stato creato da un mittente conosciuto.

Alice firma un documento per Bob



Alice firma un documento (alterato) per Bob

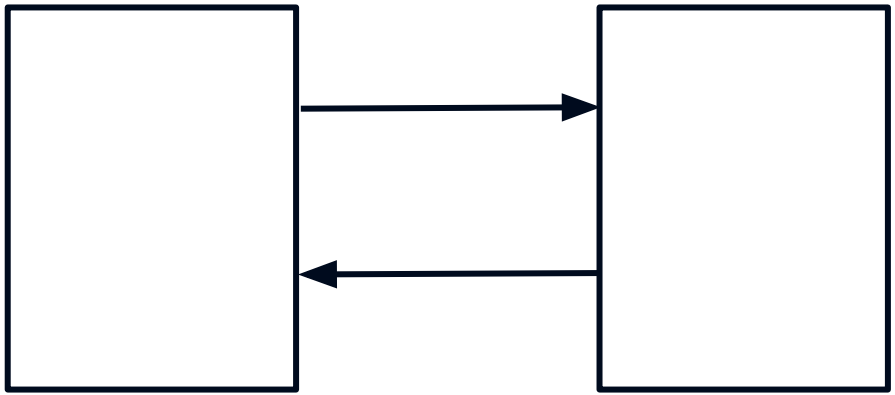


Introduzione ai Sistemi Distribuiti

Sistema Distribuito

Sistema informatico costituito da un insieme di **processi** interconnessi tra loro in cui le comunicazioni avvengono solo esclusivamente tramite lo scambio di opportuni messaggi.

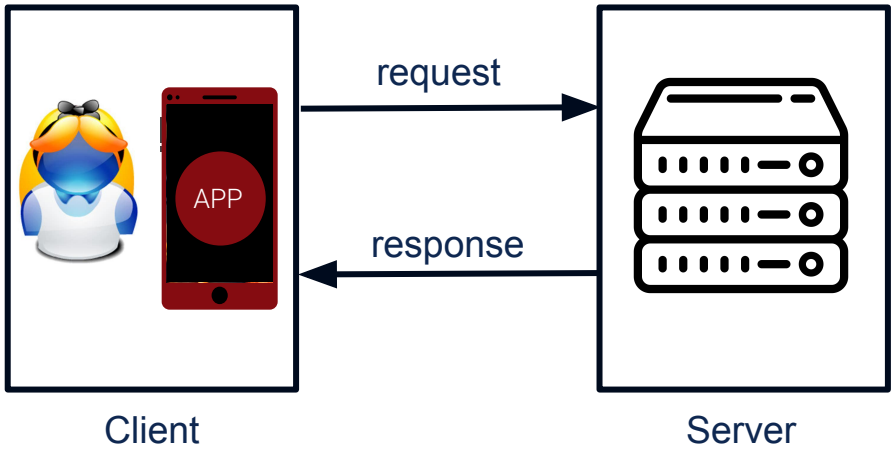
Nodo



Qualsiasi **dispositivo** hardware del sistema in grado di **comunicare** con gli altri dispositivi che fanno parte della rete

Dispongono di una memoria propria, di un proprio sistema operativo e di risorse locali

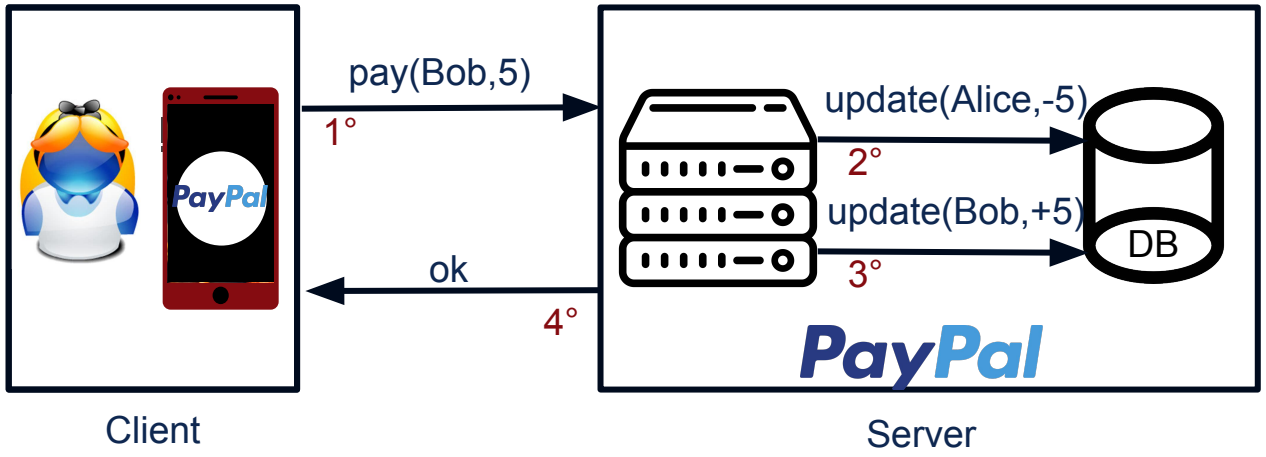
Architettura Client/Server



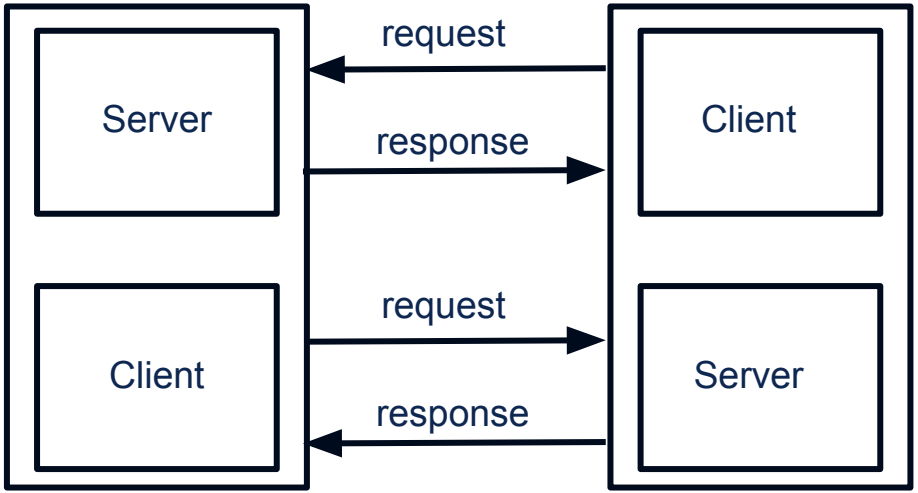
Un'architettura di sistema è il **modello concettuale** che definisce la struttura, il comportamento e più prospettive di un unico sistema

Architettura Client/Server
Esempio

Alice
paga
Bob
5 euro



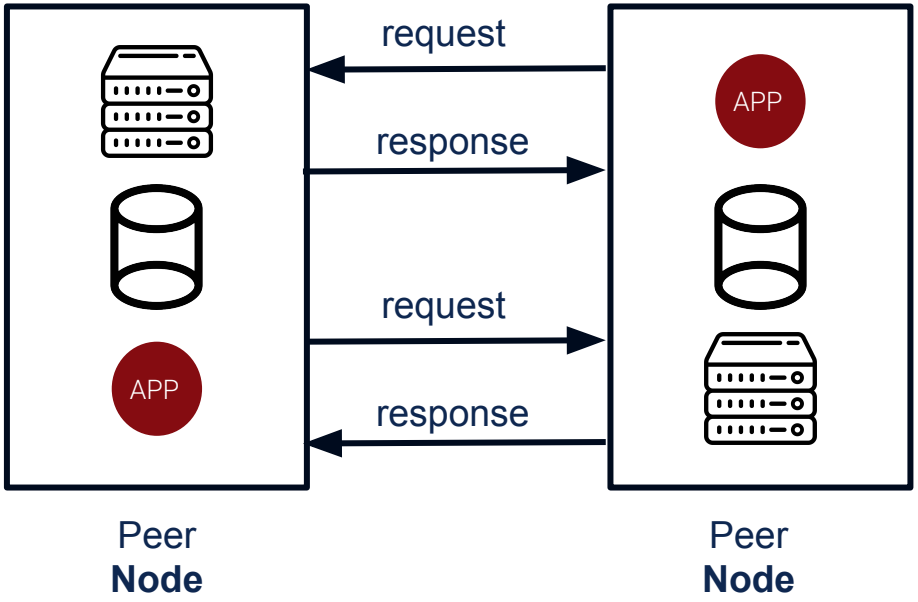
Architettura Client/Server



I nodi Peers sono **Client** e **Server** simultaneamente

Architettura Peer to Peer (P2P)

I nodi Peers sono **Client** e **Server** simultaneamente

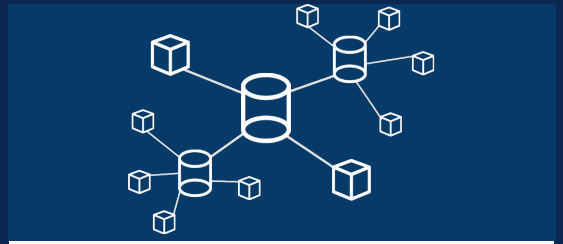


Sistema Distribuito

Sistema informatico costituito da un insieme di **processi** interconnessi tra loro in cui le comunicazioni avvengono solo esclusivamente tramite lo scambio di opportuni messaggi.

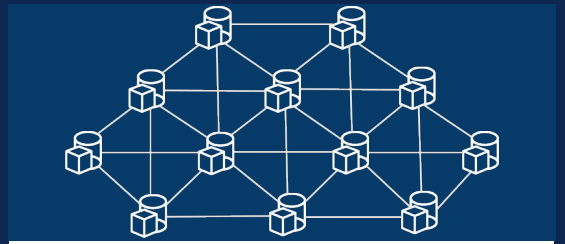
Centralizzato

Un nodo si occupa di tutto

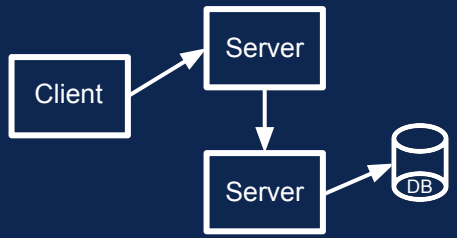


Distribuito Semi-centralizzato

Nodo distribuisce lavoro a sotto-nodi



Distribuito De-centralizzato (P2P)



Blockchain

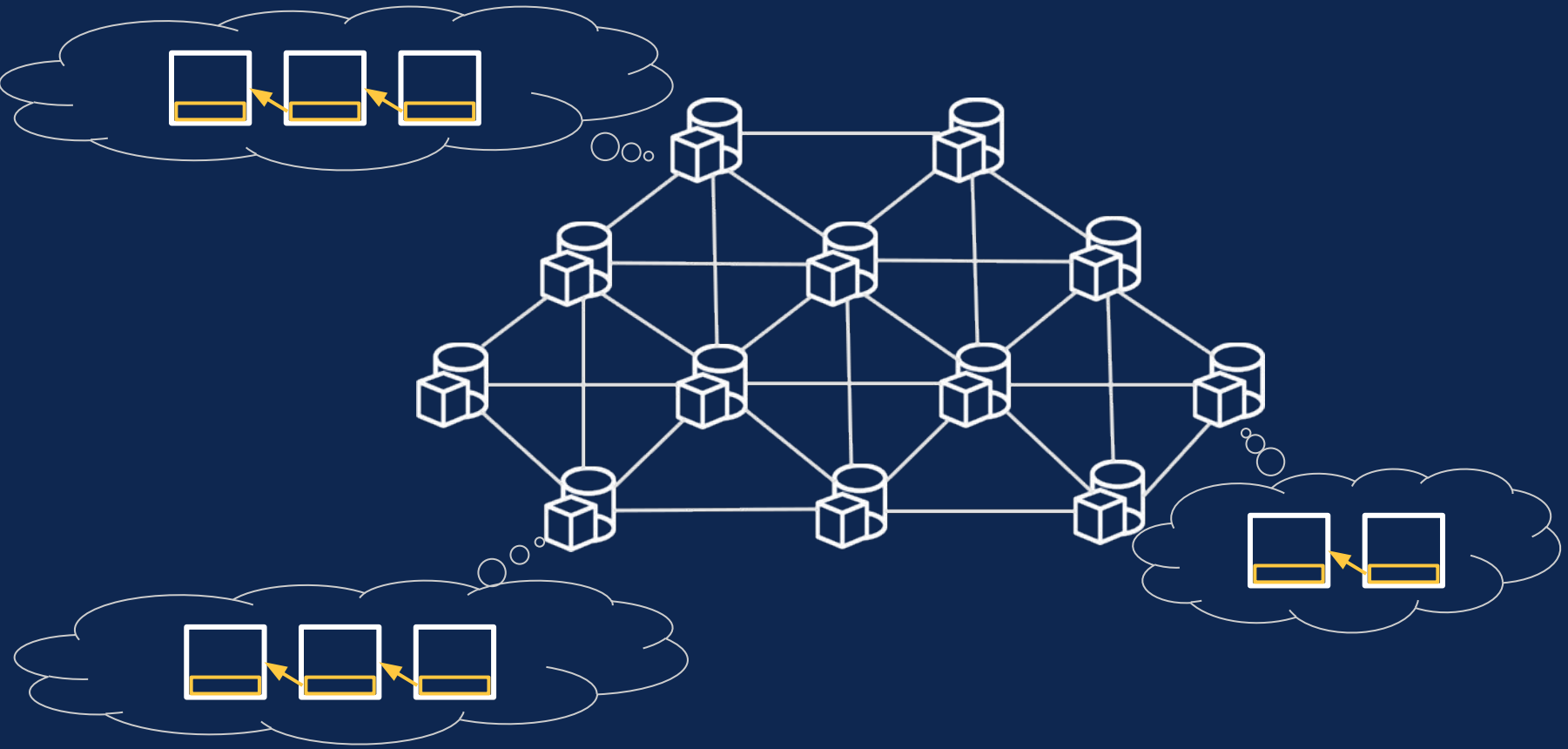
BLOCKCHAIN

Sistema Distribuito basato su una rete di nodi P2P

- È una tecnologia che fa parte del regno delle DLTs: **Distributed Ledger Technologies**
- Nelle DLTs un registro viene distribuito tra i nodi di una rete P2P, che aggiornano la loro **copia locale** secondo un unico meccanismo di consenso
- Una blockchain è una DLT in cui il registro assume la forma di un **insieme di blocchi di dati (relativamente) ordinati cronologicamente**

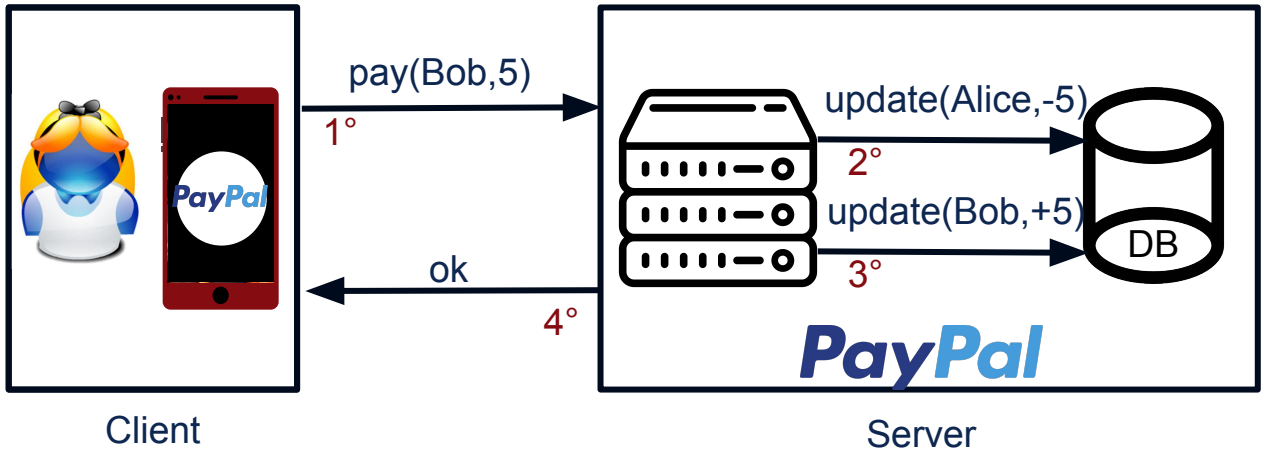
BLOCKCHAIN

Sistema Distribuito basato su una rete di nodi P2P



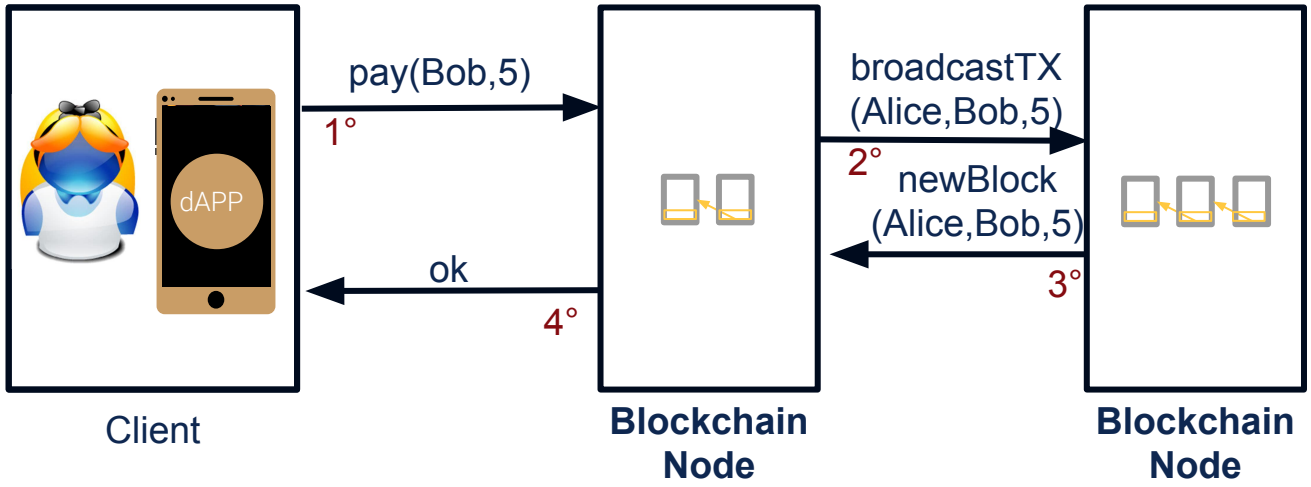
Comparazione Blockchain e Client/Server

Alice
paga
Bob
5 euro



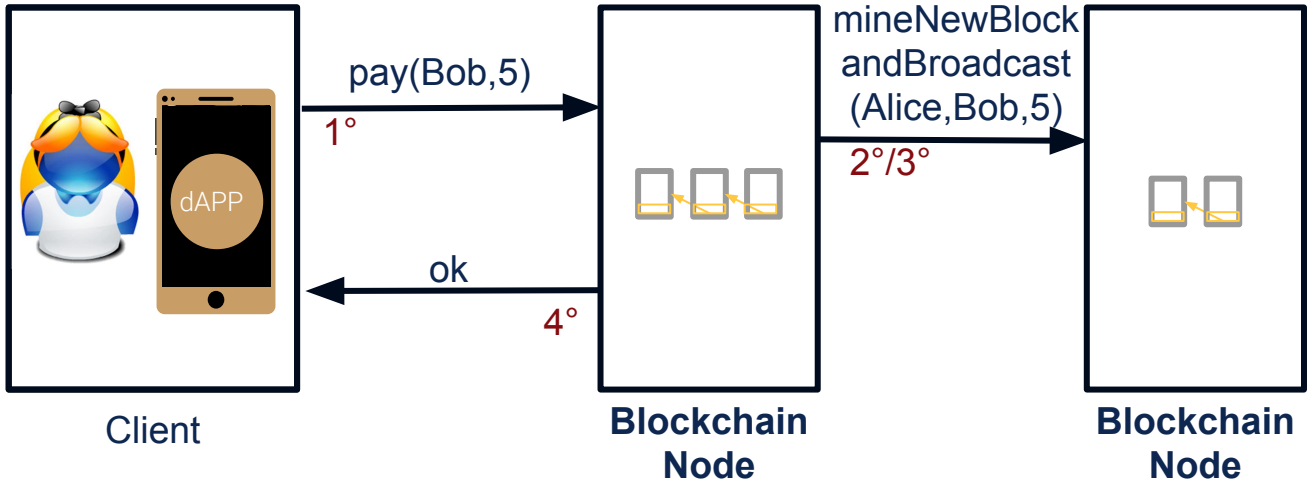
Comparazione Blockchain e Client/Server

Alice
paga
Bob
5 bitcoin



Comparazione Blockchain e Client/Server

Alice
paga
Bob
5 bitcoin

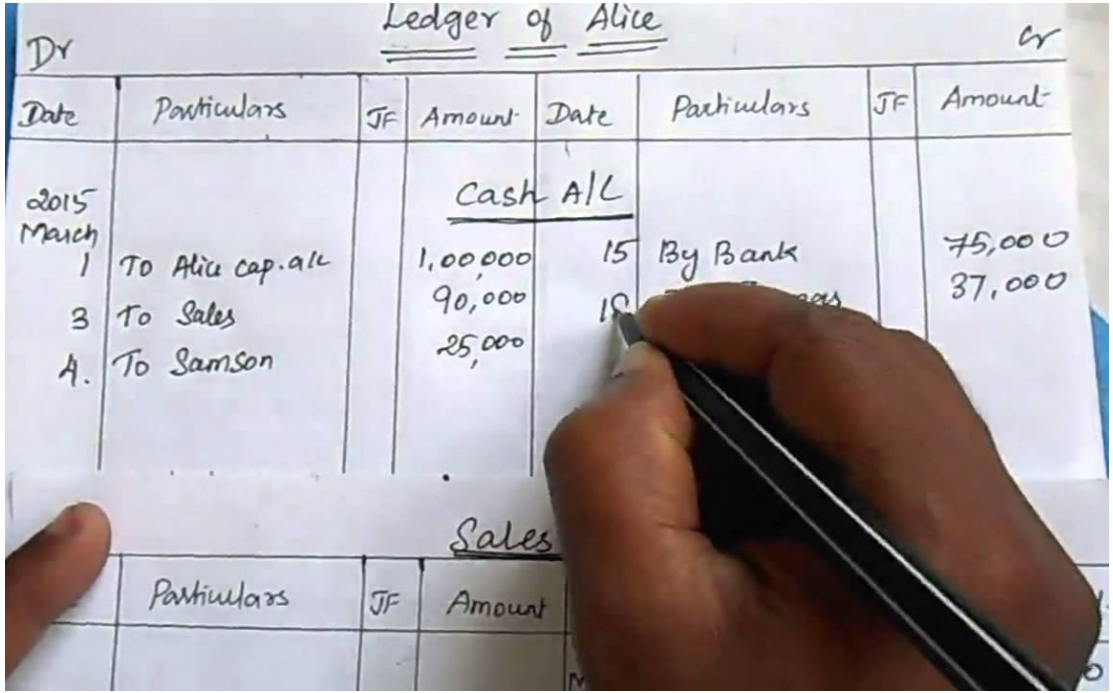


Blockchain

- Cosa scrivere sul registro⟶**transazioni**

- Struttura del registro⟶**chain of blocks**

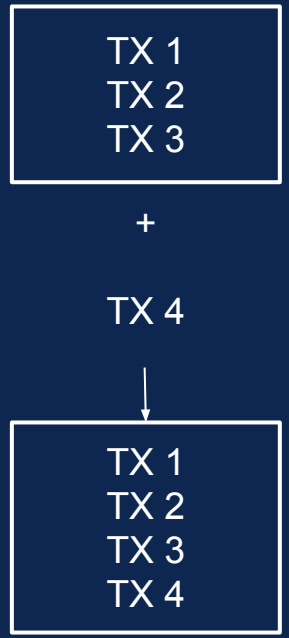
Registro → Libro Mastro



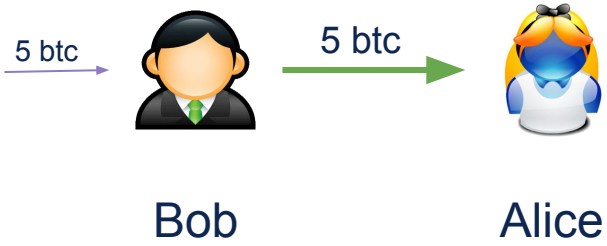
https://asecuritysite.com/encryption/ethadd

Transazioni

- Se il registro mantiene lo stato del sistema →
  allora una transazione è l'operazione che modifica questo stato
- Lo stato del sistema in un certo momento (snapshot) è un elenco di transazioni
- Una nuova transazione si riferisce ad una precedente e aggiorna la stato del sistema
- Una transazione valida viene firmata utilizzando la firma digitale dell'account a cui fa riferimento la transazione precedente

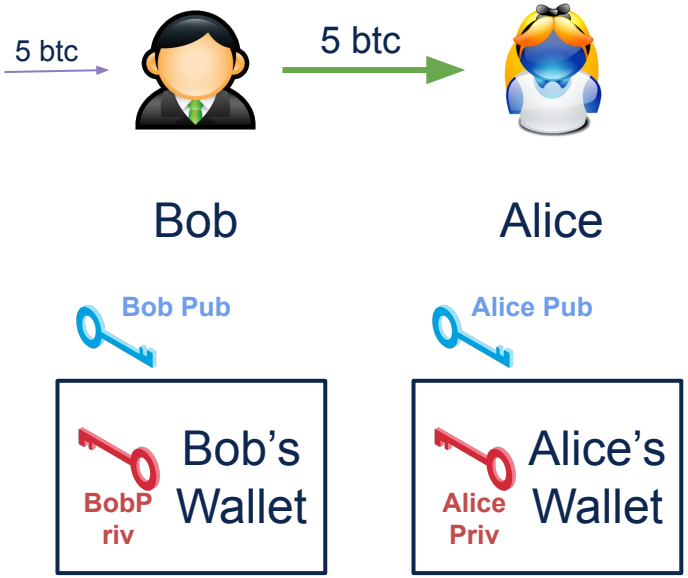


Transazioni: Esempio



TX 1 : 5 btc
TX 2 : 5 btc |--> Bob
TX 3 : Bob |--> Alice

Identità

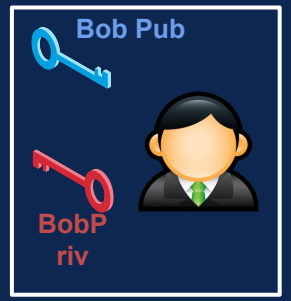

5 btc
5 btc
Bob
Alice
Bob Pub
Alice Pub
BobPriv
Bob's Wallet
Alice Priv
Alice's Wallet

Transazioni e Identità: Esempio

Bob Alice

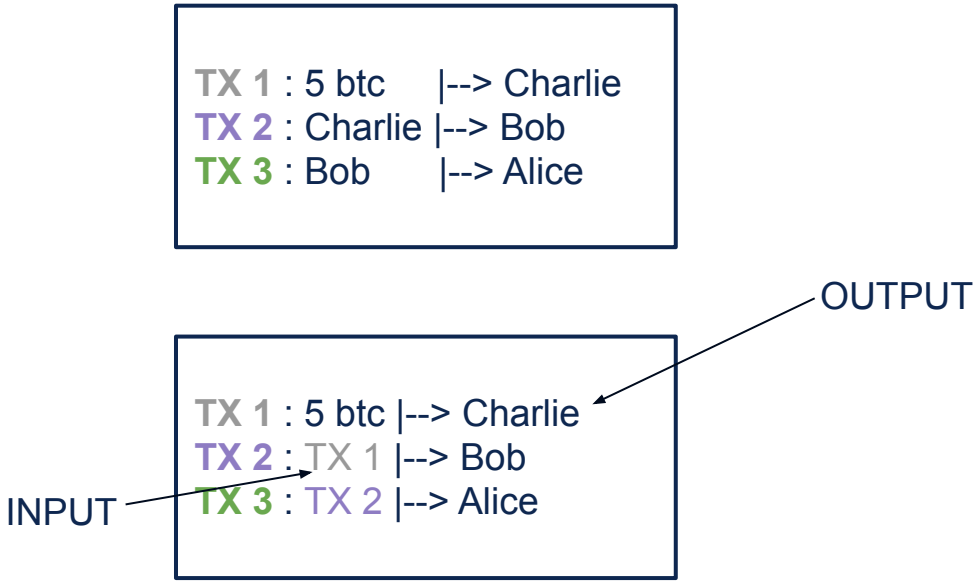
TX 1 : 5 btc
TX 2 : 5 btc |--> Bob Pub
TX 3 : Bob Pub |--> Alice Pub

Alice's Wallet



Bob's Wallet

UTXO: Unspent Transaction (TX) Output



Ogni transazione nella blockchain contiene almeno un OUTPUT

Gli output vengono poi spesi dagli INPUT di transazioni successive

Transazioni

Gli INPUT devono essere sbloccati con una firma digitale

TX 1 : 5 btc |--> Charlie Pub
TX 2 : TX 1 |--> Bob Pub
TX 3 : TX 2 |--> Alice Pub

\+ TX 4 : TX 3 |--> Dana Pub

sign(TX 4, 

)

TX 1 : 5 btc |--> Charlie Pub
TX 2 : TX 1 |--> Bob Pub
TX 3 : TX 2 |--> Alice Pub
TX 4 : TX 3 |--> Dana Pub

Alice's Wallet

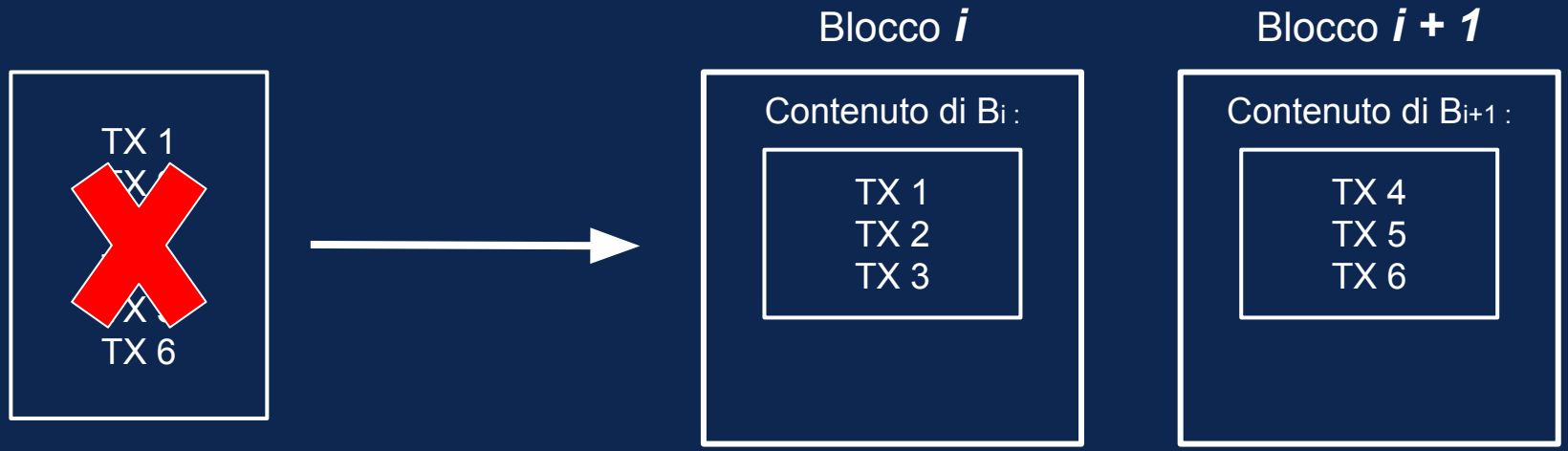
Blockchain

- Cosa scrivere sul registro⟶**transazioni**

- Struttura del registro⟶**chain of blocks**

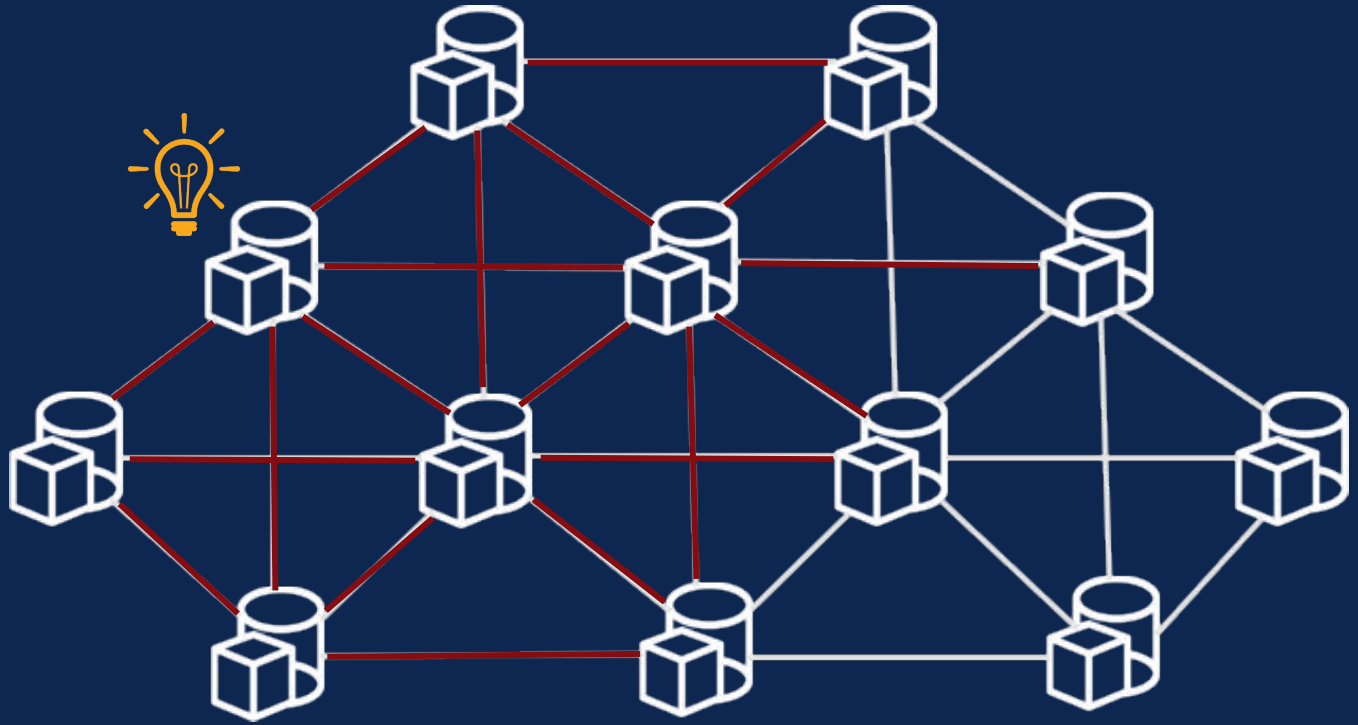
Registro a Blocchi

invece di avere un unico documento contenente tutto il registro di transazioni, la blockchain lo divide in BLOCCHI:

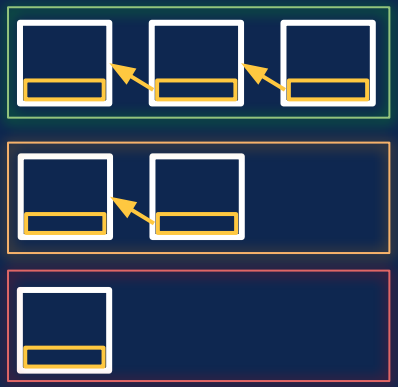


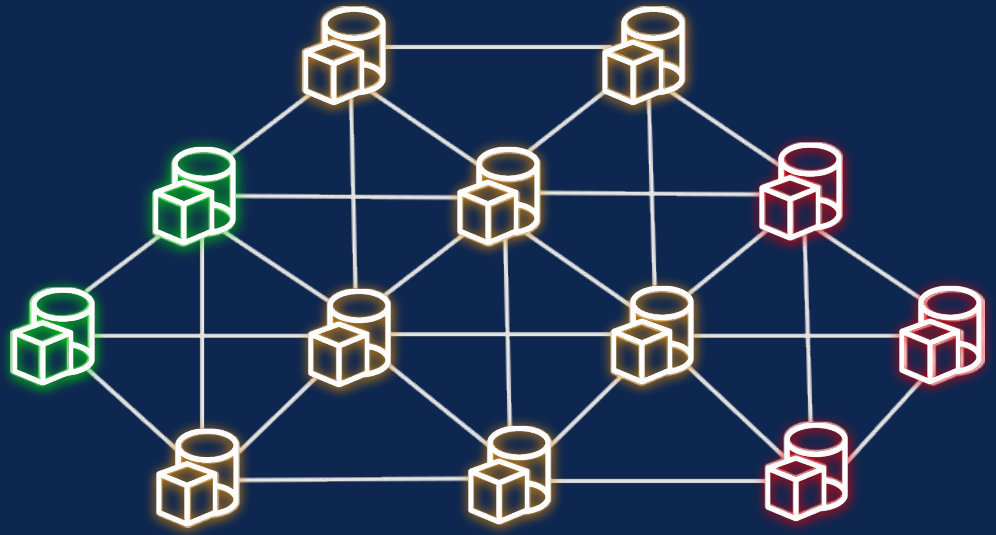
Registro distribuito: creazione di un nuovo blocco

Registro distribuito: propagazione di un blocco



Registro distribuito: sincronizzazione

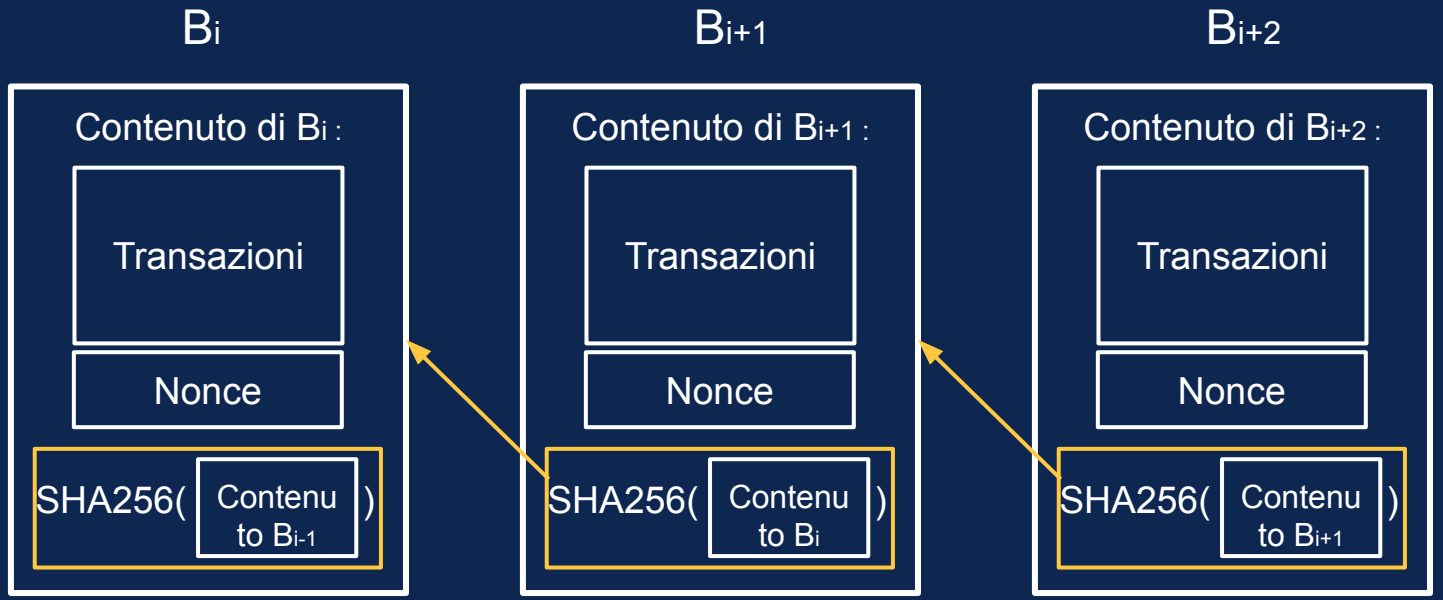




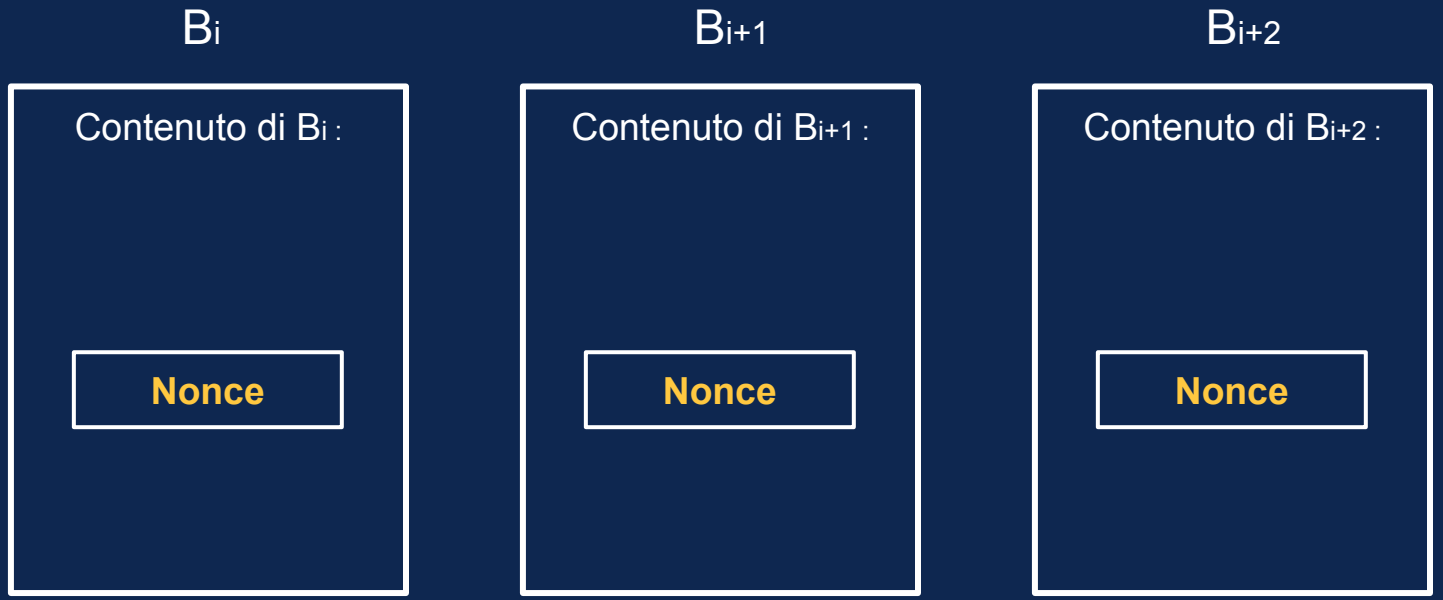
Come mantenere l’ordine cronologico?

ovvero: come formare la catena?

Struttura Blocchi



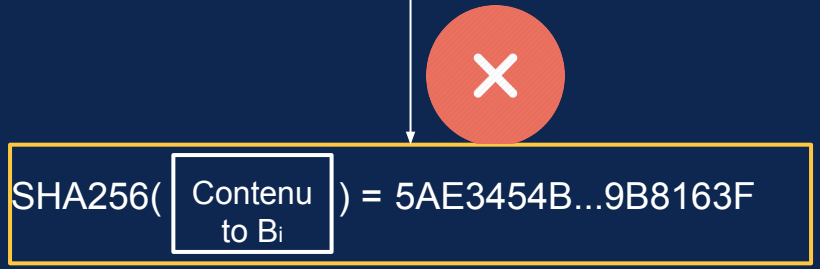
Struttura Blocchi



Mining

Risolvere un puzzle crittografico, es. trovare “l'ago in un pagliaio”

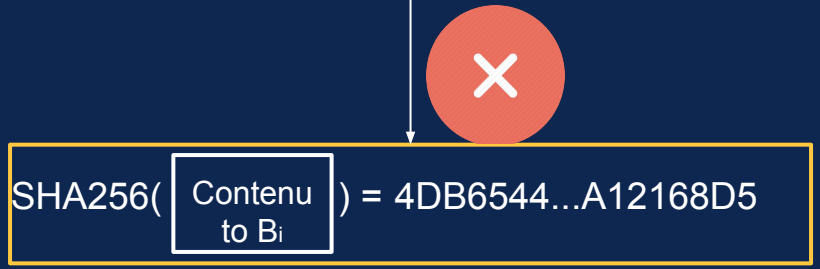
Puzzle:
questo digest deve iniziare con
9 zeri



Mining

Risolvere un puzzle crittografico, es. trovare “l'ago in un pagliaio”

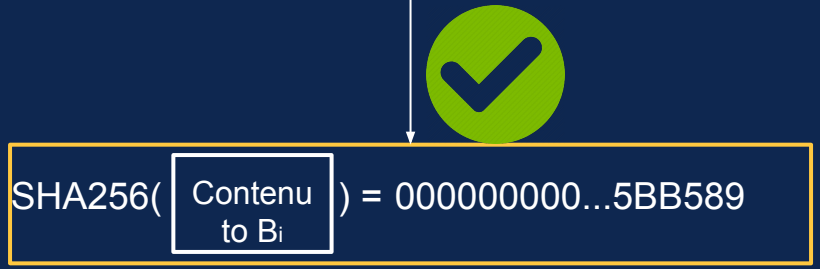
Puzzle:
questo digest deve iniziare con
9 zeri



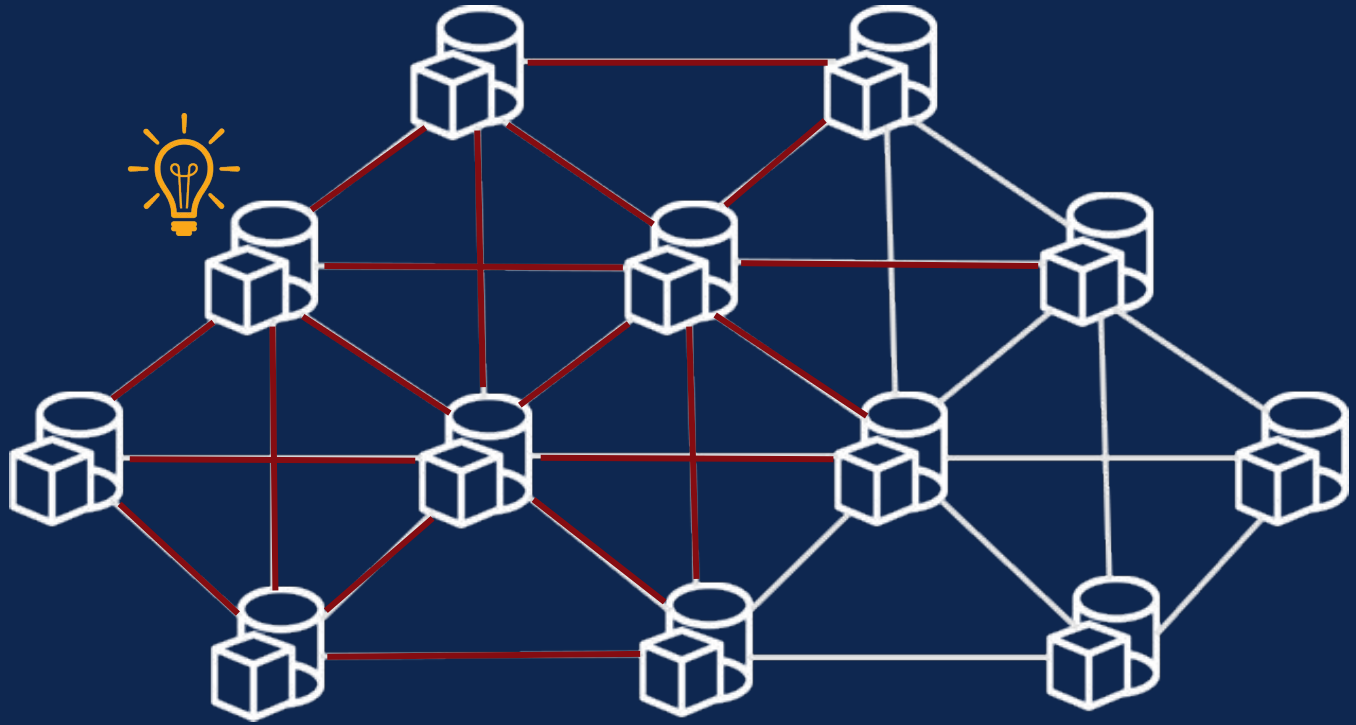
Mining

Risolvere un puzzle crittografico, es. trovare “l'ago in un pagliaio”

Puzzle:
questo digest deve iniziare con
9 zeri

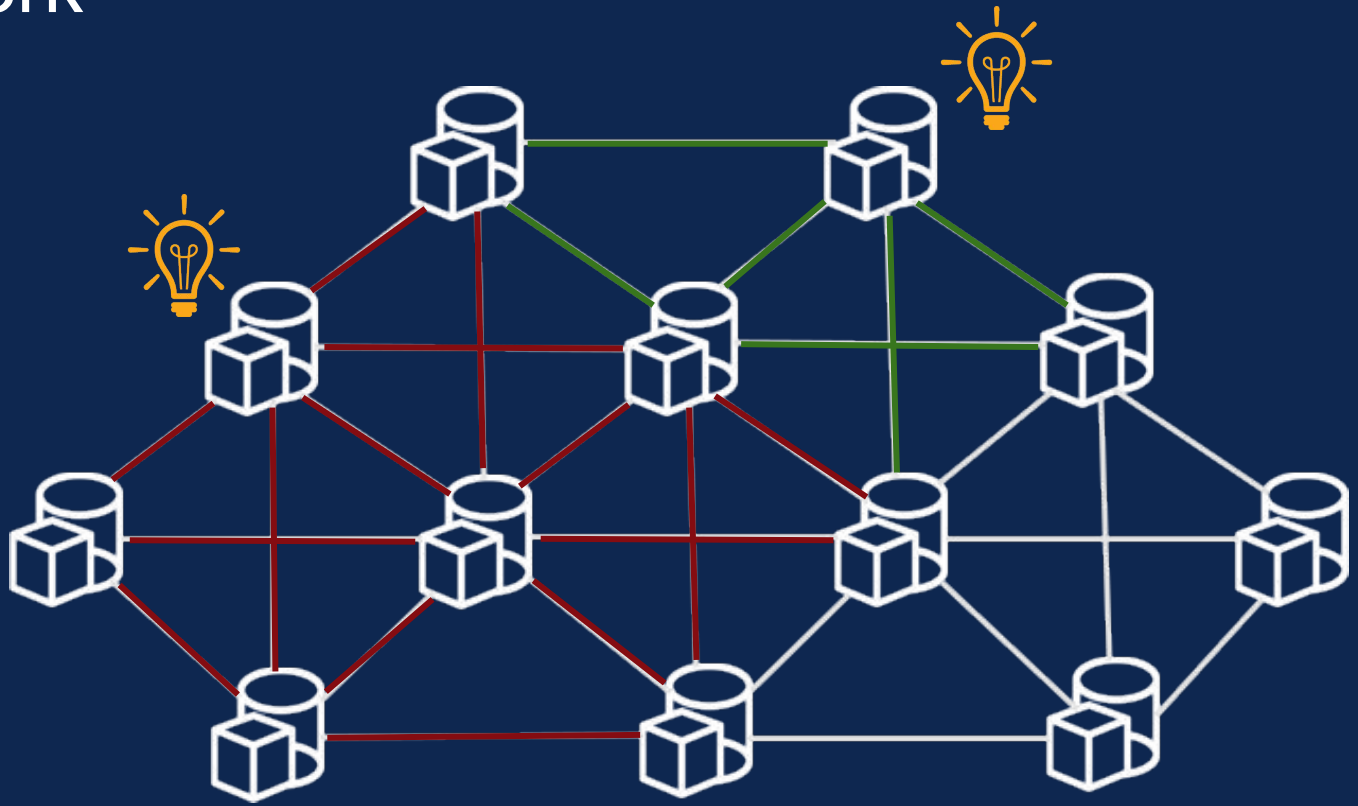


Proof of Work

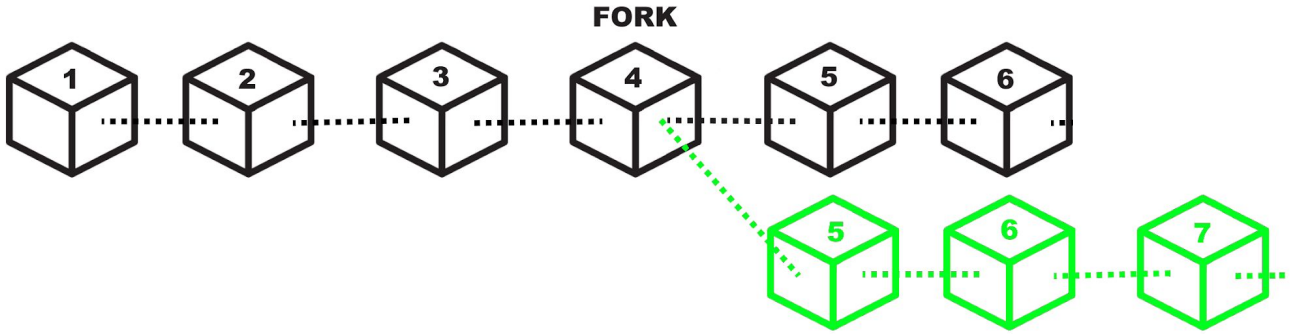


1. Nodo risolve PoW
2. Trasmette il blocco ai vicini
3. Tutti i nodi controllano che:
- Il PoW è valido
- Il contenuto del blocco è valido
4. Quindi lo trasmettono ai loro vicini

Fork

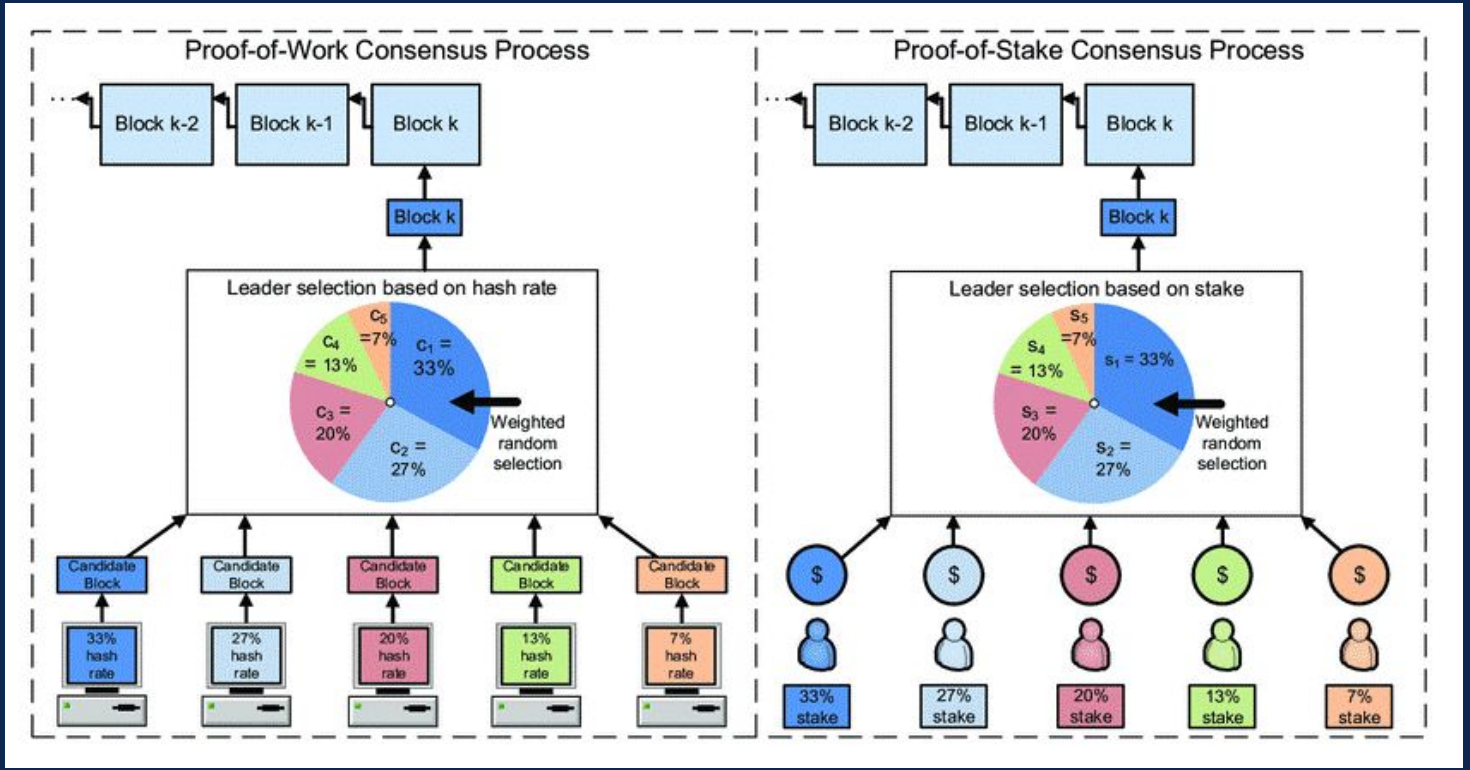


Fork



Il **fork accidentale** si verifica quando:
- due o più minatori trovano un blocco quasi nello stesso momento
Si risolve quando vengono aggiunti uno o più blocchi successivi e una delle cate
diventa più lunga →
La rete abbandona i blocchi che non sono nella catena più lunga

PoW -> Proof of Stake



Demo

https://andersbrownworth.com/blockchain/blockchain