Randomness e Pseudorandomness nella Computazione

SO-ReCoDi
ERC Advanced Grant 834861

Luca Trevisan
Università Commerciale *Luigi Bocconi*

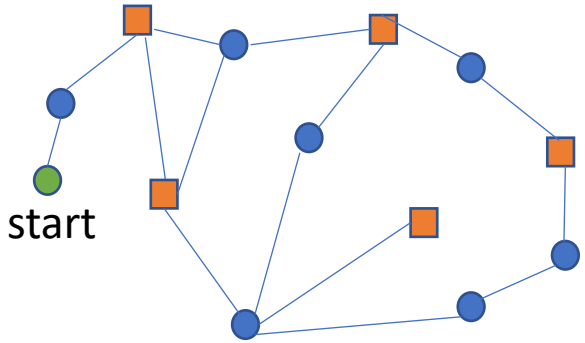
Informatica teorica

- Nuove tecniche per progettare algoritmi
  dimostrare correttezza, analizzare efficienza
- Studiare *limitazioni* degli algoritmi
  complessità computazionale, crittografia
- Applicare modelli algoritmici in altre scienze
  algorithmic game theory, sistemi complessi, neuroscienze, evoluzione
- Applicare techniche matematica dell'informatica alla matematica pura

Randomness nella computazione?

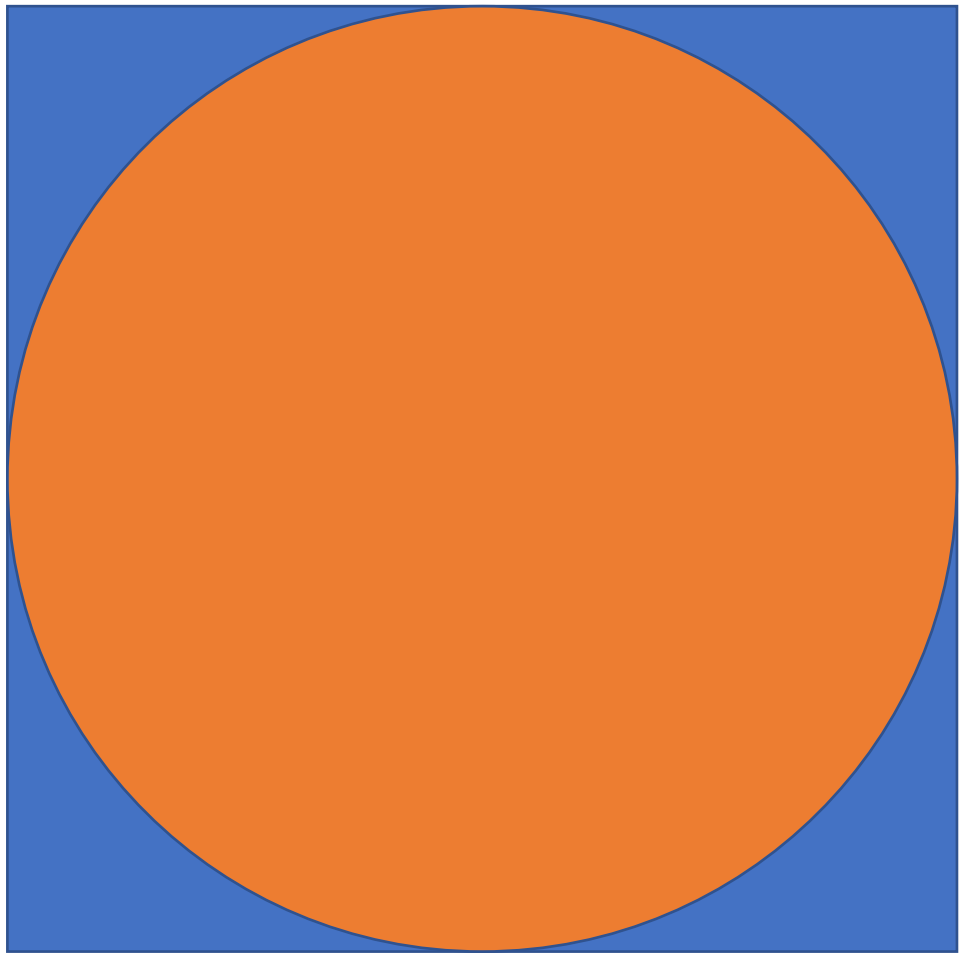
- $\int_0^\infty \frac{\cos x}{x^2-3x}\,dx = ?$

- Trova percorso più breve che passi per tutti i siti quadrati



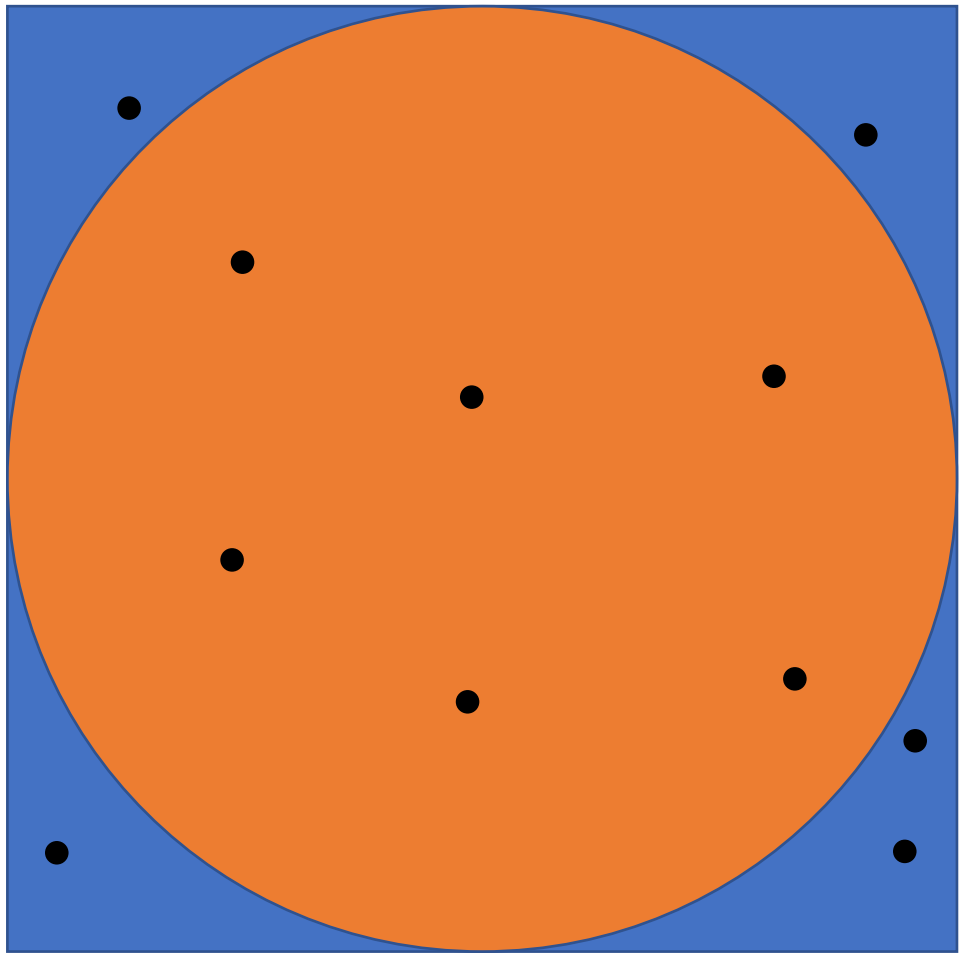
- Trova il mediano nella lista 45,21,16,76,34,97,42,171,32

- 23555091804486281357 è un numero primo?

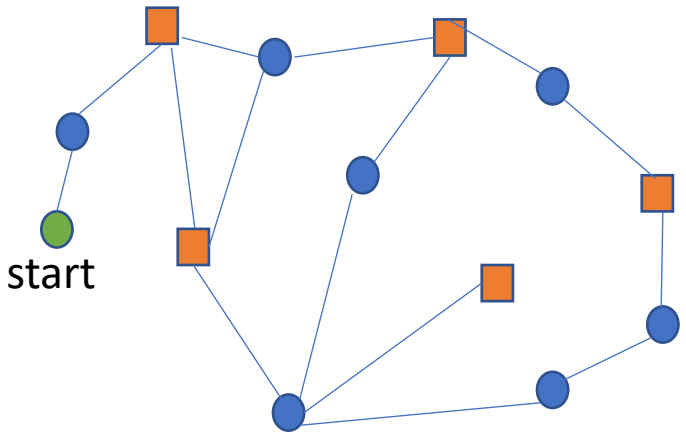


$$\pi = 4 \cdot \frac{\text{area cerchio}}{\text{area quadrato}}$$

$$\pi = 4 \cdot Prob[\text{punto preso a caso appartiene al cerchio}]$$



Ottimizzazione combinatoria



- Semplice approccio per risolvere problemi di ottimizzazione con un grande numero di soluzioni possibili:
  - Cominciare con una soluzione semplice da costruire (potenzialmente molto meno buona dell'ottimo)
  - Fare piccoli cambiamenti che migliorino la soluzione, finché ciò non sia più possibile

Algoritmo di Metropolis-Hastings

- Metropolis-Hastings:
  - Cominciare con una soluzione semplice da costruire (potenzialmente molto meno buona dell'ottimo)
  - Ripetutamente:
    - Fare un piccolo cambiamento con probabilità proporzionale a

$$e^{\beta \cdot (\text{miglioramento della soluzione})}$$

[Metropolis 1950s][Hastings 1970]

235550918044862813 57 è un numero primo?

- Provare tutti i possibili divisori di 2355509180448628135 7

235550918044862813357 è un numero primo?

- ~~Provare tutti i possibili divisori~~

- Se n è primo, allora ("piccolo teorema" di Fermat):

$$\text{per ogni } x, x^{n-1} = 1 \; mod \; n$$

23555091804486281357 è un numero primo?

- ~~Provare tutti i possibili divisori~~

- Se 23555091804486281357 è primo, allora per ogni x,

$$x^{23555091804486281356} = 1 \; mod \; 23555091804486281357$$

- ma, per esempio, per $x = 5$:

$$5^{23555091804486281356} = 497460017214450105 6 \; mod \; 23555091804486281357$$

Miller-Rabin


[Miller 1976]
[Rabin 1980]

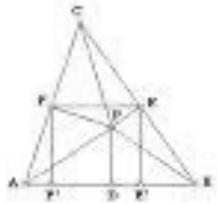
- Dato n
- Scegliere x a caso:
  - Se $x^{n-1} \neq 1 \; mod \; n$ allora output: *non primo*
  - Se $x^{n-1} = 1 \; mod \; n$ allora output: *forse primo*

(Con qualche aggiustamento) per ogni numero, risposta giusta con alta probabilità

- Nessun algoritmo *deterministico* con simile efficienza fino al 2002

Metodo probabilistico in combinatoria

$$\lim_{n\to\infty} \frac{M(n,k,l)}{\binom{n}{l}/\binom{k}{l}} = 1$$



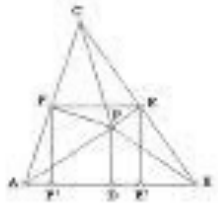
- Erdös (anni 50) voleva dimostrare l'esistenza di grafi (reti) con una certa proprietà (basso numero di Ramsey)

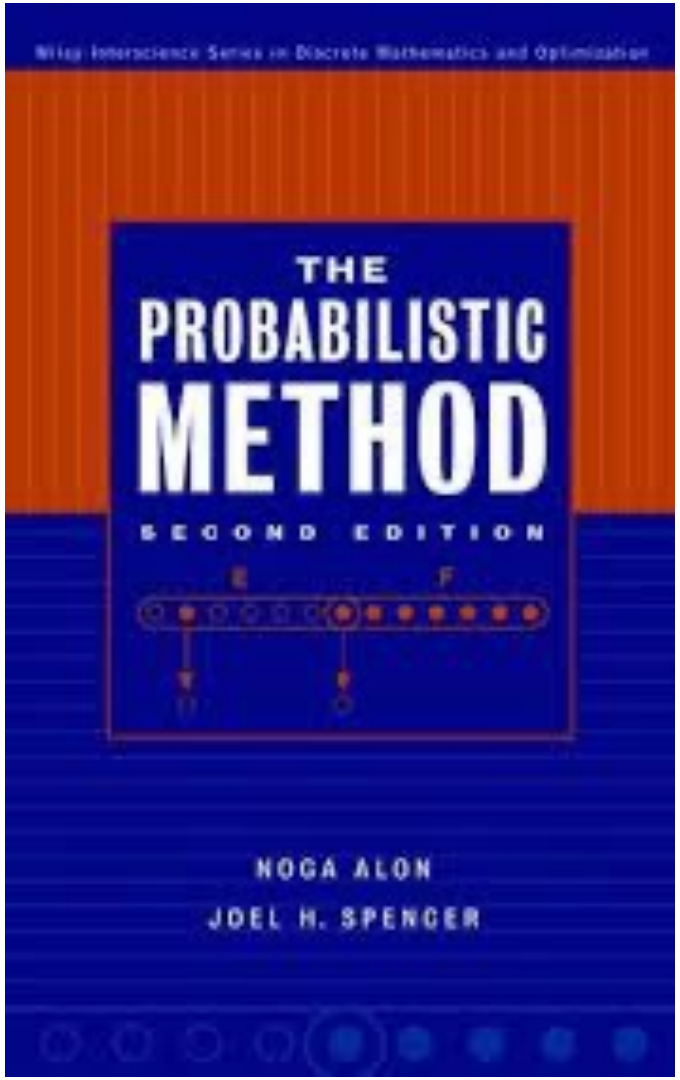
- Dimostrazione di Erdös:
  - Immaginiamo di scegliere una rete a caso
  - Calcoliamo la probabilità che *non* abbia la proprietà desirata
  - Tale probabilità è < 1
  - **Devono esistere reti con proprietà desiderata**

- Ancora non si sa come costruire queste reti deterministicamente
  (ma progressi importanti negli ultimi dieci anni)

Metodo probabilistico in combinatoria

$$\lim_{n\to\infty} \frac{M(n,k,l)}{\binom{n}{l}/\binom{k}{l}} = 1$$





Randomness negli algoritmi

- L’uso di scelte casuali negli algoritmi può facilitare la costruzione di soluzioni con proprietà desiderate

- Se un problema computazionale ammette un algoritmo efficiente randomizzato, esiste necessariamente un algoritmo deterministico con efficienza simile?

- O c’è un gap tra il potere computazionale di algoritmi randomizzati rispetto agli algoritmi deterministici?

Crittografia

Mittente

Messaggio

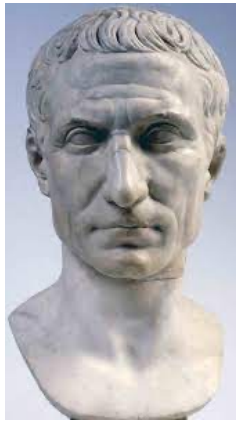
Canale di comunicazione non sicuro

Intercettatore

Destinatario

Crittografia

Mittente



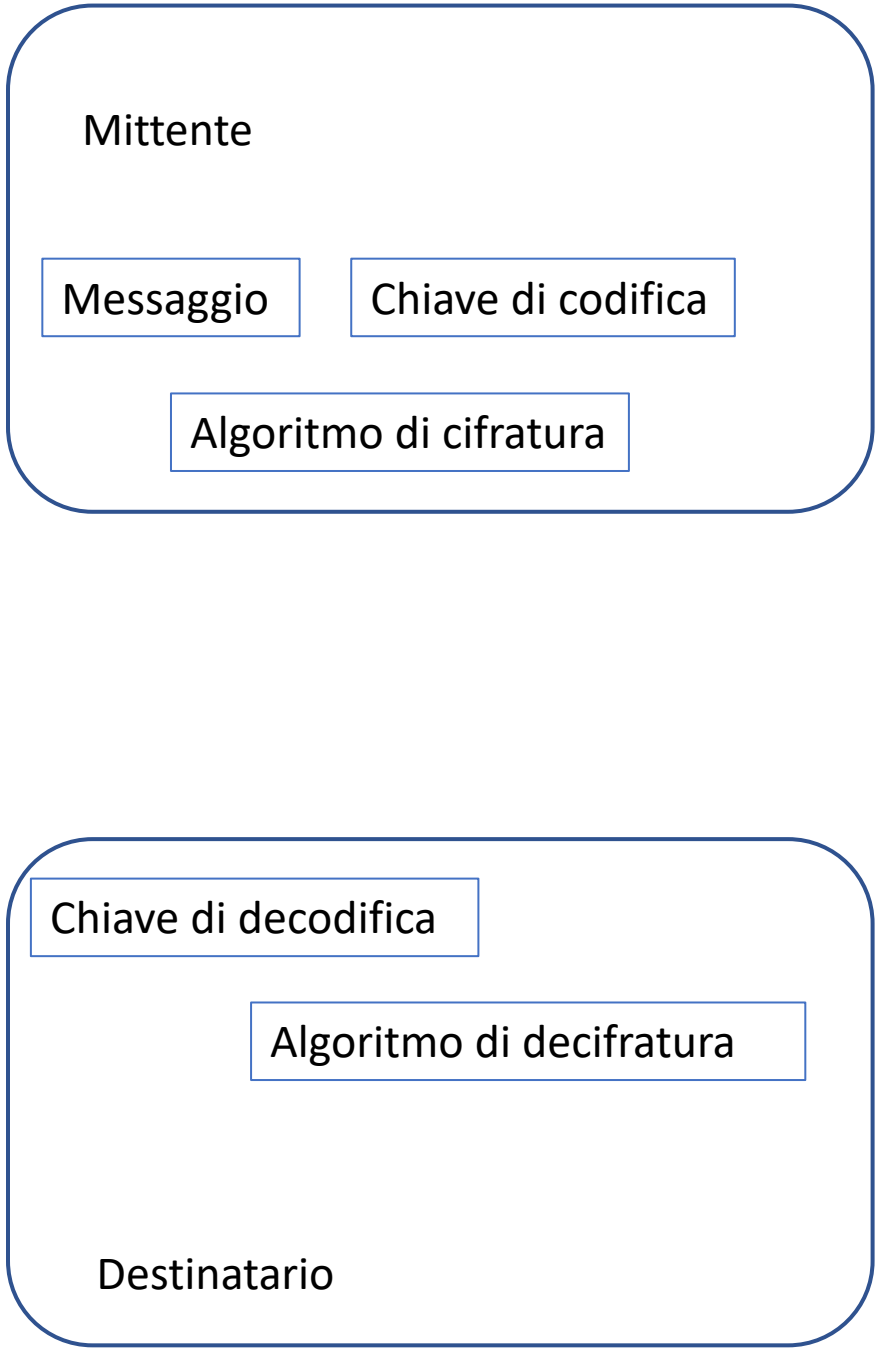
Messaggio

Canale di comunicazione non sicuro

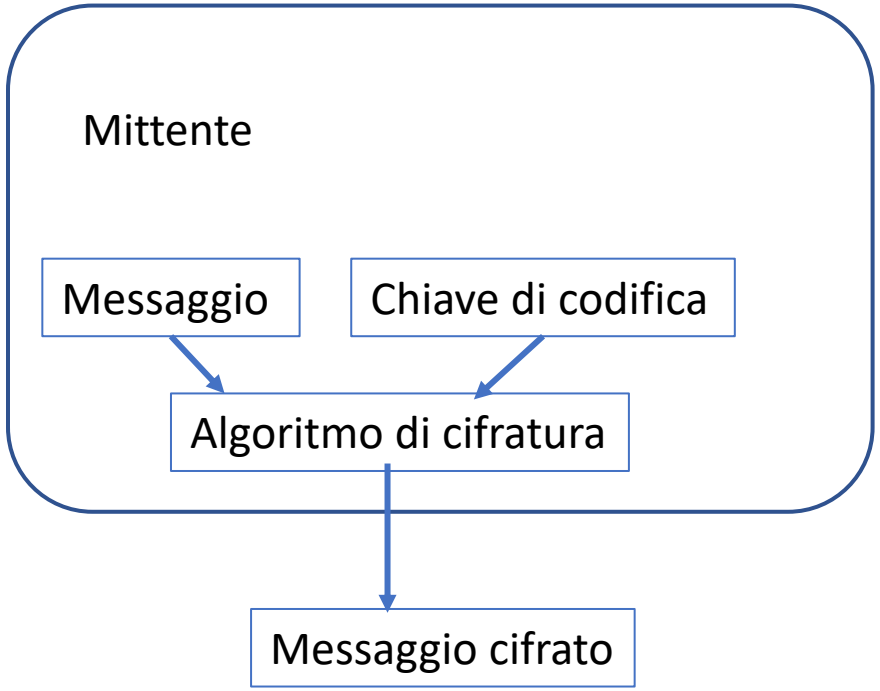
Intercettatore

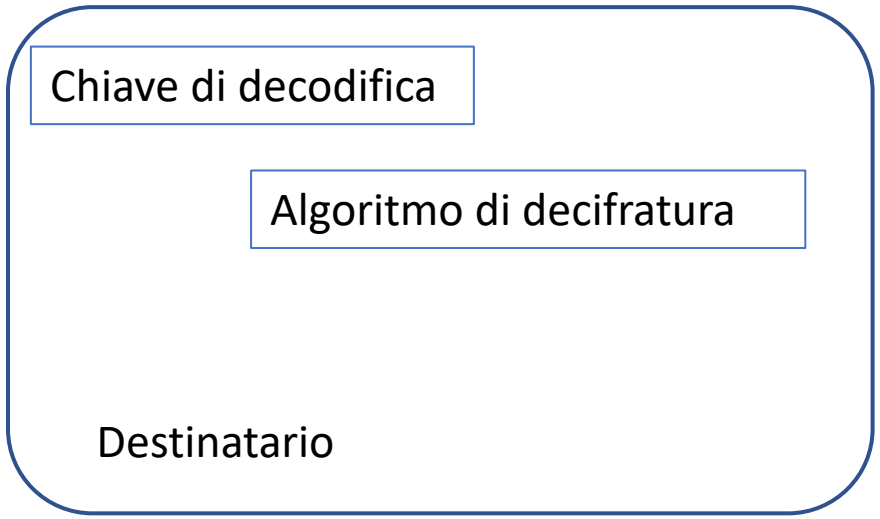
Destinatario

Crittografia

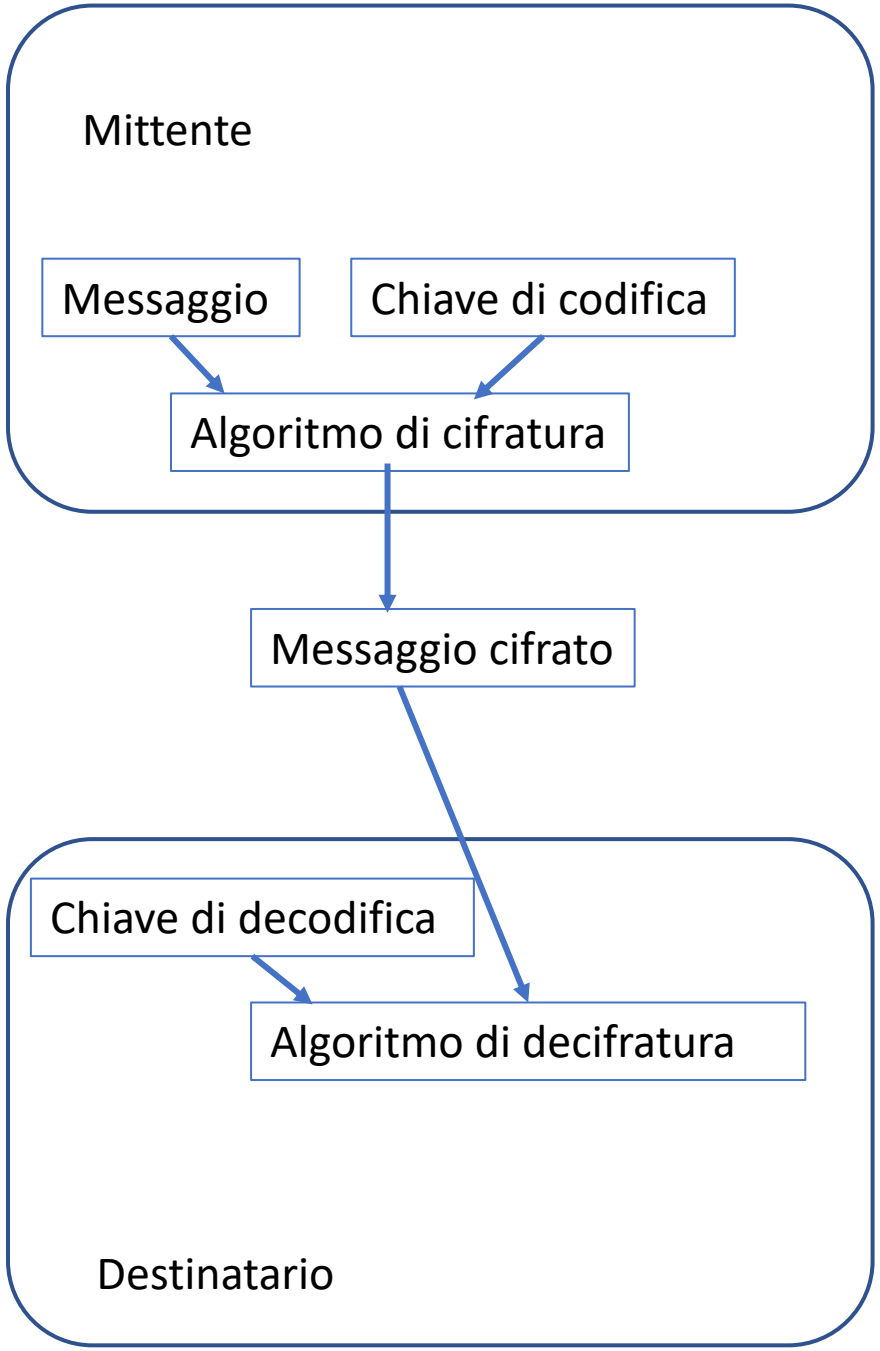


Crittografia

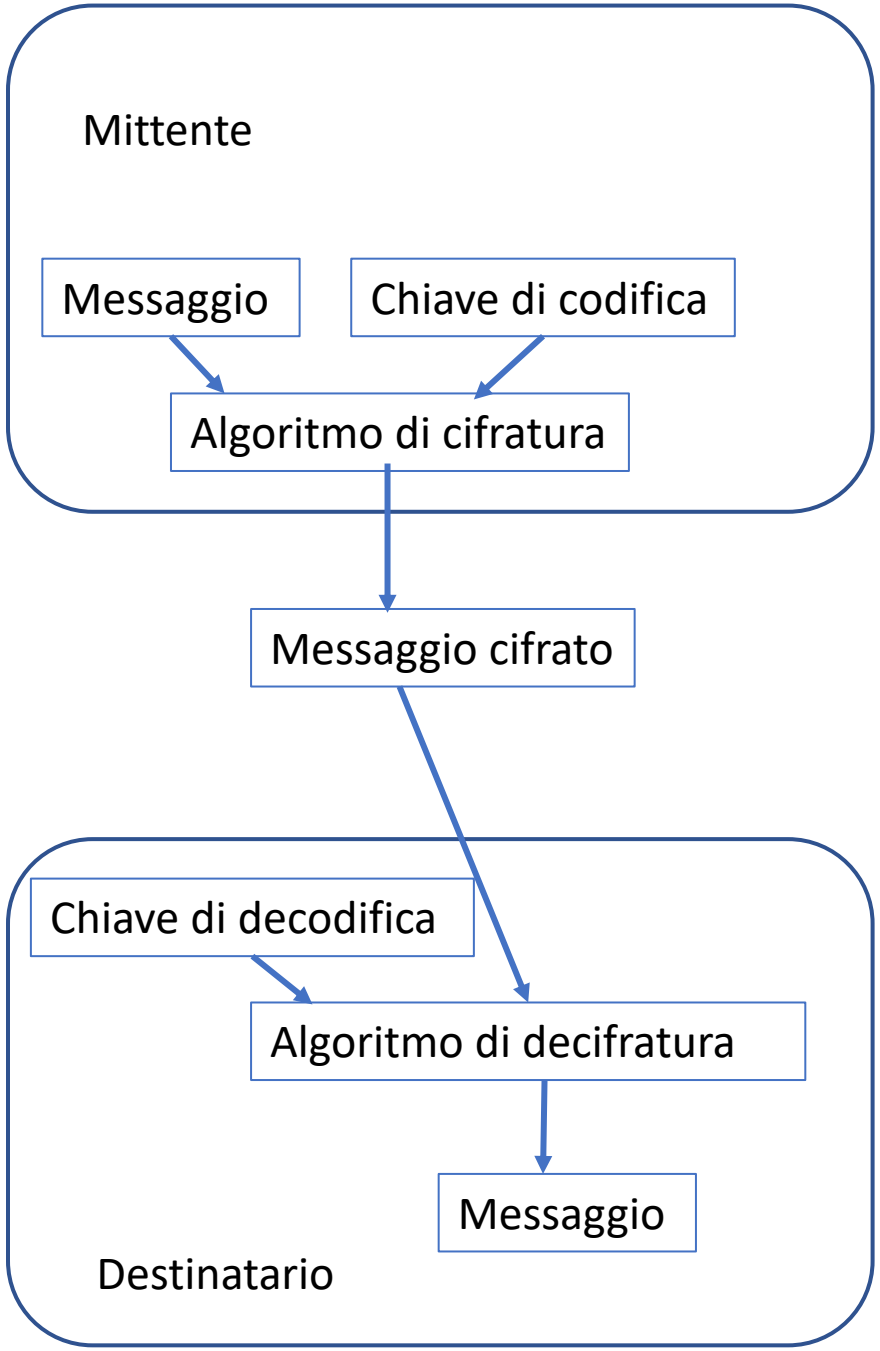




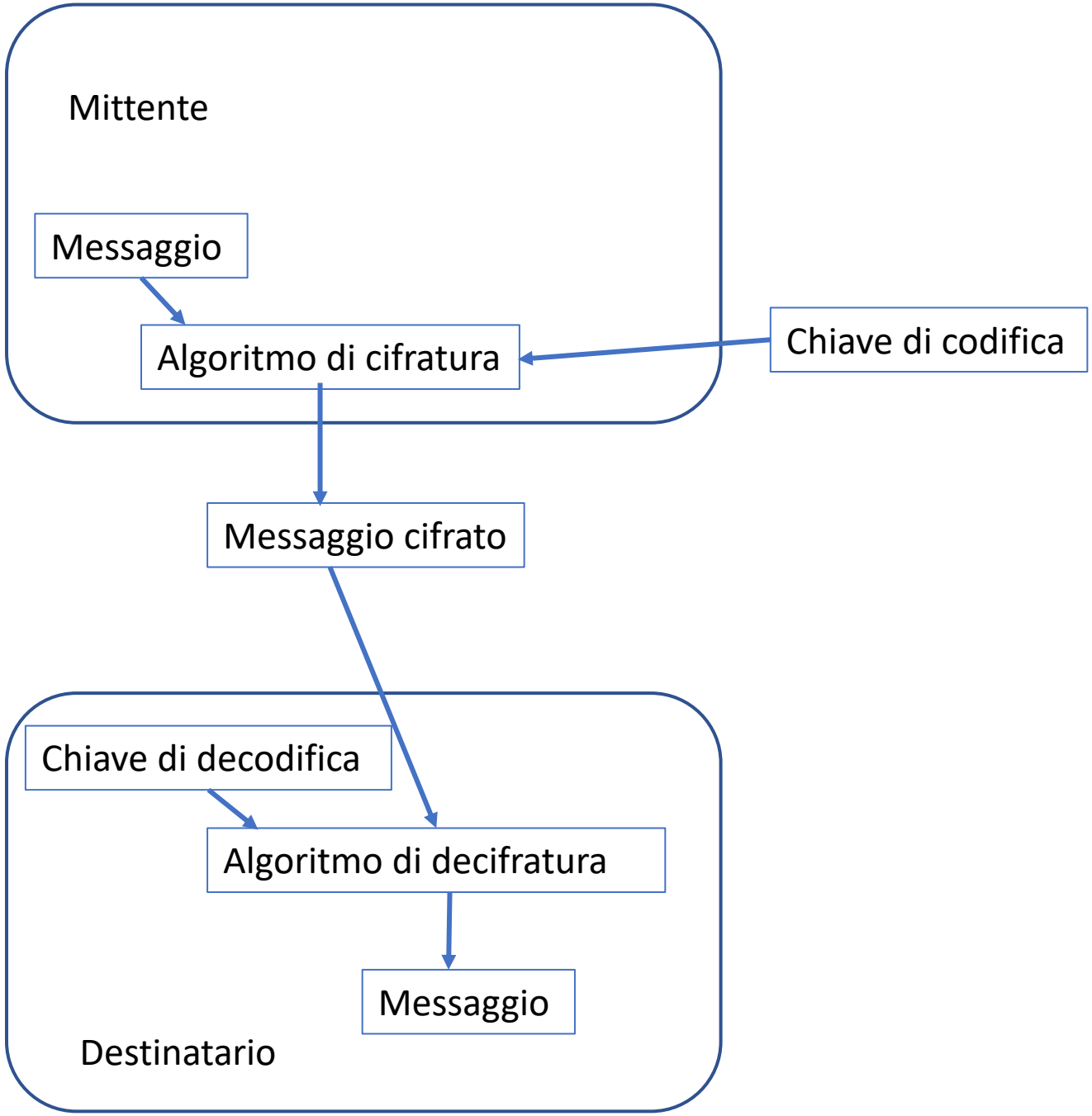
Crittografia



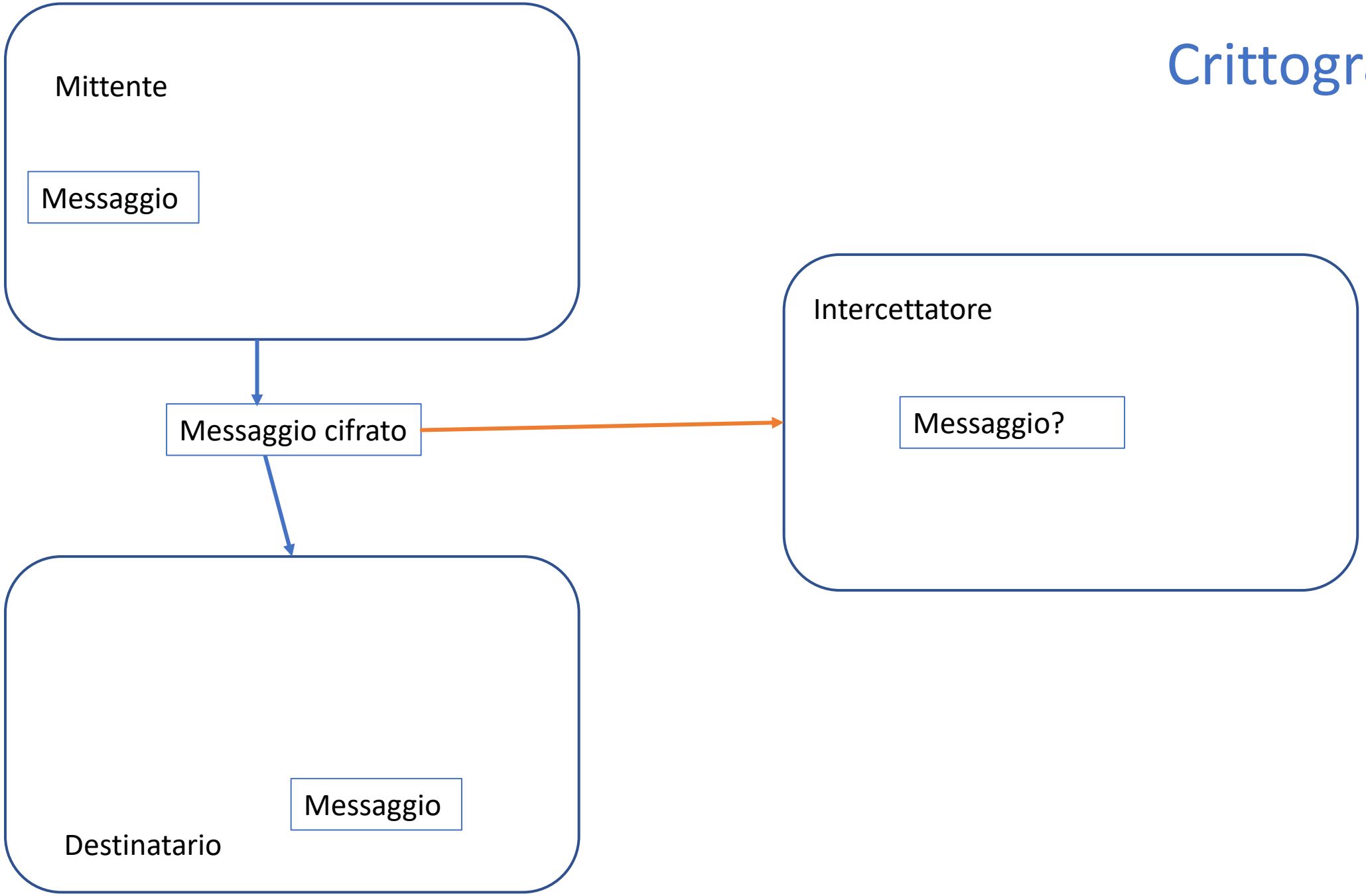
Crittografia



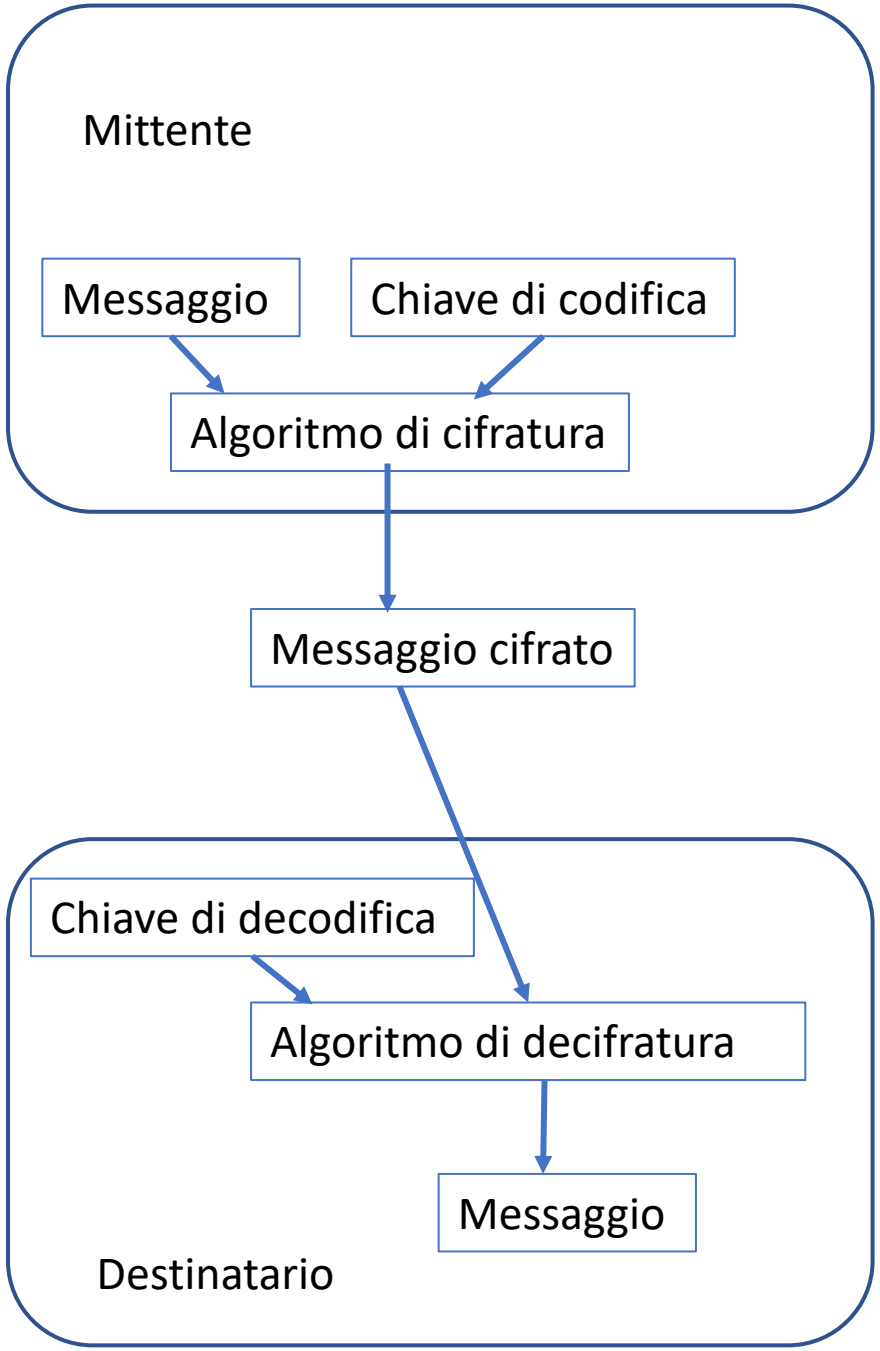
Crittografia



Crittografia

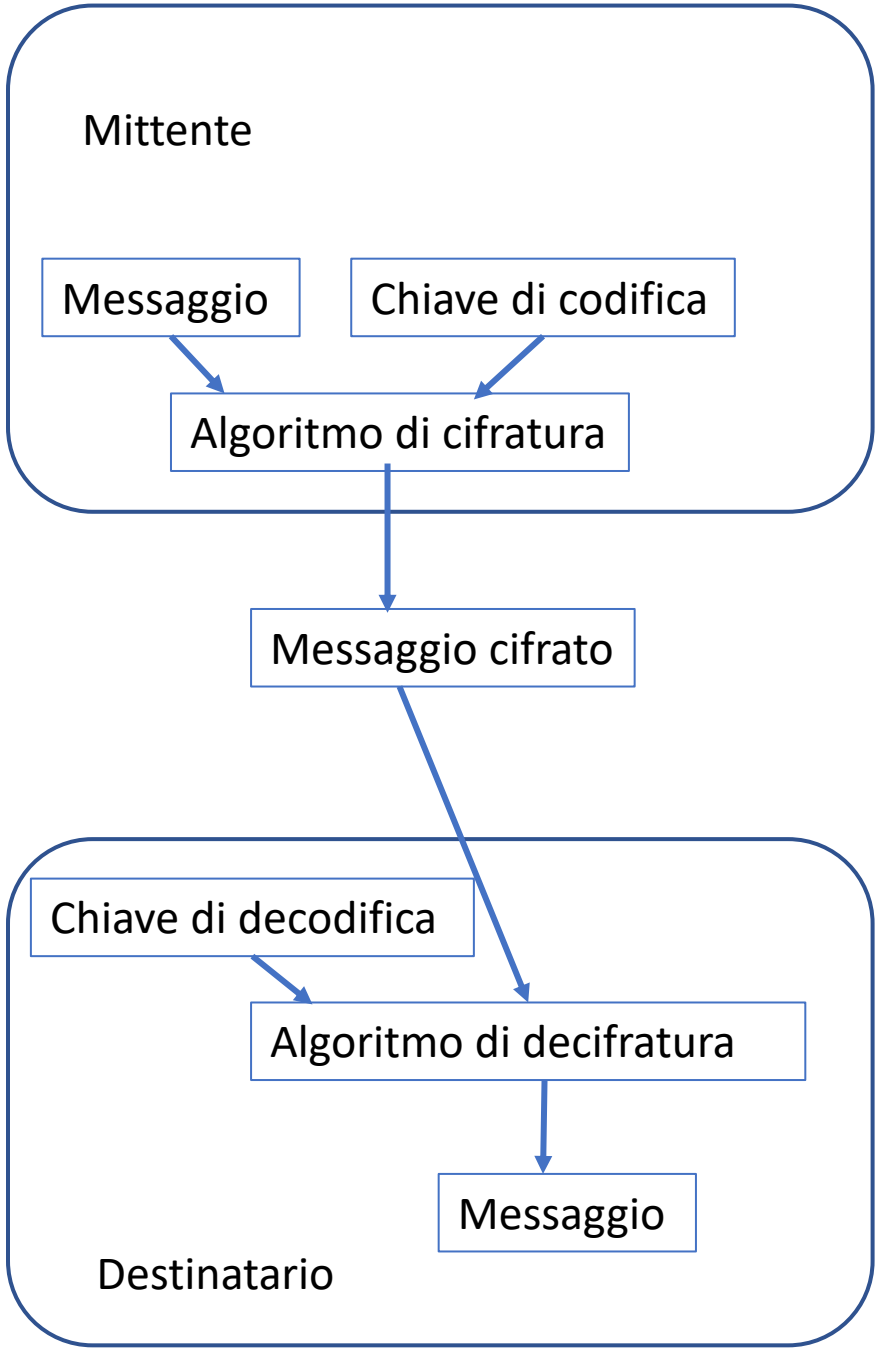


Crittografia



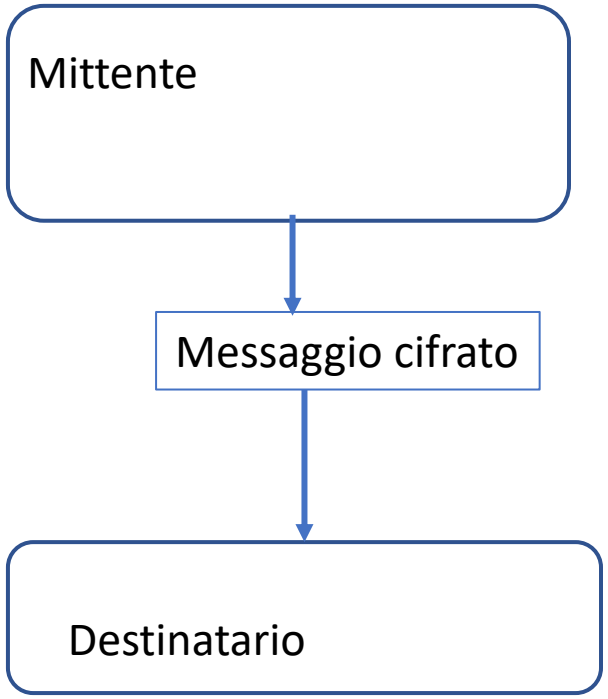
Importante che chiavi di codifica/decodifica siano scelte in modo casuale e impredicibile

Crittografia



Importante che chiavi di codifica/decodifica siano scelte in modo casuale e impredicibile

Crittografia

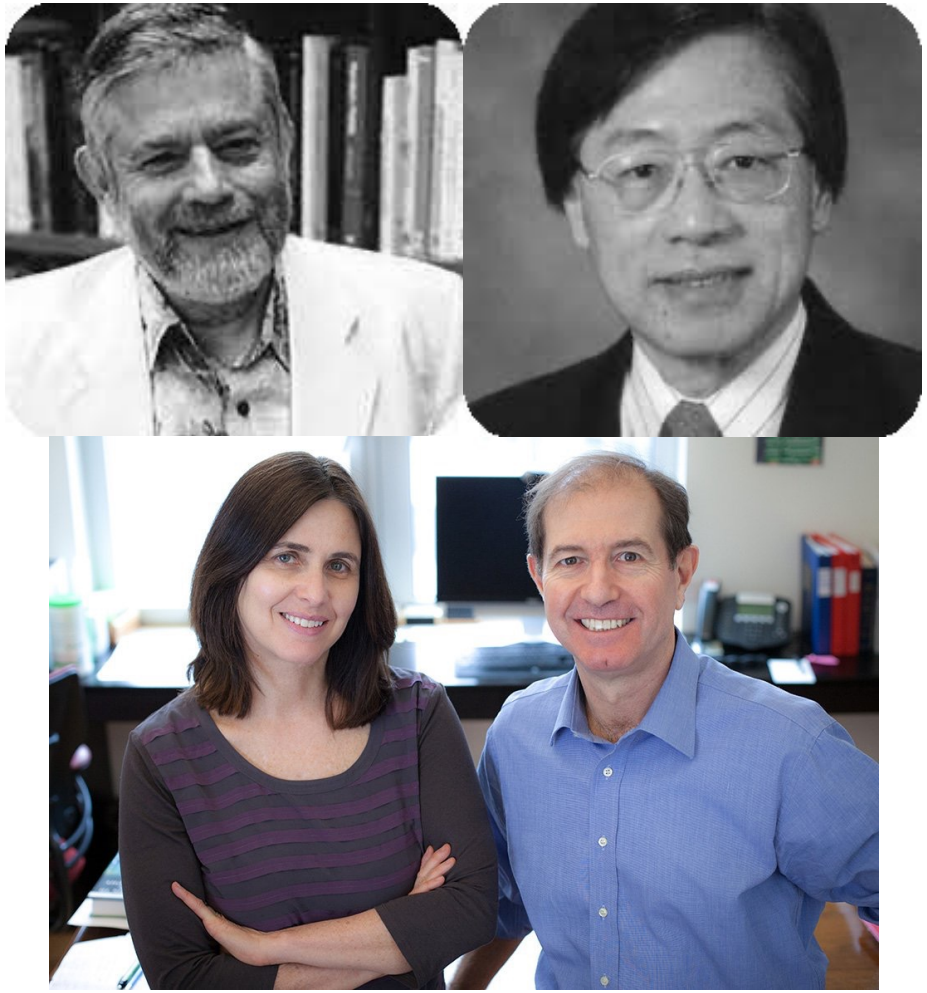


Se il messaggio cifrato viene intercettato, *non deve dare alcuna informazione* sul contenuto del messaggio originale

Intuizioni di Goldwasser e Micali:

1. la cifratura deve essere una operazione probabilistica

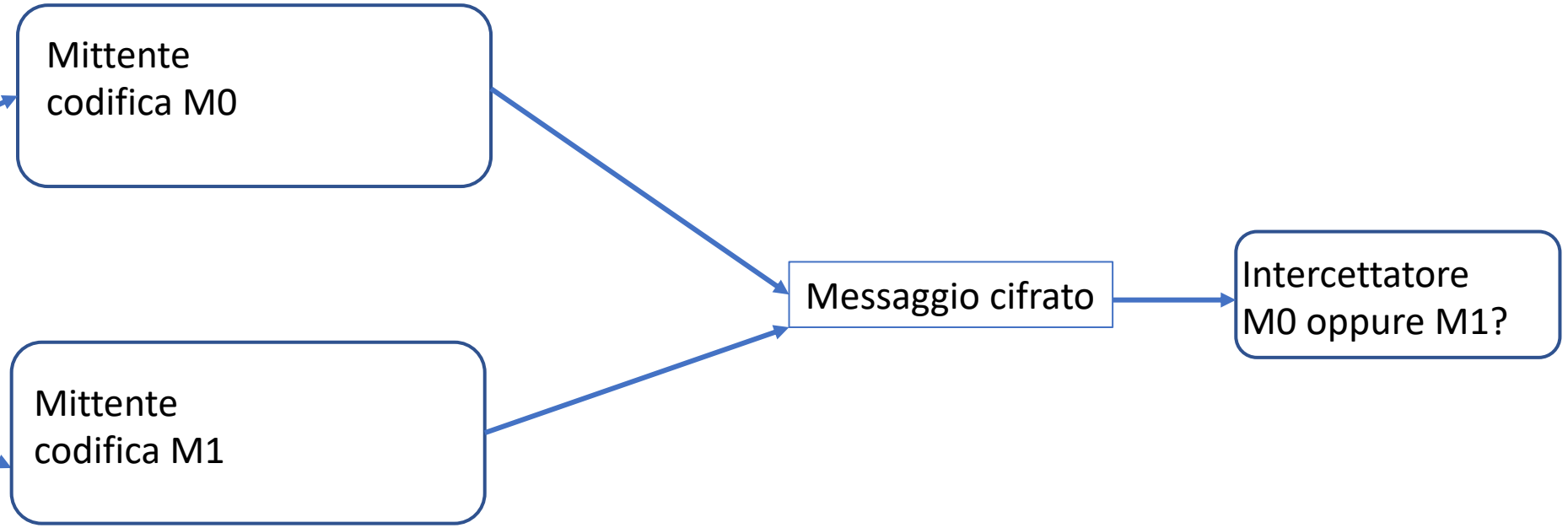
2. È possibile dare una definizione rigorosa di *non dare informazione*, ottenibile in pratica



[Blum, Micali 1982]
[Goldwasser, Micali 1982]
[Yao 1982]

Indistinguibilità

Il metodo di cifratura ha *sicurezza semantica* se, per ogni coppia di messaggi M0 e M1, le loro cifrature sono *indistinguibili*



[Goldwasser, Micali 1982]

Definizione di indistinguibilità:
Se l'intercettatore può eseguire meno di 10^{40} istruzioni, allora riesce a indovinare se il messaggio cifrato dato proviene da M0 oppure M1 con probabilità $< \frac{1}{2} + 10^{-40}$

Randomness in crittografia

- In crittografia, le chiavi segrete devono essere scelte in modo casuale e altamente impredicibile
  - Molti attacchi di crittoanalisi si basano su difetti di progettazione che creano chiavi segrete con troppa poca entropia

- Anche una volta selezionate chiavi segrete, protocolli crittografici devono usare randomness fresca in ogni esecuzione per garantire alti livelli di sicurezza

- Come si genera randomness impredicibile?

Due domande

- Esiste un gap di potenza di calcolo tra algoritmi randomizzati e algoritmi deterministici?

- Come si genera randomness impredicibile per applicazioni crittografiche?

1. Esiste un gap di potenza di calcolo tra algoritmi randomizzati e algoritmi deterministici?

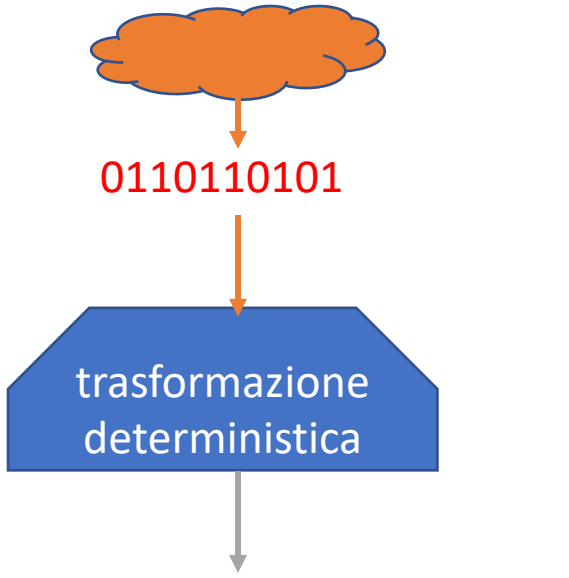
2. Come si genera randomness impredicibile per applicazioni crittografiche?

(1) no, assumendo delle congetture standard [Impagliazzo-Wigderson 1997]

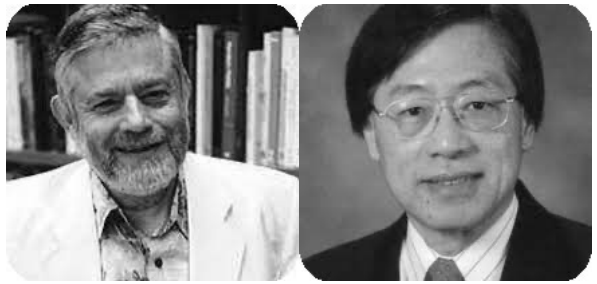
(2) molte tecniche

Un aspetto della mia ricerca: c'è una connesione tra (1) e (2)

Generatori pseudorandom



[Blum, Micali 1982]
[Yao 1982]



Ricevono un input completamente casuale

Producono un output, molto più grande, “pseudocasuale”

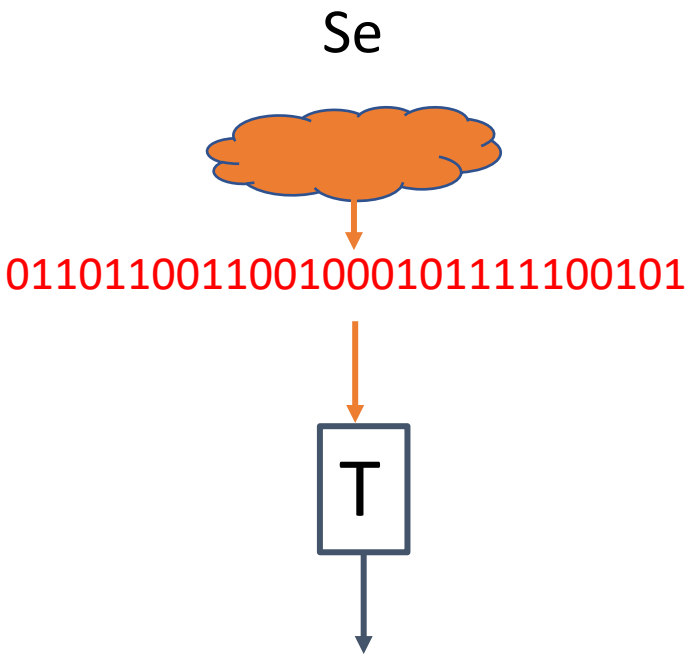
Che vuol dire “pseudocasuale”?

Generatori pseudorandom

Per ogni test statistico T
efficientemente calcolabile

Generatori pseudorandom

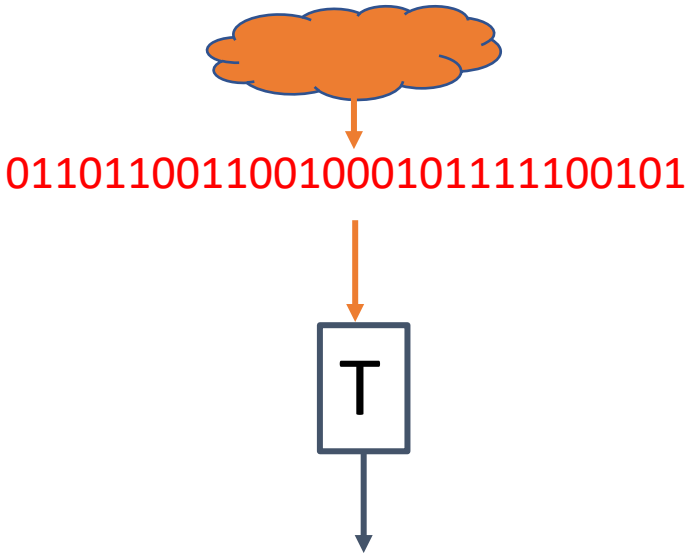
Per ogni test statistico T
efficientemente calcolabile



Generatori pseudorandom

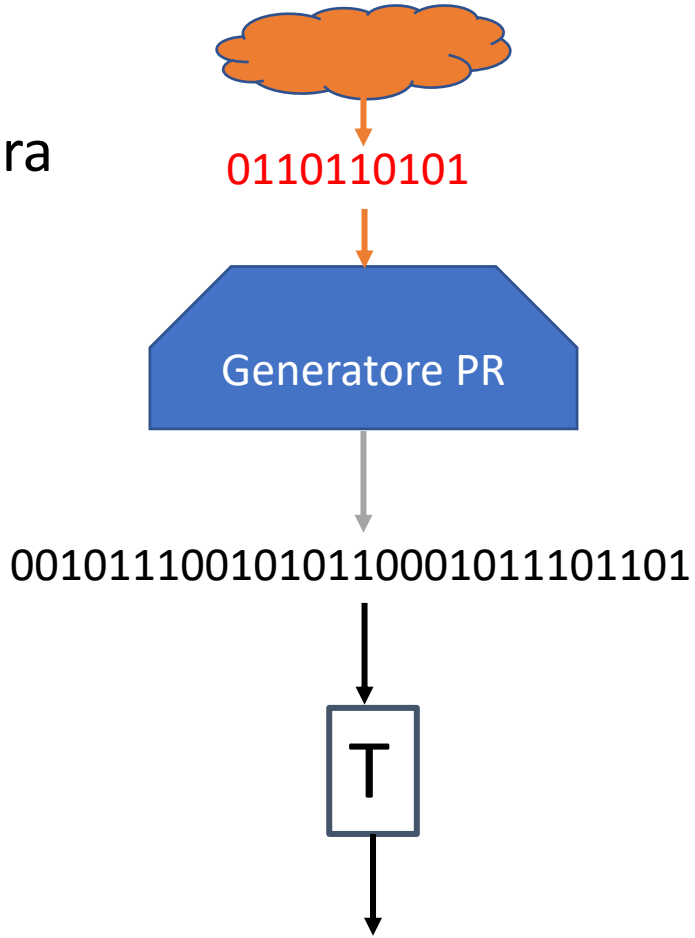
Per ogni test statistico T
efficientemente calcolabile

Se



“vero” con probabilità p
“falso” con probabilità $1 - p$

Allora

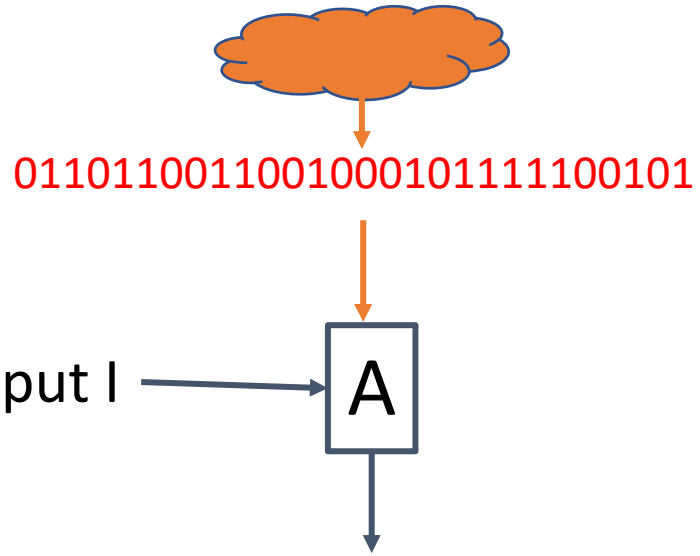


“vero” con probabilità $p \pm 0.00001$
“falso” con probabilità $1 - p \pm 0.00001$

Generatori pseudorandom

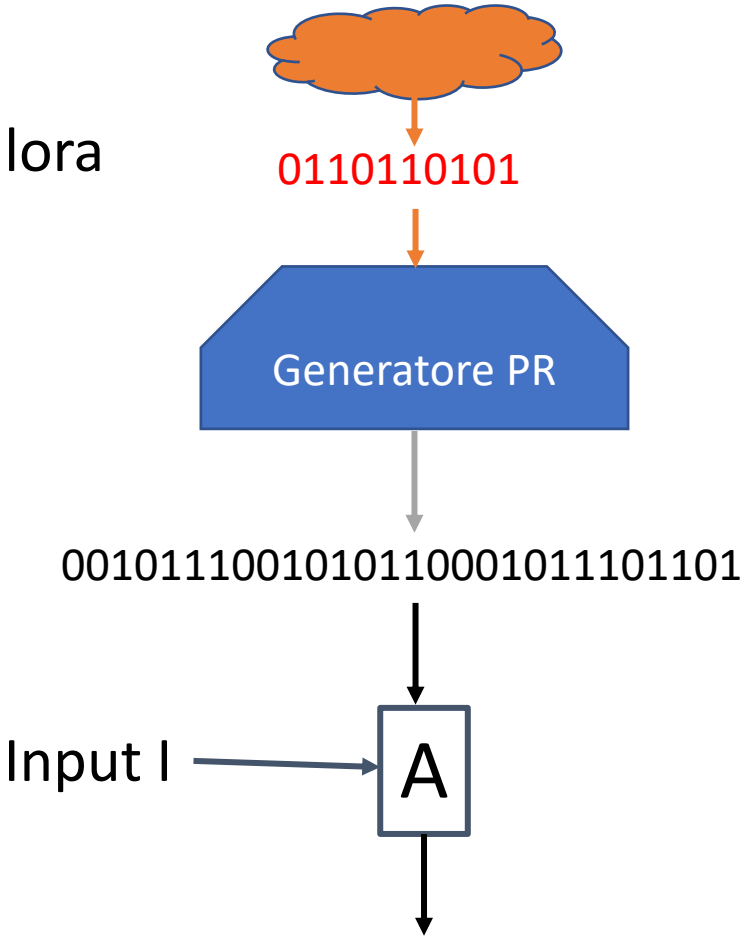
Per ogni algoritmo A efficiente

Se



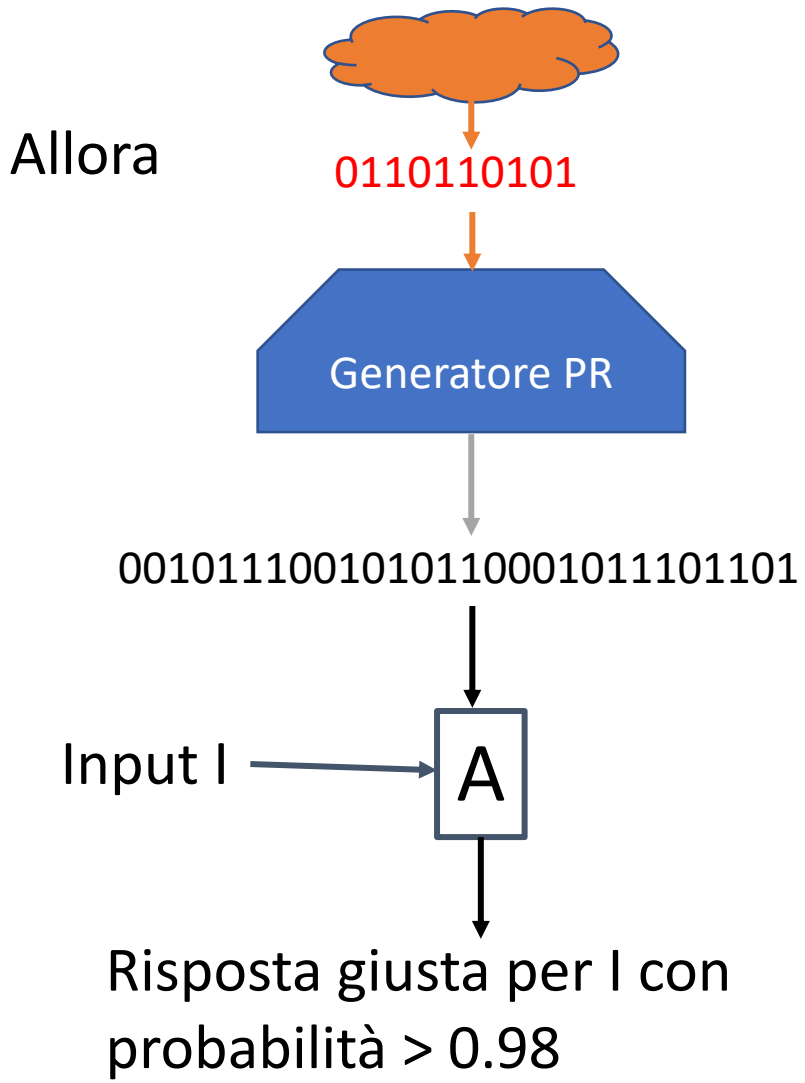
Risposta giusta per I con probabilità > 0.99

Allora



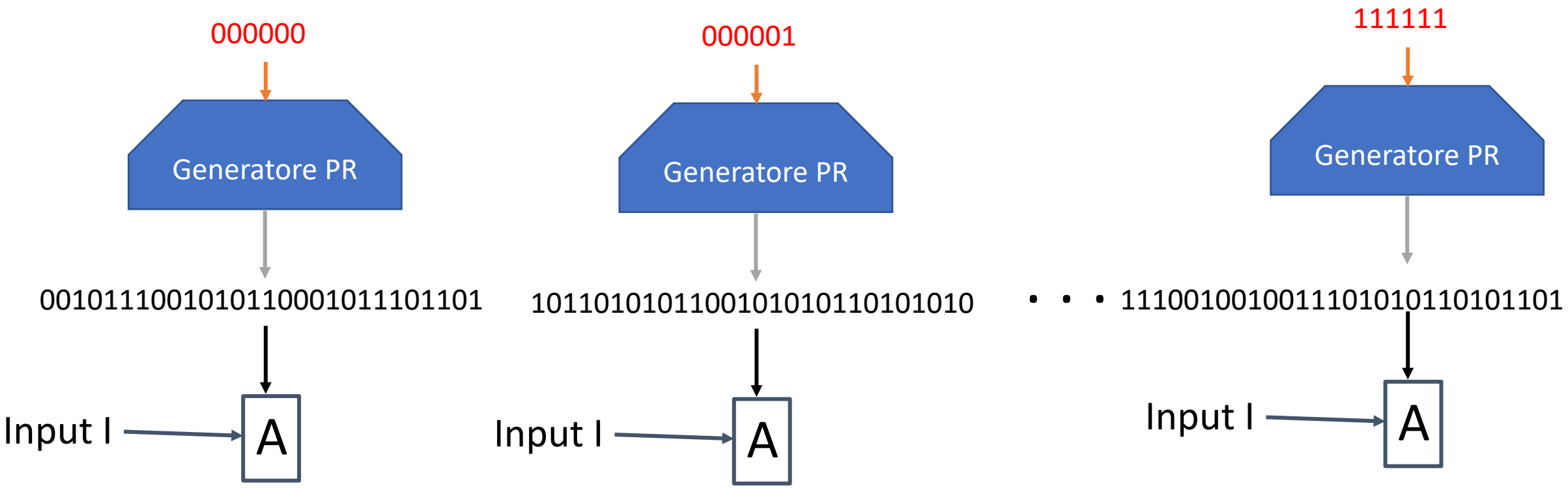
Risposta giusta per I con probabilità > 0.98

Generatori pseudorandom



- Un generatore pseudorandom consente di ridurre la quantità di randomness usata da algoritmi randomizzati
- Un algoritmo randomizzato può essere simulato deterministicamente enumerando tutte le possibili scelte casuali
- Se il generatore riduce molto il fabbisogno di randomness, questa simulazione può essere efficiente

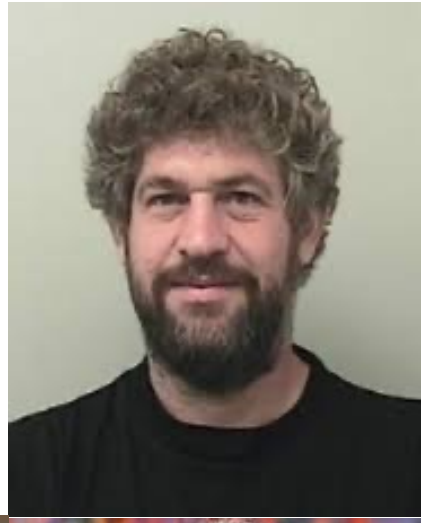
Simulazione deterministica di algoritmi randomizzati



Simulazione deterministica di algoritmi randomizzati

- Generatori pseudorandom con “seed” molto corta esistono se certe congetture standard sono vere

- (Assumendo certe congetture) tutti gli algoritmi randomizzati ammettono simulazioni efficienti completamente deterministiche
  [Nisan-Wigderson 1989]
  …
  [Impagliazzo-Wigderson 1997]
  …



Simulazione deterministica di algoritmi randomizzati

- (Assumendo certe congetture) tutti gli algoritmi randomizzati ammettono simulazioni efficienti completamente deterministiche
  [Nisan-Wigderson 1989]
  …
  [Impagliazzo-Wigderson 1997]
  …

- Congettura ($E \not\subseteq CSIZE(2^{o(n)})$) non è sempre possibile accelerare un algoritmo scrivendo codice diverso per inputs a scale diverse

Take-aways

1. L’uso appropriato di scelte casuali è necessario nella crittografia
2. È possibile dare definizioni rigorose (e molto forti) della sicurezza di sistemi crittografici
3. L’uso appropriato di scelte casuali è anche utile nel progetto di algoritmi per problemi la cui soluzione è univocamente definita
4. Gli algoritmi (3) possono essere “de-randomizzati” in modo molto efficiente se certe congetture sono vere
5. La dimostrazione di (4) usa tecniche originariamente sviluppate nella crittografia e mostra che la non-esistenza di certi algoritmi implica l’esistenza di certi altri