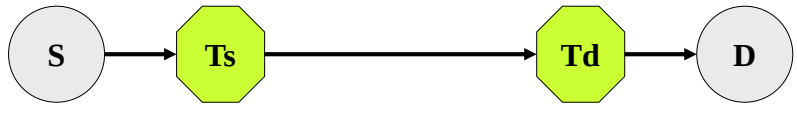
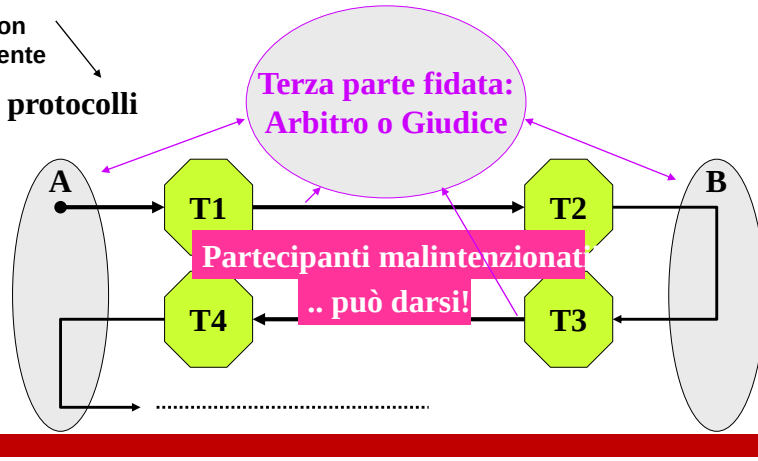
Trasformazioni per la sicurezza

1: algoritmi

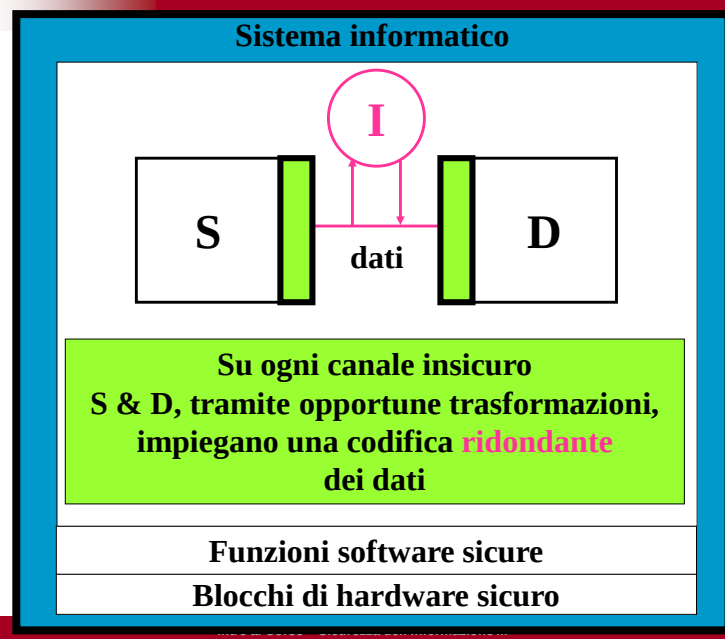


Se non sufficiente

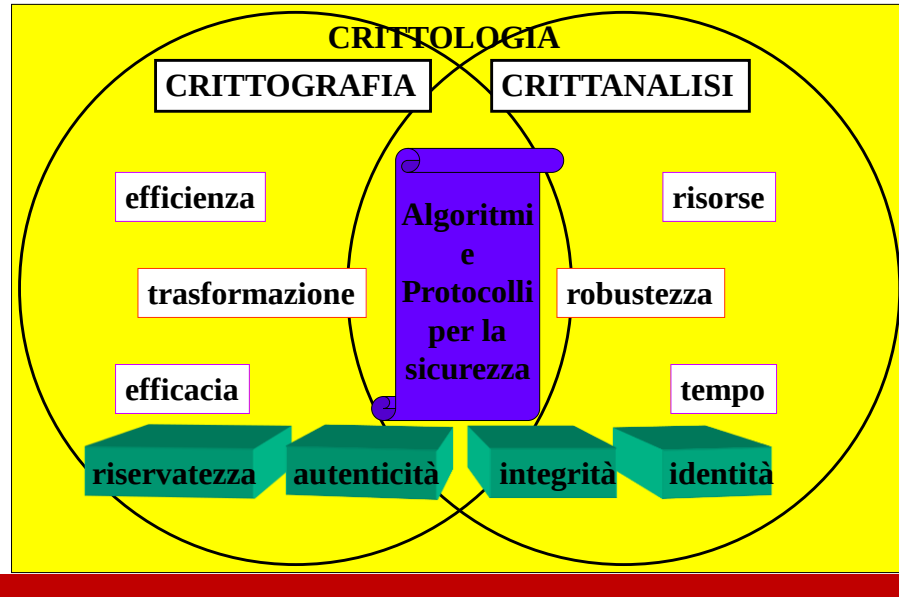
2: protocolli



La tecnica che può rendere i dati sicuri



Crittografia e Crittanalisi



1940-44

Crittografia classica

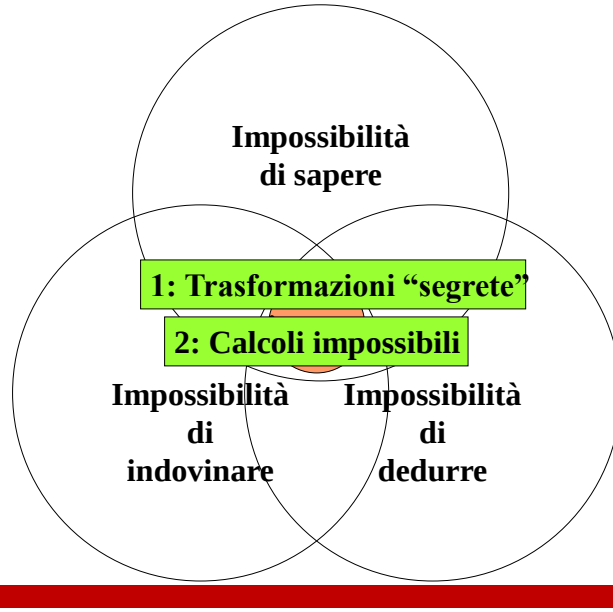
Algoritmi semplici

Algoritmi complessi

Crittografia moderna

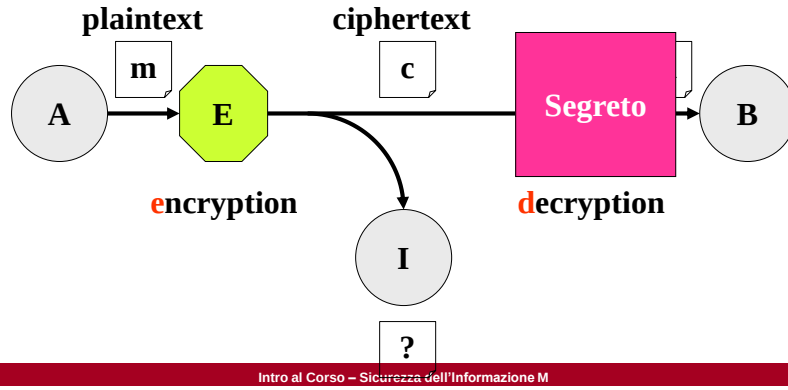
Teoria degli algoritmi
Teoria dell'informazione
Teoria dei numeri
Teoria della probabilità

I principi della difesa



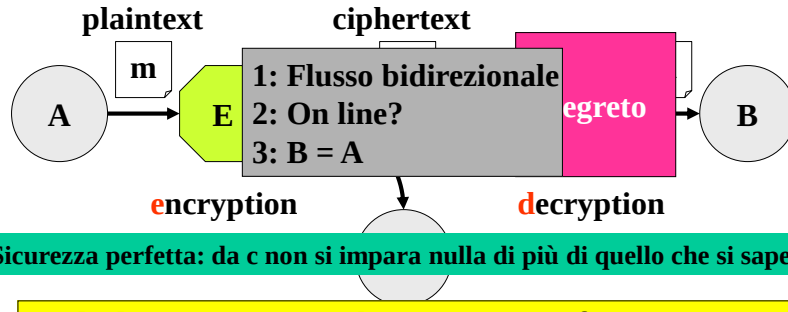
Elaborazioni per la riservatezza di un messaggio

La sorgente trasforma la rappresentazione originaria delle informazioni riservate in una rappresentazione che le renda apparentemente incomprensibili; la destinazione è l'unica a saper eseguire la trasformazione inversa



Elaborazioni per la riservatezza di un messaggio

La sorgente trasforma la rappresentazione originaria delle informazioni riservate in una rappresentazione che le renda apparentemente incomprensibili; la destinazione è l'unica a saper eseguire la trasformazione inversa



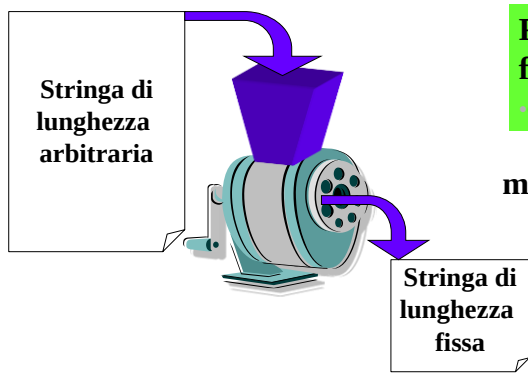
Sicurezza perfetta: da c non si impara nulla di più di quello che si sapeva già

I calcoli per mettere in chiaro un testo cifrato senza conoscere l'algoritmo di decifrazione devono essere difficili

Integrità

Integrità: rilevazione di attacchi intenzionali

La sorgente deve affiancare al documento un "riassunto" che ne rappresenti in modo pressoché univoco il contenuto; la destinazione deve ricalcolare il riassunto e confrontare i due dati



Funzione hash crittografica
facile da eseguire
oracolo casuale

m → **H** → **h = H(m)**

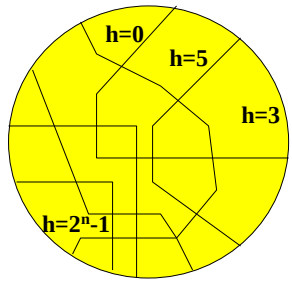
riassunto (*digest*)
o impronta (*finger-print*)
di un documento

Resistenza alle collisioni: l'individuazione di due messaggi con la stessa impronta deve essere un calcolo difficile

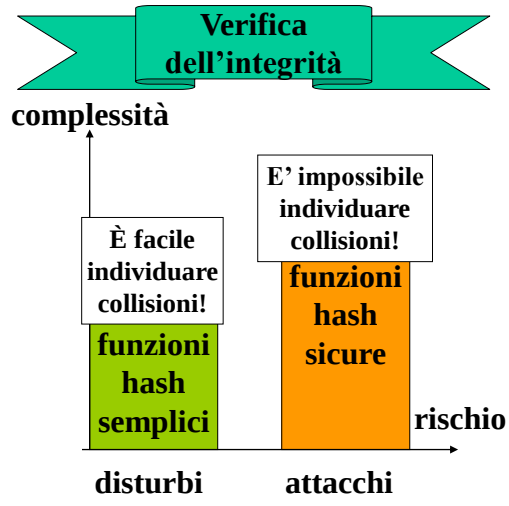
Collisioni

n bit di hash: 2^n valori d'uscita

Input space: 2^m con m > n



Le stringhe d'ingresso che forniscono uguale impronta sono dette essere in collisione

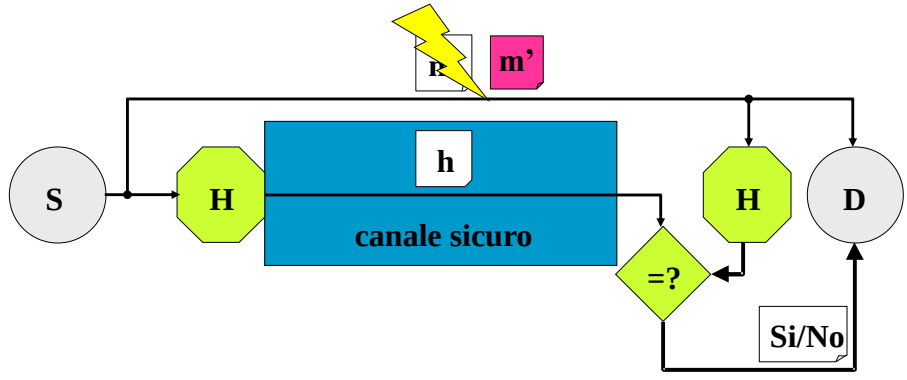


Attestazione ed accertamento dell'integrità

A-> B: m || H(M)

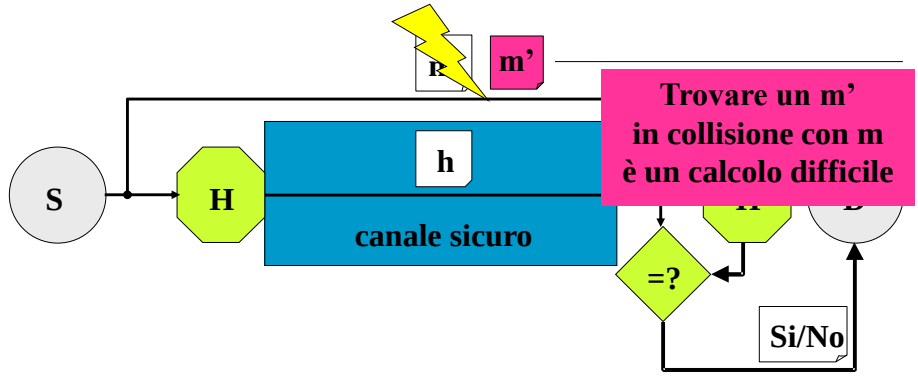
Funziona?

Attestazione ed accertamento dell'integrità



Produttori di software
Portachiavi del PGP

Attestazione ed accertamento dell'integrità



Produttori di software
Portachiavi del PGP

Protocollo per la difesa di Riservatezza & Integrità

Come possiamo combinare insieme le trasformazioni E ed H?

Protocollo per la difesa di Riservatezza & Integrità

1.	**p = m||H(m)**	***testo in chiaro concatenato con l'attestazione d'integrità***
2.	**c = E(p)**	***testo cifrato trasmesso***
3.	**p* = D(c*)= m*||H*(m)**	***testo in chiaro ricevuto***
4.	**H*(m) =? H(m*)**	***controllo d'integrità***

Protocollo per la difesa di Riservatezza & Integrità

1. **p = m** — *testo in chiaro concatenato con l'attestazione d'integrità*
2. **c = E(p)||H(m)** — *testo cifrato trasmesso*

– **Funziona?**

Protocollo per la difesa di Riservatezza & Integrità

1. **p = m** — *testo in chiaro concatenato con l'attestazione d'integrità*
2. **c = E(p)||H(m)** — *testo cifrato trasmesso*

Violazione della riservatezza: un intruso può fare ipotesi su possibili m*, fare H(m*) e controllare se H(m*)=H(m)

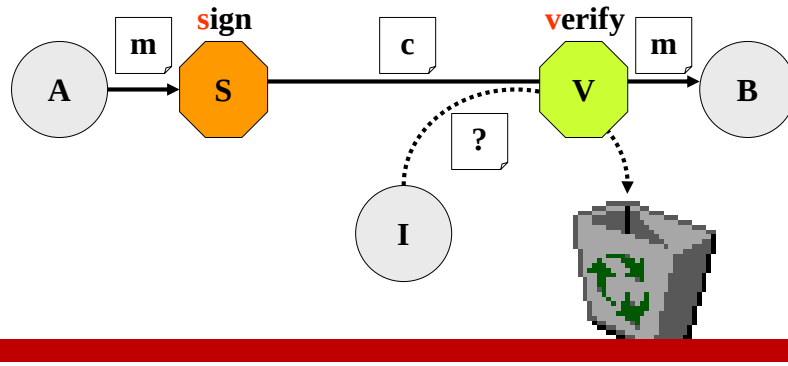
Protocollo per la difesa di Riservatezza & Integrità

1. **p = m** — *testo in chiaro concatenato con l'attestazione d'integrità*
2. **c = E(p)||H(E(p))** — *testo cifrato trasmesso*

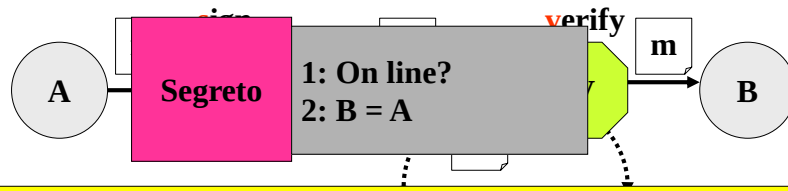
Violazione dell'integrità: un intruso può modificare E(p) e ottenere E*(p) e sostituire H(E(p)) con H(E*(p)); se m è un messaggio privo di significato (ad esempio un numero) D(E(p)) restituisce m* e la destinazione potrebbe non accorgersi che m* non è corretto. Se invece m è un messaggio dotato di significato la destinazione potrebbe accorgersi che m* non ha senso e quindi comunque non accettarlo per buono

Elaborazioni per l'autenticazione di un messaggio

La sorgente aggiunge al documento informazioni non imitabili e atte ad attestare chi l'ha predisposto; la destinazione verifica che il documento ricevuto sia stato originato proprio da chi dichiara di averlo fatto".



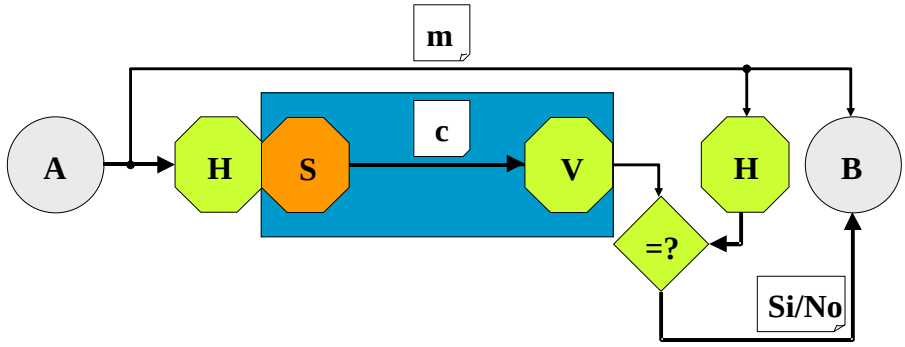
Elaborazioni per l'autenticazione di un messaggio



I calcoli per costruire un messaggio apparentemente autentico senza conoscere la trasformazione della sorgente devono essere difficili

Autenticazione di un messaggio in chiaro

1: schema firma digitale



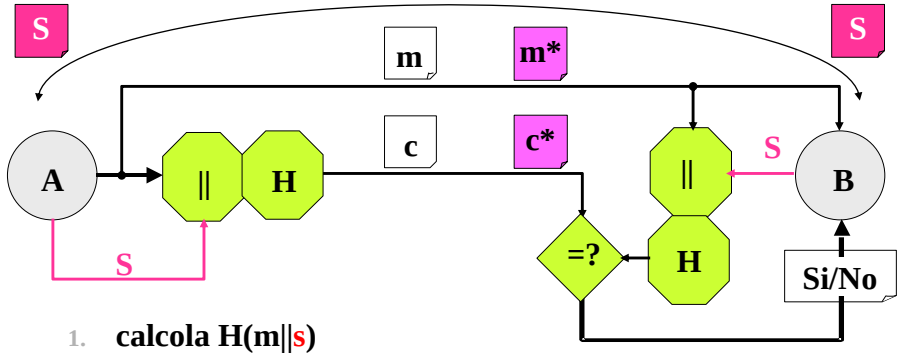
NON ripudiabile

a) canale sicuro

b) canale reso sicuro

Autenticazione di un messaggio in chiaro

2: hash del messaggio e di un segreto



1. calcola H(m||s)
2. invia m e H(m||s)

ripudiabile

H^{-1} difficile

3. riceve m*e H*(m||s)
4. calcola H(m*||s)
5. H*(m||s) =? H(m*||s)

Protocollo per la difesa di Riservatezza & Autenticazione

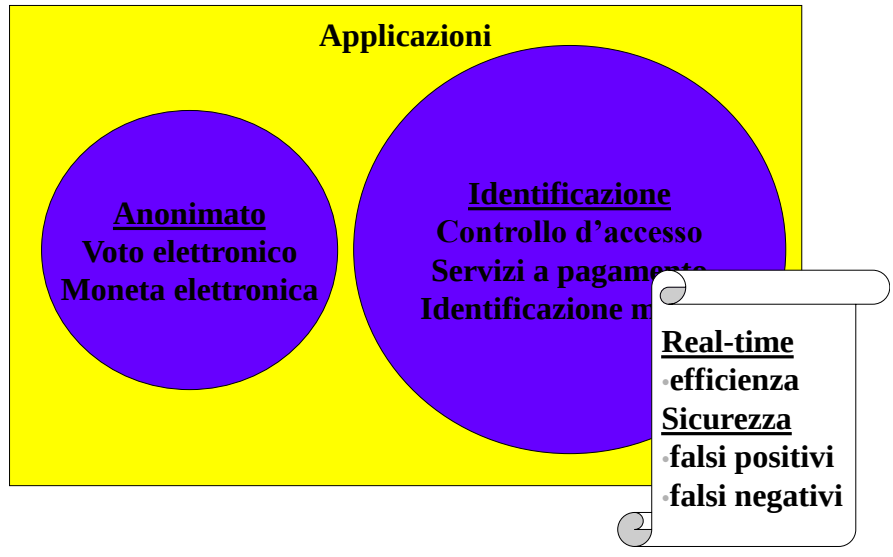
1. **p = m||H(m||s)** — *testo in chiaro concatenato con l'attestazione di autenticità*
2. **c = E(p)** — *testo cifrato trasmesso es. SSL*
3. **p* = D(c*)= m*||H*(m||s)** — *testo in chiaro ricevuto*
4. **H*(m||s) =? H(m*||s)** — *controllo d'integrità*

Protocollo per la difesa di Riservatezza & Autenticazione

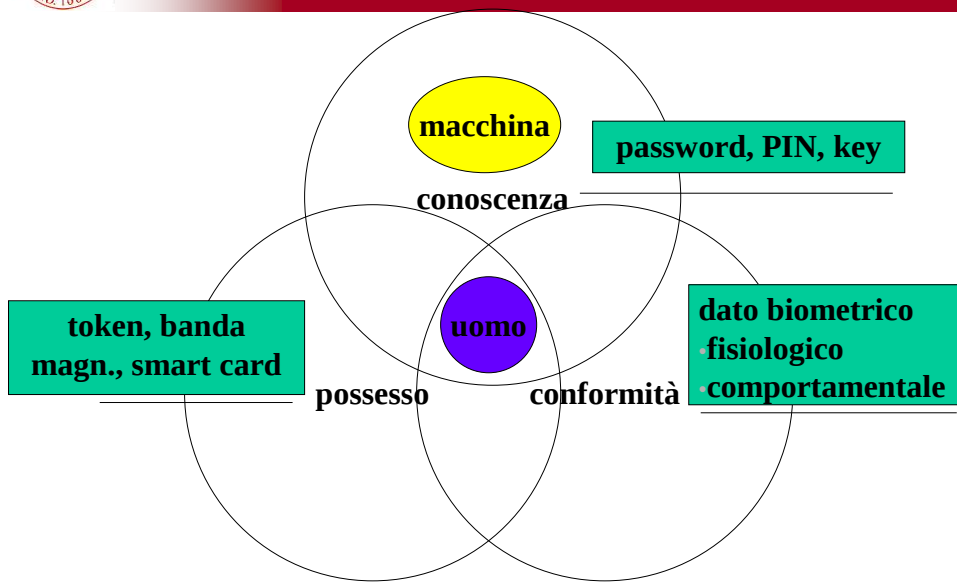
1. **p = m** — *testo in chiaro*
2. **c = E(p), H(m||s)** — *es. SSH*

1. **p = m**
2. **c = E(p), H((E(p)||s)** — *es. IPSEC*

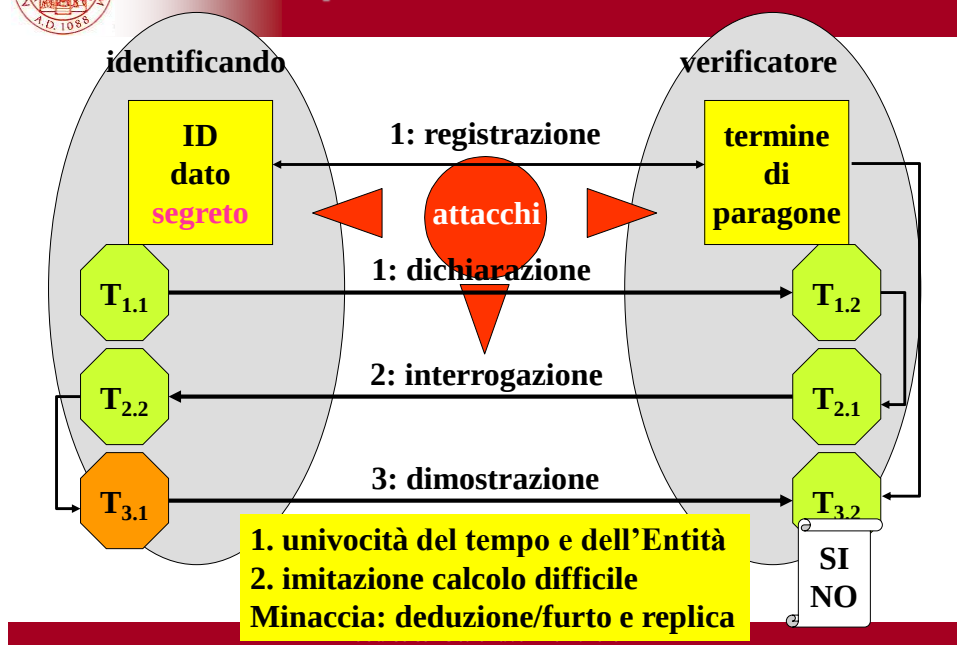
Anonimato/Identificazione



Strumenti di identificazione



Il protocollo d'identificazione



Il modello della funzione unidirezionale (one-way function)

L'intruso non deve riuscire ad invertire

- **una funzione hash**
- **una funzione di cifratura**
- **una funzione di verifica**

.....

Una funzione f è detta unidirezionale se
è *invertibile*,
***facile* da calcolare**
e se per quasi tutti gli x appartenenti al dominio di f è *difficile* risolvere per x il problema y = f (x).

Funzione unidirezionale e pseudo-unidirezionale

Elenco telefonico

N° di X?
Ricerca binaria: O(n)

X di N°?
Ricerca esauriente: O(2^n)

113? Carabinieri!

In teoria non esistono, in pratica SI:
- **Manipolazioni a livello di bit**
- **Problemi difficili della Teoria dei numeri**

Una funzione F è detta pseudo-unidirezionale (trapdoor one-way) se appare come unidirezionale per chiunque non sia in possesso di una particolare informazione sulla sua costruzione

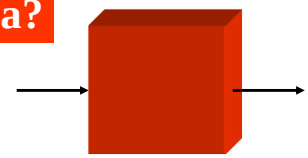
Trasformazioni segrete

Trasformazioni segrete

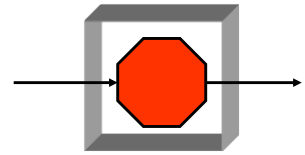
Decifrazione **Autenticazione**

Segretezza si, ma di che cosa?

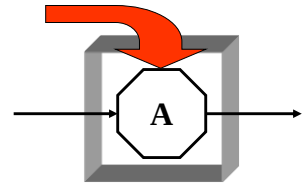
- **Macchina**



- **Algoritmo**



- **Parametro**



Vulnerabilità delle macchine e degli algoritmi segreti

- Ispezione
- Installazione
- Progetto
- Produzione
- Certificazione

Algoritmo pubblico e parametro segreto

Kerckhoffs : “La criptogràphie militaire”1883

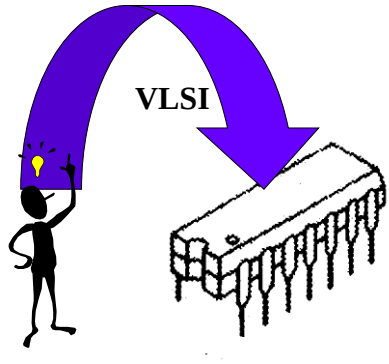
☺

Responsabilità dell’utente — **cassaforte**

Valutazione pubblica — **AES**

Software open e free — **JCE, JSSE**

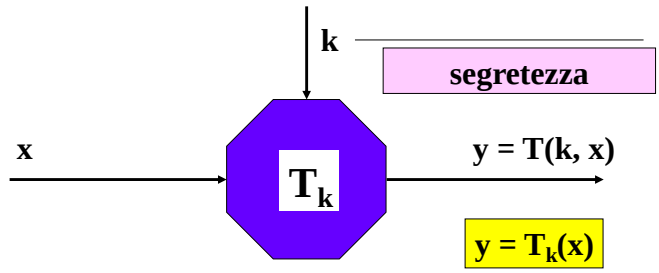
Algoritmi segreti e parametri segreti



Clipper

GSM

Segretezza di una trasformazione: algoritmo con chiave segreta!



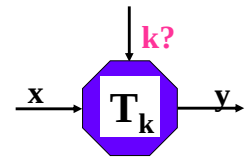
insieme delle trasformazioni: $T = \{t_1, t_2, \ldots, t_N\}$
spazio delle chiavi: $K = \{k_1, k_2, \ldots, k_N\}$

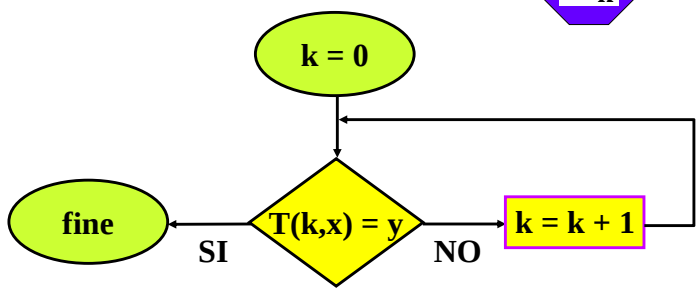
N grandissimo!

Attacco con forza bruta alla segretezza di una chiave

IPOTESI:

1. $0 \leq k \leq 2^n-1$, $y = T(k,x)$
2. E' nota almeno una coppia x, y

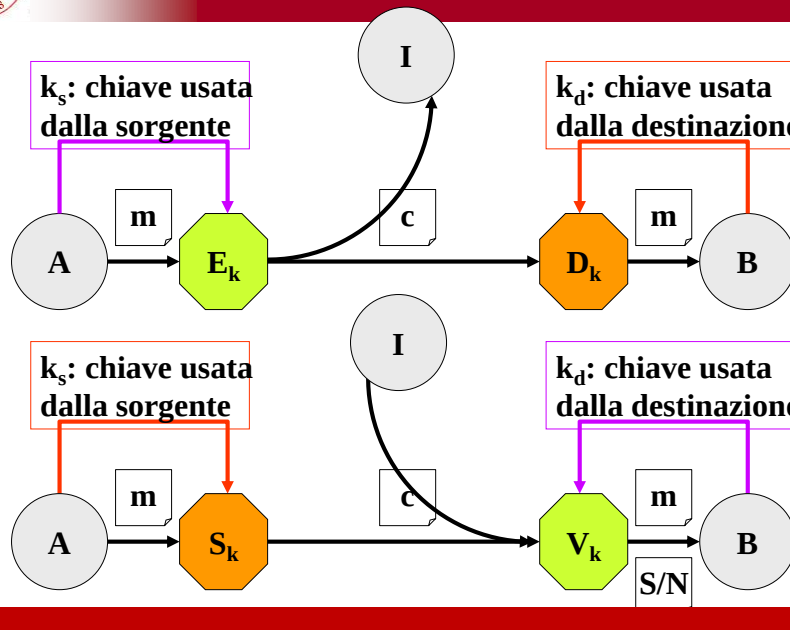




O(exp(n))
n > 128

Chiavi

Riservatezza e autenticazione



La relazioni tra le chiavi

ks ∈ K
kd ∈ K

K → K : ks = f(kd).

Algoritmo a chiavi simmetriche o simmetrico:
le chiavi ks, kd sono o *uguali* o *facilmente calcolabili* una dall'altra; la situazione usuale è
ks = kd.

Algoritmo a chiavi asimmetriche o asimmetrico:
le chiavi ks, kd sono *diverse* ed una delle due è *difficilmente calcolabile* dall'altra

Autenticazione	Riservatezza
kd = f(ks) facile!	ks = f(kd) facile!
ks = f^{-1}(kd) difficile!	kd = f^{-1}(ks) difficile!

La relazioni tra le chiavi

ks ∈ K
kd ∈ K

K → K : ks = f(kd).

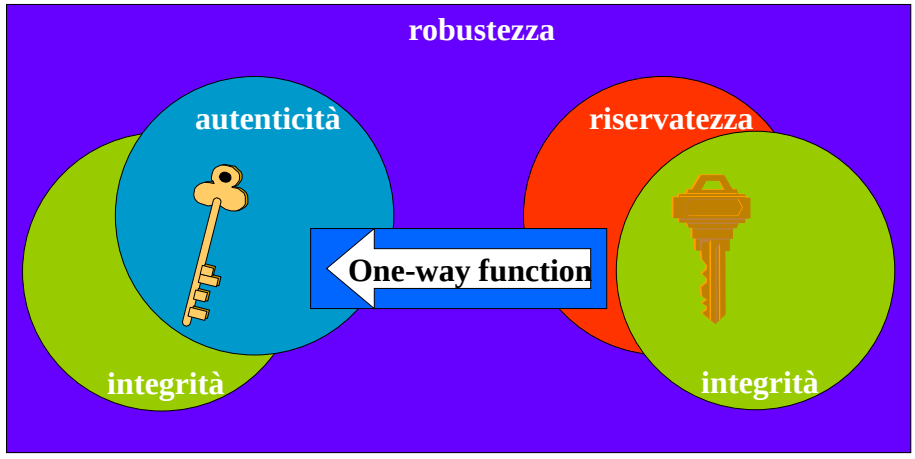
ks & kd segrete

ks segreta
kd pubblica

kd segreta
ks pubblica

Proprietà delle chiavi simmetriche

Proprietà delle chiavi asimmetriche



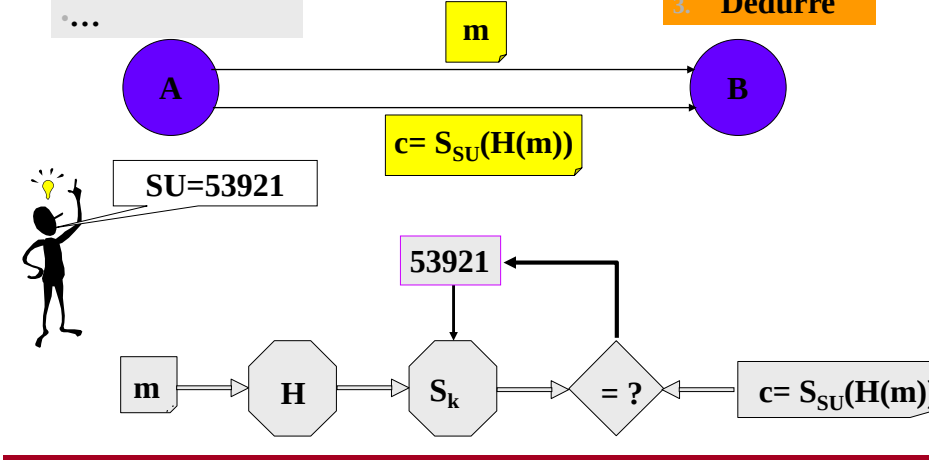
Chiave pubblica
PU

Chiave privata
SU

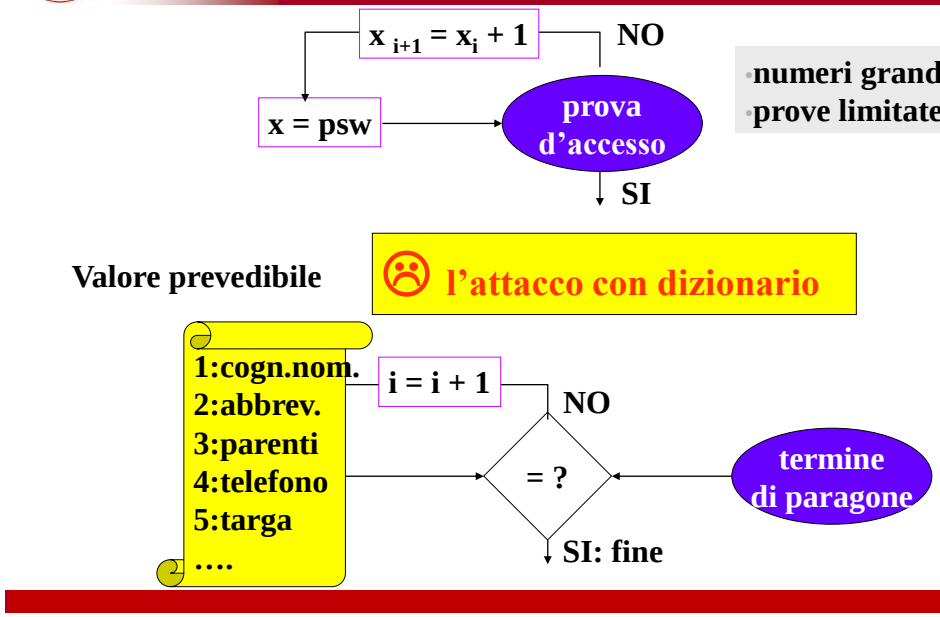
Dati segreti

- Chiave condivisa
- Chiave privata
- Prova d'identità
- …

1. Indovinare
2. Intercettare
3. Deduzione

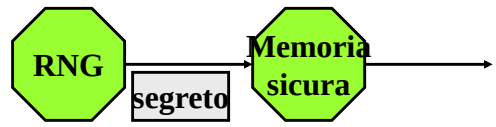


Attacco con forza bruta e con dizionario



Tirare ad indovinare

I simboli della stringa che rappresenta un segreto devono essere molti e scelti a caso

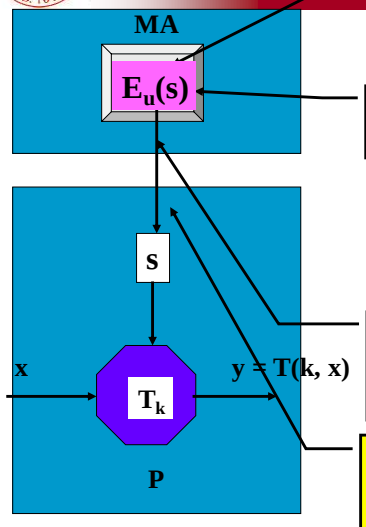


Valutazioni

1. **Tirare a indovinare** 2^{-n} **> 20 bit**
2. **Provare per k volte** $k.2^{-n}$ **> 30 bit**
3. **Ricerca esauriente** **O(exp(n)) > 80 bit** $\rightarrow 10^{11}$ **anni MIPS**

Un dato segreto deve essere frequentemente modificato

Memorizzazione ed uso di una chiave segreta



Solo il proprietario della chiave segreta deve avere accesso alla cella in cui è memorizzata.

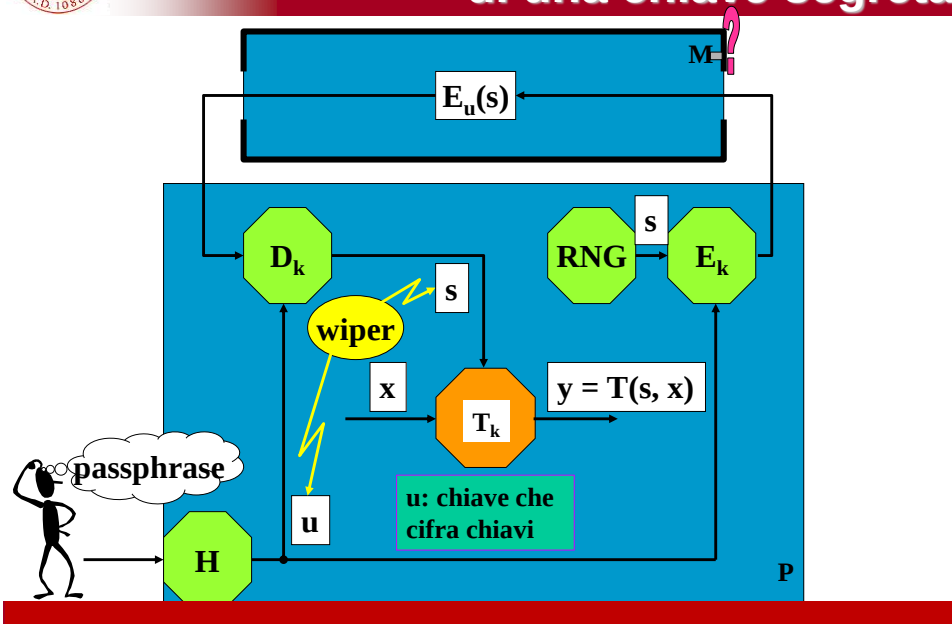
La chiave segreta deve essere memorizzata in forma cifrata.

1. ~~Indovinare~~
2. Intercettare
3. Dedurre

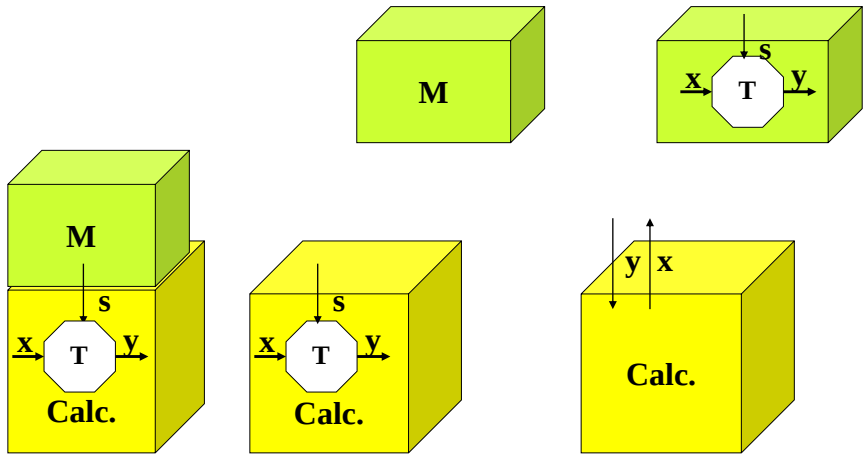
La forma cifrata deve essere mantenuta anche nella comunicazione della chiave dalla memoria ausiliaria al processore.

L'unità di elaborazione deve decifrare la chiave, impiegarla nell'esecuzione dell'algoritmo crittografico e cancellare poi accuratamente il dato dalla sua memoria di lavoro.

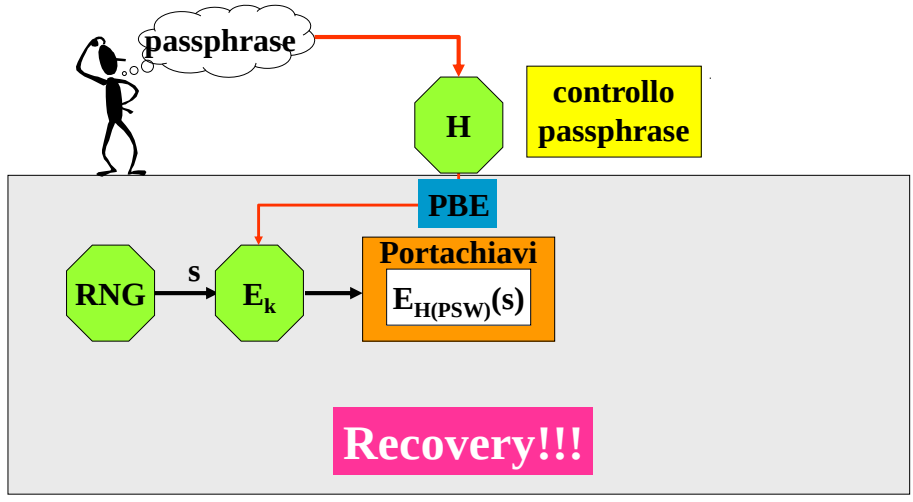
Generazione, memorizzazione ed uso di una chiave segreta



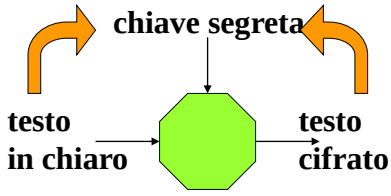
HardDisk < MemoryCard < SmartCard



Memorizzazione della chiave su File System



Deduzione di un segreto dal suo uso



☹ l'attacco con statistiche

1. **Indovinare**
2. **Intercettare**
3. **Dedurre**

ATTACCO	CONOSCENZE DELL'INTRUSO
con solo testo cifrato	linguaggio e probabilità d'occorrenza
con testo in chiaro noto	coppie di testo in chiaro e cifrato
con testo in chiaro scelto	testi cifrati di testi in chiaro scelti
con testo cifrato scelto	testi in chiaro di testi cifrati scelti

Contromisura preventiva

l'uscita di un algoritmo crittografico deve apparire come una variabile aleatoria che assume con eguale probabilità tutti i suoi possibili valori

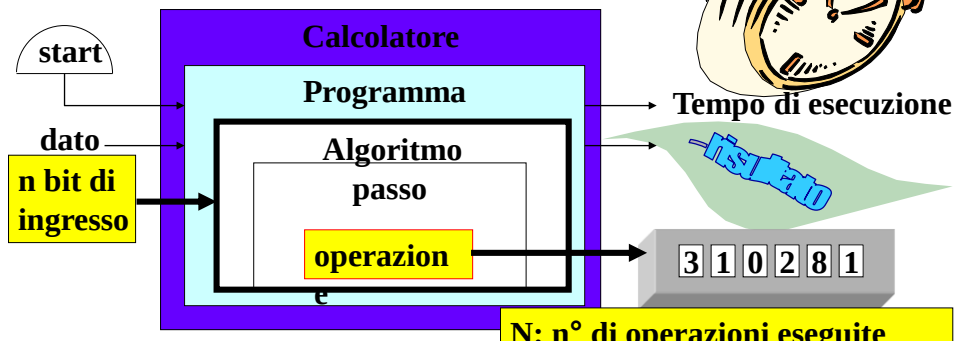
Complessità computazionale di un algoritmo

Teoria della complessità computazionale

Calcoli FACILI e Calcoli DIFFICILI

Indicatori di complessità:

- **tempo di esecuzione,**
- **memoria occupata dal programma,**
- **memoria occupata dai dati**



Teoria della Complessità di un Algoritmo

Tempo di esecuzione di un algoritmo: numero di operazioni N che occorre eseguire per terminarlo quando il dato d'ingresso è rappresentato da una stringa di n bit (n = log [valore del dato])

$$N = f(n)$$

***In generale, a parità di* n*, si hanno diversi valori di* N.**

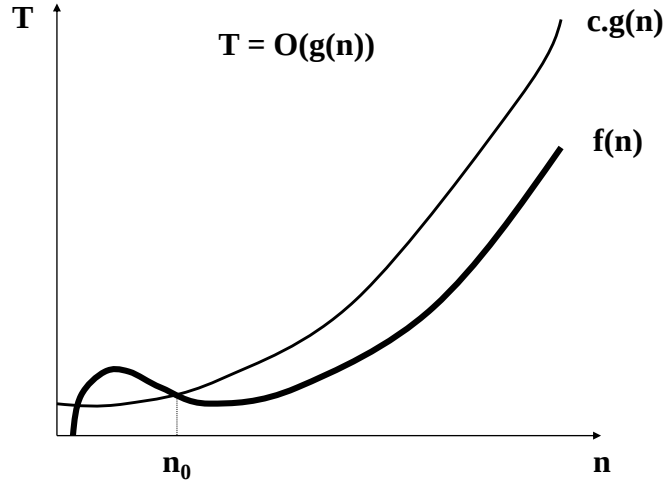
Tempo di esecuzione nel caso peggiore: numero massimo di operazioni
N_{max} che occorre eseguire per qualsiasi dato d'ingresso di n bit

$$N_{max} = f(n)$$

***Si considera la modalità d'incremento di* N_{max} *al crescere senza limiti di* n**

Andamento asintotico del tempo di esecuzione nel caso peggiore detto Ordine di grandezza del tempo di esecuzione: T = O(g(n)) ove g(n) è una funzione tale che $0 \leq f(n) \leq c.g(n)$ per $n \geq n_0$ e c cost.

La notazione del "grande O"



Se è nota l'espressione di f(n), si considera come g(n) il termine di f(n) che cresce più rapidamente con n

Classificazioni

Classificazione degli algoritmi

1. con **tempo polinomiale**:
   $T = O(n^t)$ con t esponente più grande in g(n),
2. con **tempo esponenziale**:
   $T = O(b^n)$, con b costante, o anche T = O(exp (n))

Classificazione dei problemi

1. **facile**, se esiste un algoritmo polinomiale in grado di risolverlo su una macchina di Turing deterministica,
2. **difficile**, se non sono stati fino ad ora individuati (e probabilmente non saranno mai individuati) algoritmi che lo risolvono in tempo polinomiale

Complessità e Sicurezza

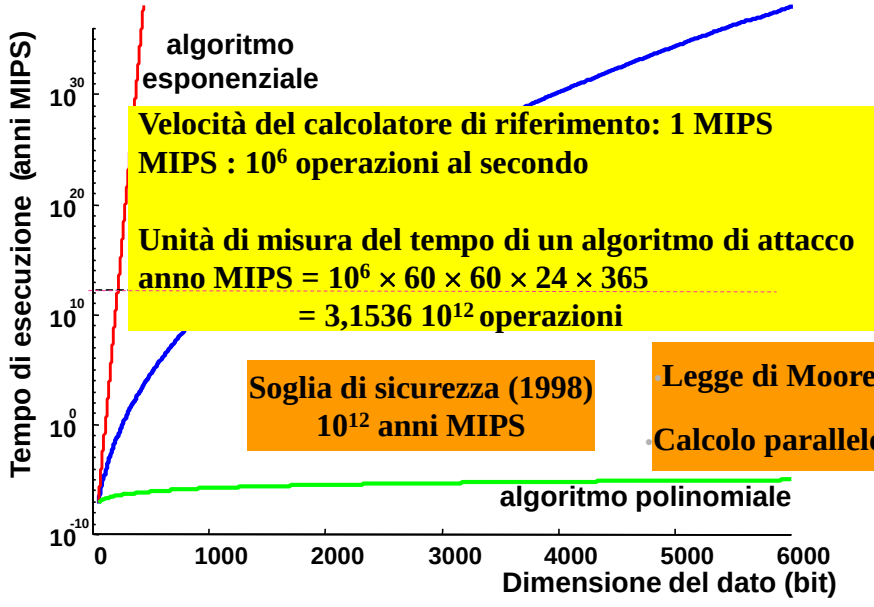
- Caso peggiore e istanze facili
- Modalità di incremento e valori di n

R10:"ogni algoritmo che consente di difendere una proprietà critica dell'informazione deve avere tempo polinomiale"
ESEMPI: O(1), O(n), $O(n^3)$

R11:"ogni algoritmo che consente di violare una proprietà critica della informazione deve avere tempo esponenziale"
ESEMPI: O(exp (n)), O(exp (n) $^{1/2}$)

- Algoritmi sub-esponenziale: $O(\exp((n)^{\alpha}(\ln(n)^{1-\alpha}))$ con $0 < \alpha < 1$

Anni MIPS e dimensione del dato



Velocità del calcolatore di riferimento: 1 MIPS
MIPS : 10^6 operazioni al secondo

Unità di misura del tempo di un algoritmo di attacco
anno MIPS = $10^6 \times 60 \times 60 \times 24 \times 365$
= 3,1536 10^{12} operazioni

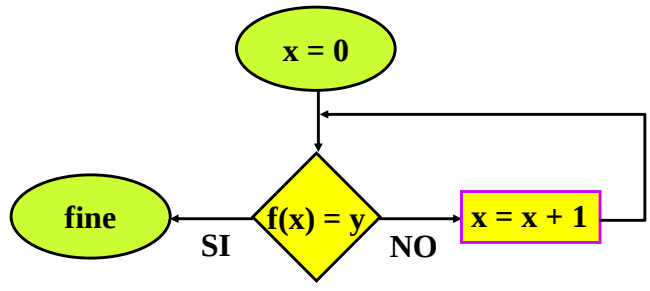
Soglia di sicurezza (1998)
10^{12} anni MIPS

- **Legge di Moore**
- **Calcolo parallelo**

Livello di sicurezza

1-L'algoritmo di ricerca esauriente risolve ogni problema difficile

PROBLEMA: inversione di una funzione
ES. Per quale intero x, con $0 < x < 2^n-1$, vale la proprietà f(x) = y?



Nel caso peggiore occorrono 2^n passi: O(exp(n))

Livello di sicurezza

2-La complessità di ogni algoritmo di attacco può essere misurata facendo riferimento all'attacco con forza bruta

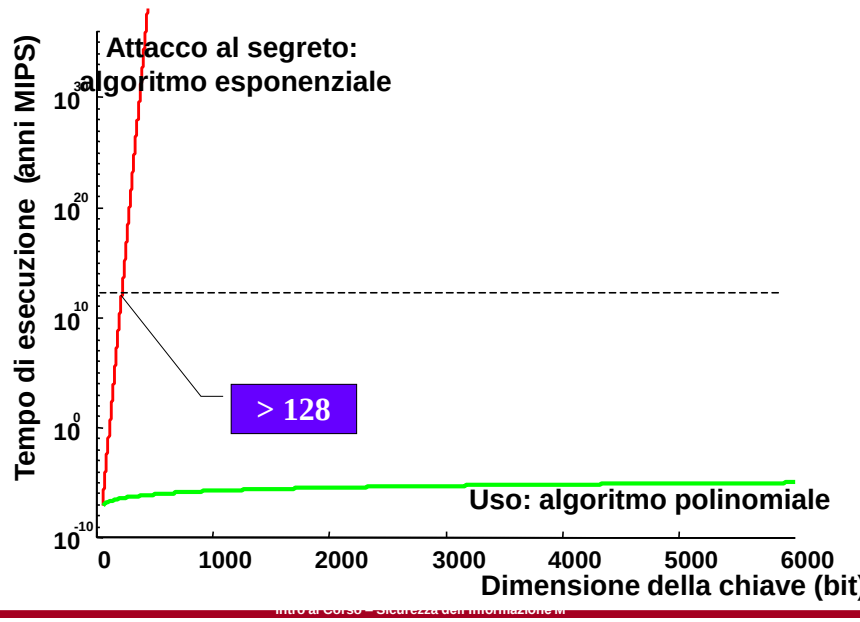
Algoritmo A: N passi per terminare nel caso peggiore

Si individua un n tale che $2^n \leq N < 2^{n+1}$
in altre parole si confronta la complessità di A con quella dell'algoritmo di ricerca esauriente
Si assegna all'algoritmo A un livello di sicurezza di n bit

Misura di robustezza indipendente dal calcolatore e dalle operazioni

10^{12} anni MIPS equivalgono ad un livello di sicurezza di 85 bit.
Attualmente si cerca di conseguire un livello di 128 bit

Dimensionamento di una chiave simmetrica



Dimensionamento di una chiave asimmetrica