

# Funzioni Hash Crittografiche: confronto tra Poseidon e SHA-256

Corso di Laurea in Informatica - Macroarea di Scienze Matematiche,  
Fisiche e Naturali



Laureando: Mihai Alexandru Sandu

Relatore: Francesco Pasquale

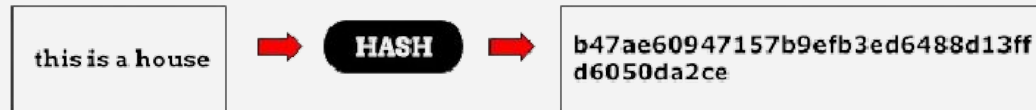
Anno Accademico: 2024-2025

---

# Le Funzioni Hash

Le funzioni hash sono fondamentali in crittografia perché permettono di trasformare input di lunghezza arbitraria in output a lunghezza fissa.

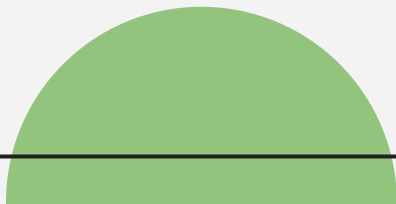
$$H: \{0, 1\}^* \rightarrow \{0, 1\}^n$$



Vengono utilizzate per garantire l'integrità dei dati, per le firme digitali, nelle blockchain, per la memorizzare le password, ...

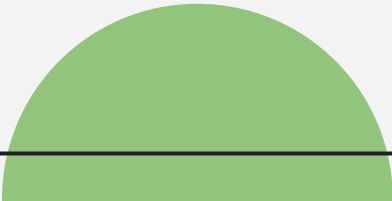
# Proprietà fondamentali

- **One-wayness:** Dato un valore  $y$  nell'immagine della funzione, è computazionalmente difficile trovare un  $x$  tale che  $H(x) = y$ .
- **Second preimage resistance:** Dato un input  $x$  e il suo digest  $H(x)$ , è difficile trovare un  $x' \neq x$  tale che  $H(x) = H(x')$ .
- **Collision resistance:** È difficile trovare due input distinti  $x, x'$  tali che  $H(x) = H(x')$ .



# Funzioni Hash Note

Nel tempo sono state ideate diverse funzioni hash:

- MD5: Produce un digest 128bit, basato sulla costruzione Merkle-Damgård, ad oggi deprecato.
  - SHA-256: Funzione appartenente alla famiglia SHA-2 (standard NIST), produce un digest di 256bit, anch'esso basato sulla struttura Merkle-Damgård.
  - RIPEMD-160: Alternativa europea a SHA-1, produce un digest di 160bit e viene utilizzata nella blockchain di Bitcoin.
  - Poseidon: Algoritmo progettato nel 2019 ed ottimizzato per circuiti a conoscenza zero ZKP.
- 

# Poseidon

Poseidon è una funzione Hash progettata per essere efficiente nelle *Zero-Knowledge Proofs* (ZKP).

Queste sono tecniche crittografiche che permettono ad un **prover** di dimostrare a un **verifier** di conoscere un'informazione segreta, senza rivelarla.

Devono soddisfare tre proprietà:

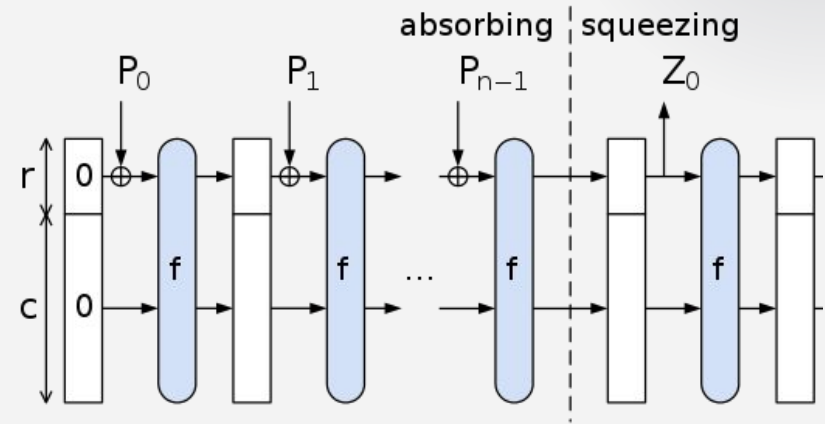
- Completezza: Se l'affermazione è vera, il verifier è convinto.
- Soundness (correttezza): Se l'affermazione è falsa, nessun prover malizioso può convincere il verifier.
- Zero-Knowledge: Il verifier non impara nulla oltre alla validità della prova.

# Design di Poseidon

Utilizza una costruzione Sponge:

- Fase di Absorbing: input diviso in *rate* e *capacity*, di seguito permutato da Poseidon  $\pi$
- Fase di Squeezing: si prende il *rate* della permutazione e la si aggiunge al restante input

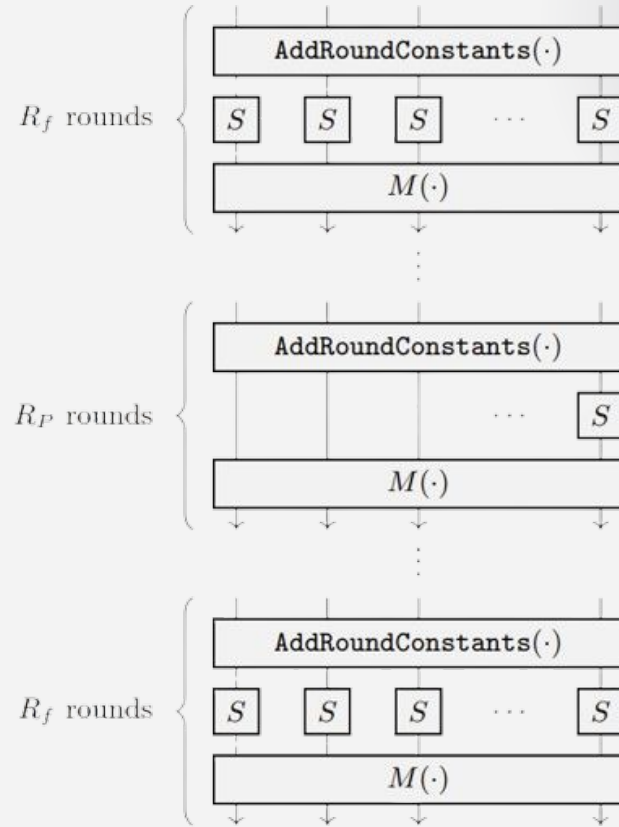
Permutazione composta da Partial Rounds  $R_p$  e Full Rounds  $R_f$



# Poseidon $\pi$

Ogni round è composto da 3 passi principali:

1. Addizione di costanti di round.
2. Applicazione dell'S-box: unica componente non lineare,  $S(x) = x^a$
3. Mix Layer: Utilizzo di matrici MDS (Maximum Distance Separable)



# Attacchi noti per Hash

Crittoanalisi Lineare: Tecnica nata per i cifrari a blocchi ha l'obiettivo di trovare relazioni lineari approssimate tra bit di input e output con probabilità  $\neq 0.5$ .

Crittoanalisi Differenziale: Si basa sull'analisi di come differenze negli input si propagano nell'output. È stata usata per attacchi a funzioni hash come MD5 e SHA-1. Più efficace rispetto a quella lineare.

Altri Attacchi: Test Statistici e Attacchi Algebrici





# Test Statistici

I principali test statistici per valutare la robustezza crittografica delle funzioni hash analizzate sono:

- Avalanche Effect
- Collision Resistance
- Uniformità & Chi-square Test
- Bit Positional Analysis & Shannon Entropy
- Hash Pattern Analysis & Autocorrelazione

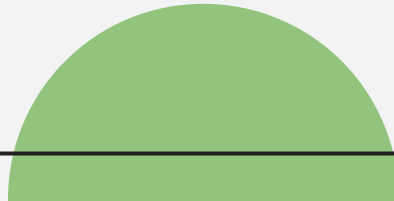


# Implementazione di Poseidon

Il lavoro svolto non ha come obiettivo la valutazione delle prestazioni computazionali bensì testare la robustezza di Poseidon.

L'implementazione è stata effettuata seguendo le specifiche e linee guida del paper originale.

Per eseguire questi test sono stati generati casualmente **1.000.000** di file da 1KB e processati dalle funzioni hash Poseidon, Sha256 e MD5 per confrontare i risultati.



## Avalanche Test

In questa sezione viene analizzato l'Avalanche Effect, una proprietà fondamentale delle funzioni hash crittografiche che garantisce che:

Una minima variazione nell'input (anche un singolo bit) generi un output completamente diverso, con circa il 50% dei bit modificati.

Il test prevede:

1. Generazione di input casuali.
2. Modifica di un singolo bit dell'input.
3. Calcolo e confronto degli hash per misurare la percentuale di cambiamento e la deviazione standard.

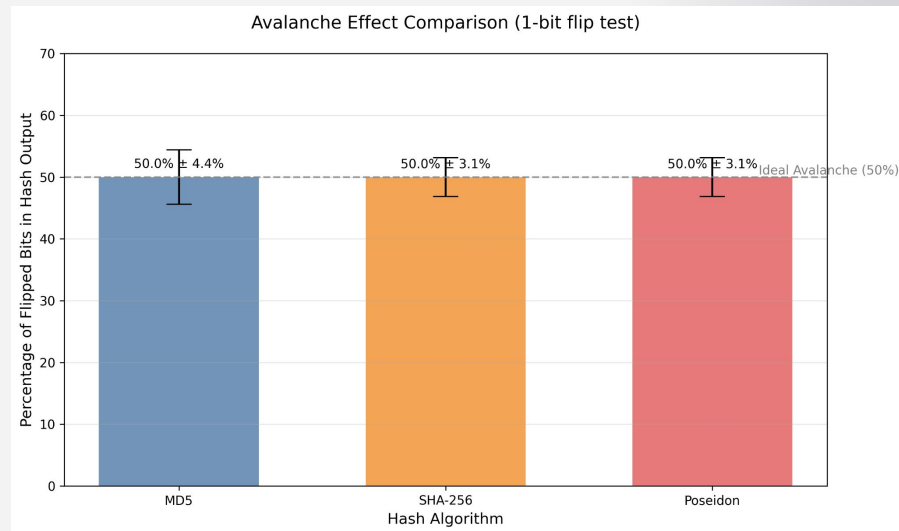
# Risultati Avalanche

I risultati ottenuti per l'Avalanche test sono:

MD5: ~**50.01%** di cambiamento, deviazione standard 4.41%.

SHA-256: ~**50.00%** di cambiamento, deviazione standard 3.13%.

Poseidon: ~**50.01%** di cambiamento, deviazione standard 3.12%.



Tutti gli algoritmi hanno dimostrato un comportamento coerente con l'Avalanche Effect teorico, garantendo alta sensibilità e robustezza crittografica.

## Paradosso del Compleanno

Viene analizzata la resistenza alle collisioni delle funzioni hash, proprietà che garantisce l'impraticabilità di trovare due input distinti con lo stesso output.

Il **Paradosso del Compleanno** è un risultato probabilistico che evidenzia come la probabilità di collisione tra valori generati casualmente cresca rapidamente all'aumentare del numero di campioni.

Per una funzione hash con output di  $n$  bit, la probabilità di trovare almeno una collisione dopo aver calcolato  $k$  hash distinti è:

$$P \approx 1 - e^{-\frac{k^2}{2^{n+1}}}$$

## Collision Resistance

Quando  $k \approx 2^{n/2}$ , la probabilità di collisione supera il 50%.

SHA-256 così come Poseidon, producono output di 256 bit, servono quindi circa  $2^{128}$  input distinti per avere il 50% di probabilità di trovare una collisione (bound computazionalmente irraggiungibile con le risorse attuali).

Il test prevede oltre che a verificare collisioni per l'intero digest anche di verificare il numero di collisioni all'aumentare della dimensione di esso

# Risultati Collision

La verifica delle collisioni sui digest completi (128 bit per MD5, 256 bit per SHA256 e Poseidon) ha prodotto i seguenti collisioni pari a **0**, come previsto dall'elevata dimensione dello spazio di codominio.

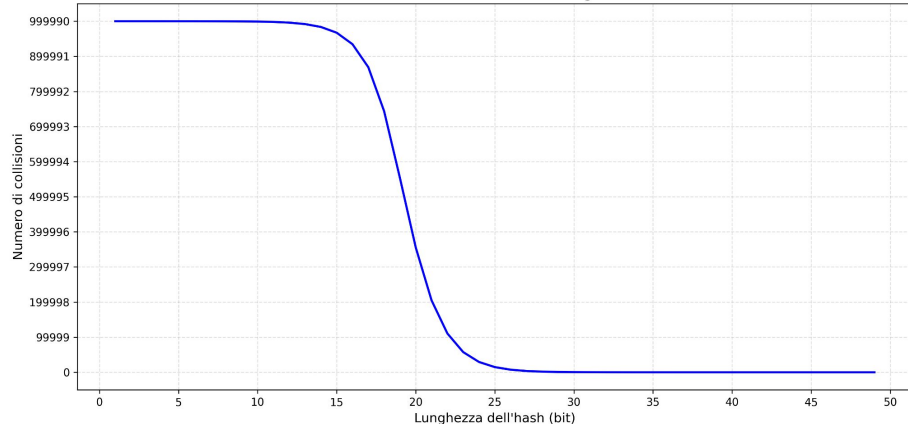
Va ricordato che MD5 è stato storicamente **rotto** proprio attraverso attacchi di collisione pratici, motivo per cui non è stato incluso nella seconda fase dell'analisi relativa al Birthday Paradox.

Bits (n)	Collisioni Poseidon	Collisioni SHA-256
10	998976	998976
15	967232	967232
20	355384	355017
25	14728	14717
30	424	454
35	18	12
40	1	0
45	0	0
50	0	0

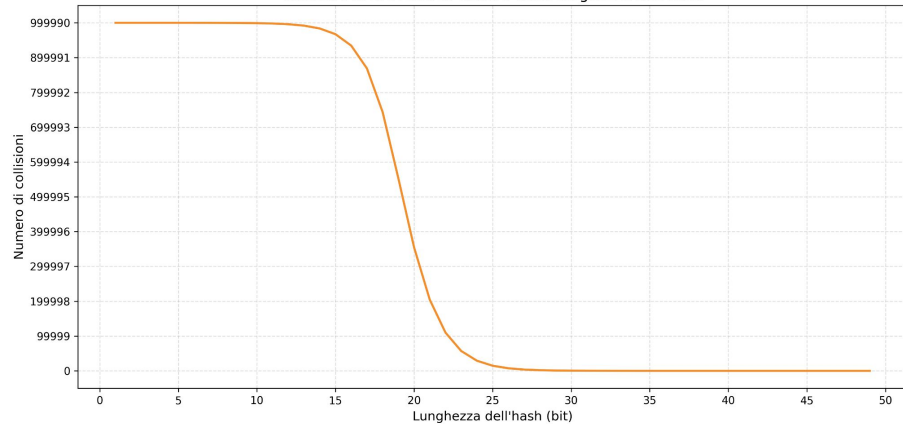
# Risultati Birthday Paradox

Comparazione grafica tra Poseidon e SHA-256

Collisioni di Poseidon al variare della lunghezza dell'hash



Collisioni di SHA-256 al variare della lunghezza dell'hash





# Risultati

Generando 1.000.000 di digest, la soglia critica per collisioni è  $\log_2(1.000.000) \approx 19.93$  bit. Finché  $n$  rimane al di sotto di questo valore, lo spazio degli hash risulta insufficiente.

Per  $n$  superiori a 20 si registra una rapida riduzione delle collisioni, tendente a zero per  $n \geq 43$ .

L'analisi conferma quindi che sia Poseidon che SHA-256 si comportano come funzioni hash ideali, mostrando collisioni in linea con i valori teorici attesi.

## Uniformità & Chi-Square

Viene analizzata l'uniformità dell'output delle funzioni hash, si verifica che i bit e i byte degli output siano distribuiti uniformemente, senza bias statistici.

Si controlla quindi che ogni bit abbia ~50% di probabilità di essere 0 o 1 e di conseguenza ogni byte ha la stessa probabilità di occorrenza.

Per quantificare l'uniformità si utilizza Chi-square Test ( $\chi^2$ ) il quale verifica se le differenze tra valori osservati e attesi sono significative o casuali.

$$\chi^2 = \sum_{i=0}^{255} \frac{(O_i - E)^2}{E}$$

$O_i$  è il numero di occorrenze osservate  
 $E$  è il numero di occorrenze attese

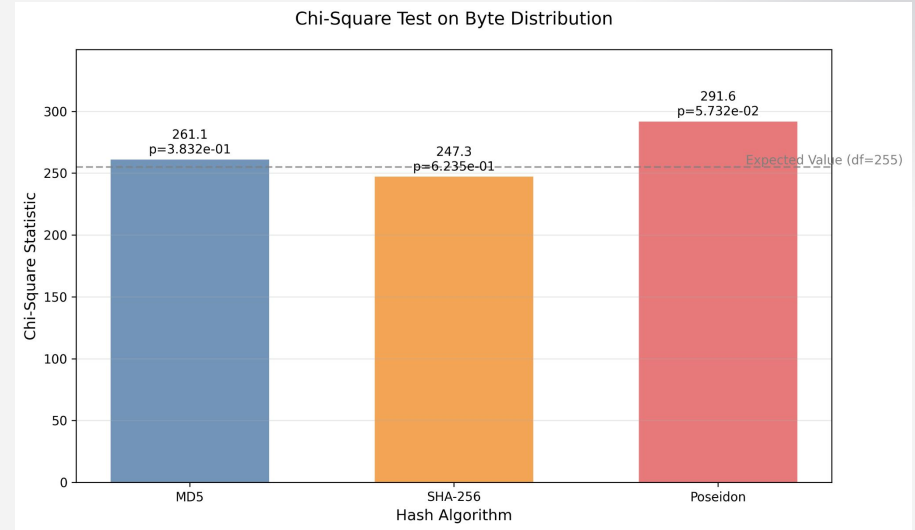
# Risultati Uniformità

Distribuzione dei bit:

- MD5: rapporto **0.9998** tra 0 e 1.
- SHA-256: rapporto **1.0003** tra 0 e 1.
- Poseidon: rapporto **0.9999** tra 0 e 1.

Test Chi-square ( $\chi^2$ ):

- MD5: p-value = **0.3832**
- SHA-256: p-value = **0.6234**
- Poseidon: p-value = **0.0573**



Tutti gli algoritmi mostrano distribuzioni dei bit bilanciate senza bias significativi e hanno superato il Chi-square Test, mostrando una distribuzione statistica coerente con quella uniforme teorica.

## Bit Position & Shannon Entropy

Con questo test si verifica se la distribuzione di 0 e 1 per ogni posizione di bit negli hash si equiprobabile con l'obiettivo di confermare che ogni bit ha ~50% di probabilità di essere 0/1.

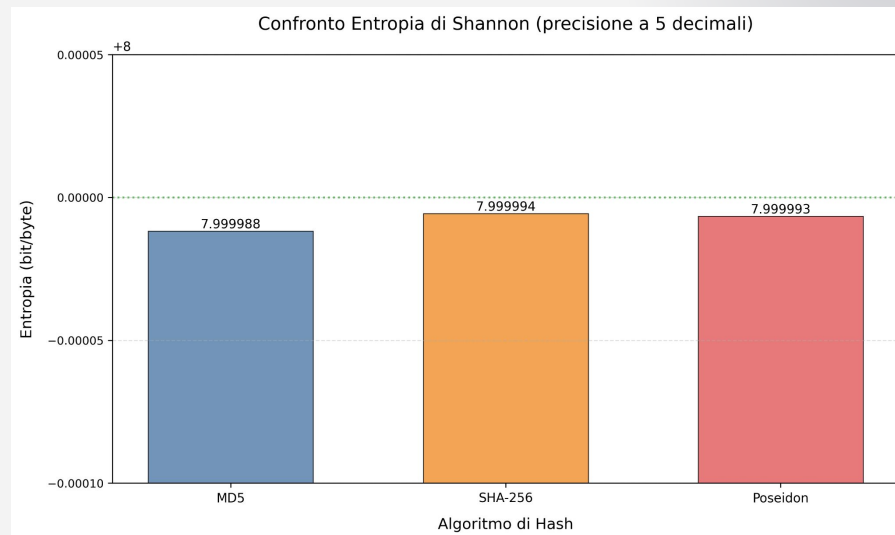
La Shannon Entropy ( $H$ ) invece quantifica l'incertezza media per ogni byte del digest, cioè indicando un'elevata casualità. Data una distribuzione di probabilità  $P(i)$ :

$$H = - \sum_{i=0}^{255} P(i) \cdot \log_2 P(i)$$

# Risultati Position e Entropy

Nessun algoritmo ha mostrato bias significativi **>0.15%** su alcuna posizione di bit. Mentre la Shannon Entropy ha dato i seguenti risultati:

- MD5: **7.99998**
- SHA-256: **7.999994**
- Poseidon: **7.999993**



In base ai bias ottenuti e all'entropia calcolata estremamente vicini al massimo teorico di 8, MD5, SHA-256 e Poseidon garantiscono uniformità e imprevedibilità.

## Hash Patterns & Autocorrelazione

L'analisi di Pattern nell'Hash verifica la presenza di pattern ripetuti all'interno dei vari digest. Si prendono sottostringhe (finestre) di bit all'interno dell'hash e si contano le eventuali collisioni per rilevare ripetizioni significative.

L'Autocorrelazione ( $R$ ), misura la correlazione tra i bit di un hash e sé stesso traslato di un certo lag (ritardo). Si va quindi a verificare che i bit all'interno dell'hash non siano correlati tra loro, ossia che la conoscenza di un bit non fornisca informazioni sui bit vicini o su quelli a distanza fissa. I valori attesi sono:

Lag = 0  $\rightarrow$  autocorrelazione massima 0.5.

Lag > 0  $\rightarrow$  valori vicini a 0.25

$$R(k) = \frac{1}{n-k} \sum_{i=1}^{n-k} b_i \cdot b_{i+k}$$

# Risultati Position e Entropy

Per l'Hash Pattern Analysis il tasso di collisioni per finestre di 32 bit è **praticamente nullo** per tutti gli algoritmi.

Per finestre più piccole (24 e 16 bit) invece si osserva un aumento progressivo delle collisioni, coerente con la riduzione dello spazio possibile.

Per invece l'Autocorrelazione tutti gli algoritmi hanno mostrato valori di autocorrelazione per lag > 0 **vicini a 0.25**, come atteso in sequenze binarie indipendenti.

MD5, SHA-256 e Poseidon mostrano quindi un comportamento coerente con funzioni hash ideali, senza pattern ripetuti o correlazioni sfruttabili, garantendo robustezza crittografica e casualità del digest.

# Conclusioni e Sviluppi Futuri

Poseidon ha dimostrato performance crittografiche **comparabili** a SHA-256 in tutti i test statistici.

Pur essendo un algoritmo recente, Poseidon si è confermato robusto e privo di debolezze strutturali, risultando idoneo non solo per ambienti *ZKP* ma potenzialmente anche per applicazioni più generali.

I risultati ottenuti aprono quindi la strada a futuri approfondimenti, come ad esempio quelli legati ad analisi crittoanalitiche avanzate (differenziale, lineare).





**Grazie per  
l'attenzione!**



# Test Statistici:

01

## **Avalanche**

Verifica cambiamento dell'output modificando singolo bit nell'input.

02

## **Collision**

Verifica difficoltà trovare collisioni tra input distinti con birthday paradox.

03

## **Uniformità**

Verifica distribuzione output equa, usata insieme al test di Chi-Square.

04

## **Bit Positional**

Verifica probabilità bit per posizione; calcola Shannon Entropy.

05

## **Hash Pattern**

Analizza pattern ripetuti e correlazioni interne per robustezza crittografica.