Appunti di Crittografia

Simone Ianniciello

A.A. 2020/2021

Contents

Chapter 1

Introduzione

1.1 La crittologia

La crittografia e' lo studio delle tecniche matematiche per:

Crittografia Metodi di cifratura.

Crittoanalisi Metodi di interpretazione.

Crittografia + Crittoanalisi = **Crittologia**

1.1.1 Scenario

A vuole spedire un messaggio a B, ma E sta' ascoltando il messaggio. Per proteggere la comunicazione, A e B utilizzano dei metodi di cifratura.

1.1.2 Cifratura

MSG	Insieme dei messaggi in chiaro
CRITTO	Insieme dei crittogrammi
C: MSG $\rightarrow$ CRITTO	Funzione di crittazione.
D: CRITTO $\rightarrow$ MSG	Funzione di decrittazione.

C e D sono una l'inversa dell'altra: $D(c) = D(C(m)) = m$
C e' iniettiva (messaggi diversi corrispondono a crittogrammi diversi).

1.2 Classificazione dei cifrari

I cifrari si dividono in:

Cifrari per uso ristretto Le funzioni C e D devono essere segrete; Poco pratici per la crittografia *di massa*.

Cifrari per uso generale Si basano su un metodo a chiave. C e D sono pubbliche ma la chiave deve essere nota ai soli interessati del messaggio.

1.2.1 Cifrari per uso generale

Le definizioni di C e D diventano:
C: MSG * KEYS → CRITTO
D: CRITTO * KEYS → MSG

Se un crittoanalista entra in possesso di una chiave, basta cambiarla. Esempi di cifrari a chiave segreta: 3DES, RC5, IDEA, AES.

Attacco esauriente (bruteforce) Il crittoanalista dovrebbe provare tutte le chiavi finché non trova quella giusta per decrittare il messaggio. Quasi impossibile da effettuare su chiavi abbastanza grandi (>20chars).

1.3 Attacchi

Gli attacchi possono avere successo completo (Si scopre la funzione D, compresa di chiave), oppure possono avere successo limitato (Si scopre solo qualche informazione su un messaggio).

1.3.1 Attacchi al sistema crittografico

Cypher Text Attack	Il crittoanalista rileva una serie di crittogrammi $c_1, \ldots, c_r$.
Known Plain-Text Attack	Il crittoanalista conosce una serie di coppie $(c_1, m_1), \ldots, (c_r, m_r)$.
Chosen Plain-Text Attack	Il crittoanalista sceglie una serie di $m_1, \ldots, m_r$ e si procura i relativi $c_1, \ldots, c_m$.

1.3.2 Attacchi Man-In-The-Middle MITM

Il crittoanalista si inserisce nel canale di comunicazione e blocca tutti i messaggi diretti. Puo' anche sostituire i messaggi originali con dei messaggi propri.

Condizione normale	Attacco MITM
$A \rightleftarrows B$	$A \rightleftarrows E \rightleftarrows B$

1.4 Cifrari perfetti

I cifrari perfetti sono totalmente sicuri, ma richiedono operazioni molto complesse perciò sono usati solo in condizioni estreme. In essi m e c sono totalmente scorrelati tra loro.

1.4.1 One-Time Pad

E un cifrario perfetto ma ha svantaggi enormi che lo rendono quasi inutilizzabile:

- Richiedono una chiave segreta nuova e perfettamente casuale per ogni messaggio.
- La chiave deve essere lunga quanto il messaggio.

1.4.2 Cifrari sicuri

I cifrari che vengono utilizzati ad ora non sono cifrari perfetti ma sono dichiarati sicuri. Cio' significa che non sono mai stati violati prima d'ora perché richiedono la risoluzione di problemi matematicamente difficili (Non essendo mai stato dimostrato $P \not\equiv NP$ non siamo *certi* che siano inviolabili).

1.5 Cifrari in uso

1.5.1 AES

E' un cifrario simmetrico (la stessa chiave viene utilizzata per crittare e decrittare), a blocchi (il messaggio e' diviso in blocchi lunghi come il messaggio). E' pubblicamente noto e utilizzato per comunicazioni *non classificate*. Si utilizzano chiavi brevi (128 o 256 bit).

Distribuzione delle chiavi Le chiavi non sono stabilite direttamente da chi le deve usare ma da sistemi sicuri in Internet. Nel 1976 viene proposto un sistema per lo scambio di chiavi su un canale insicuro.

1.5.2 Cifrari a chiave pubblica (asimmetrici)

Viene generata una coppia di chiavi diverse:

$\mathbf{k_{pub}}$ Usata per cifrare, nota a tutti.

$\mathbf{k_{priv}}$ Usata per decifrare, nota solo a chi deve ricevere il messaggio

Le funzioni diventano quindi:

$$c = C(m, k_{pub})$$
$$m = D(c, k_{priv})$$

E' quindi possibile distribuire $\mathrm{k_{pub}}$ pubblicamente sulla rete, e sono chi la ha generata puo' decrittare i messaggi crittati con essa. C e' una funzione one-way, trap-door. Un crittoanalista non puo' ricavare informazioni sui messaggi pur conoscendo C, D, e k_{pub}. Una prima implementazione di questo metodo e' l'RSA.

Vantaggi	Non e' richiesto lo scambio segreto di chiavi Il numero di chiavi necessarie per n utenti e' $2n$.
Svantaggi	Sono molto piu' lenti del cifrari simmetrici Sono esposti ad attacchi di tipo Chosen Plain-Text.

1.5.3 Cifrari ibridi

Si utilizza un cifrario a chiave pubblica per lo scambio delle chiavi segrete da utilizzare per i cifrari simmetrici. Si risolve così il problema della lentezza (Il sistema a chiave pubblica viene utilizzato solo per cifrare poche decine di byte della chiave) e il problema dell'attacco CPT (Se la chiave secreta risulta come una sequenza casuale, il crittoanalista non la sa distinguere da un qualunque altro output della cifratura a chiave pubblica).

1.5.4 Applicazioni sulla rete

I sistemi crittografici attuali devono garantire altri tre aspetti oltre alla segretezza delle comunicazioni:

Identificazione	Il sistema deve accertare l'identità di chi richiede l'accesso ai suoi servizi.
Autenticazione	Il destinatario di un messaggio deve potersi accertare che esso non sia stato manomesso o sostituito dam terzi
Firma digitale	Una volta apposta la *firma* sul messaggio, il mittente non puo' piu' ricusarne la paternità.

Chapter 2

Rappresentazione matematica di oggetti

2.1 Definizioni

Alfabeto Per alfabeto si intende un insieme finito di caratteri o simboli.

Oggetto Un oggetto e' una sequenza ordinata di elementi dell'alfabeto.

2.2 Alfabeti e sequenze

Considero l'alfabeto Γ con N oggetti da rappresentare. Si considera $s = |\Gamma|$ la cardinalità di Γ.

Con $d(s, N)$ si intende la lunghezza della sequenza piu' lunga .

$d(s, N)$	E' la lunghezza della sequenza piu' lunga della rappresentazione scelta.
$d_{min}(s, N)$	Valore minimo di $d(s, N)$ tra tutte le rappresentazioni possibili.

Tanto piu' si avvicina d_{min} a d, tanto e' migliore la rappresentazione.

2.2.1 Rappresentazione binaria

	2^k	Numero di sequenze diverse di lunghezza k
$\mathbf{s = 2, \Gamma = 0, 1}$	$2^{k+1} - 2$	Numero di sequenze di lunghezza massima k
	$log_2(N + 2) - 1$	Lunghezza della sequenza piu' lunga rappesen

Chapter 3

Teoria della Calcolabilità

3.1 Problemi computazionali

Sono classificati in:

- Non decidibili
- Decidibili
  - Trattabili *(polinomiali)*
  - Intrattabili *(esponenziali)*

3.1.1 Calcolabilità e Complessità

Calcolabilità E' lo studio delle nozioni di algoritmo e di problema non decidibile.

Complessità E' lo studio di algoritmi efficienti e di problemi intrattabili.

3.1.2 Problema della rappresentazione

Algoritmo Sequenza finita di operazioni, completamente e univocamente determinate. Gli algoritmi possono essere formulati con modelli diversi come: modello matematico, algoritmo in pseudocodice, programma eseguibile... Qualunque modello venga scelto, gli algoritmi devono essere descritti perciò sono possibilmente **infiniti** ma **numerabili**.

Problemi computazionali Sono funzioni matematiche che associano ad ogni insieme di input un risultato; **non** sono numerabili. Cio' significa che:

$$\#\{\text{Problemi}\} >> \#\{\text{Algoritmi}\}$$

Non esiste quindi un algoritmo di calcolo per ogni problema.

Il problema dell'arresto E' la dimostrazione data da Turing nel 1930 dell'esistenza di problemi non decidibili.

Prendiamo in considerazione il generico algoritmo

$$A : \{I\} \rightarrow \{0, 1\}$$

Che, in base a I puo' terminare o non terminare. Adesso poniamo per assurdo che esista un altro algoritmo $Arresto(A, D)$ che in tempo finito ritorna:

true se $A(D)$ termina

false se $A(D)$ non termina

$Arresto$ non puo' semplicemente simulare il comportamento di $A(D)$ perché se esso non terminasse, non terminerebbe neanche $Arresto$. Se l'algoritmo $Arresto$ esistesse, esisterebbe anche l'algoritmo $Paradosso$ definito come:

Algorithm 1: Paradosso(A)

```
while Arresto(A, A) do
    ;
```

Se provo ad eseguire $Paradosso(Paradosso)$

$Paradosso(Paradosso)$ termina
$Arresto(Paradosso, Paradosso) = 0$
$Paradosso(Paradosso)$ non termina !!! ERR !!!

Cio' significa che l'algoritmo $Paradosso$ **non puo' esistere** (quindi neanche $Arresto$)

3.2 Tesi di Church-Turing

Tutti i *(ragionevoli)* modelli di calcolo risolvono la stessa classe di problemi; perciò la decidibilità e' una proprietà del problema e non del modello utilizzato.

Chapter 4

Teoria della Complessità

4.1 Problemi

4.1.1 Legenda

Π Problema

I Insieme delle istanze in ingresso

S Insieme delle soluzioni

4.1.2 Tipologie di problemi

Problemi decisionali

- $S = \{0, 1\}$
- Istanze positive: $x \in I$ t.c. $\Pi(x) = 1$
- Istanze negative: $x \in I$ t.c. $\Pi(x) = 0$

Problemi di ricerca

- S *"libera"*
- Trovare *una* soluzione al problema.

Problemi di ottimizzazione

- S *"libera"*
- Trovare la miglior soluzione $s \in S$

I problemi di interesse pratico sono spesso di ottimizzazione. E' possibile pero' esprimerli sotto forma di problemi decisionali:

- MAX-CLIQUE(G): Richiede di trovare la CLIQUE piu' grande in un grafo G.
- CLIQUE(G, k): Chiede se esiste una clique in G di almeno k vertici; non e' piu' difficile di MAX-CLIQUE.

4.2 Classi

4.2.1 Classi di Complessità

Dati Π e A, diciamo che A risolve Π se: $\exists x \in I$ t.c. $A(x)\Pi(x) = true$.

4.2.2 Classi Time e Space

$Time(f(n))$ e' l'insieme dei problemi decisionali risolvibili in tempo $O(f(n))$

$Space(f(n))$ e' l'insieme dei problemi decisionali risolvibili in spazio $O(f(n))$

4.2.3 Classe P

E' la classe dei problemi risolvibili in tempo polinomiale *($O(n^c)$, c costante, n dati in ingresso)*

4.2.4 Classe PSpace

E' la classe dei problemi risolvibili in spazio polinomiale *($O(n^c)$, c costante, n dati in ingresso)*

4.2.5 Classe ExpTime

E' la classe dei problemi risolvibili in tempo esponenziale *($O(c^n)$, c costante, n dati in ingresso)*

4.2.6 Relazioni tra le classi

$P \subseteq PSpace$ Un algoritmo polinomiale ha accesso al piu' ad un numero polinomiale di locazioni.

$PSpace \subseteq ExpTime$

4.3 Certificati

Un certificato e' un attestato di esistenza della soluzione a un problema. Si definisce solamente per istanze accettabili.

4.3.1 Verifica

Un problema $\mathbf{\Pi}$ e' verificabile it tempo polinomiale se:

- $x \in I$ of $len = n$ ammette un certificato y of len polinomiale in n.
- Esiste un algoritmo di verifica che, applicato alle coppie $< x, y >$ permette di attestare che x e' accettabile.

4.4 Classe NP

E' la classe dei problemi risolvibili in tempo polinomiale non deterministico. Cio' significa che sono verificabili in tempo polinomiale. Se si ha una soluzione si puo' verificare la sua legittimità in tempo polinomiale; Altrimenti la si puo' individuare con una ricerca esaustiva in tempo esponenziale.

4.4.1 P e NP

Sappiamo per certo che

$$P \subset NP$$

Ma non e' stato matematicamente dimostrata la congettura

$$P \not\equiv NP$$

4.4.2 NP-Completo

I problemi NP-Completi sono i problemi piu' *"difficili"* all'interno della classe NP. Tutti i problemi NP sono riducibili in tempo polinomiale a problemi NP-Completi.

Riduzioni polinomiali Dati i problemi Π_1, Π_2 e le rispettive istanze di input I_1, I_2, si dice che Π_1 si riduce in tempo polinomiale a Π_2 se esiste una funzione $f : I_1 \to I_2$ t.c. per ogni istanza x di Π_1

$$\begin{gathered} x \text{ e' istanza accettabile di } \Pi_1 \\ \equiv \\ f(x) \text{ e' istanza accettabile di } \Pi_2 \end{gathered}$$

4.4.3 NP-Arduo

4.4.4 Teorema di Cook

Cook ha dimostrato che:

> Dato un problema Π in NP e una qualunque istanza x per Π
> Si puo' esprimere Π sotto forma di espressione booleana in forma normale, la quale restituisce *true* IIF x e' accettabile per Π

4.4.5 Gerarchia delle classi

$$(P \cup NP - Completi) \subseteq NP \subseteq PSpace \subseteq Exp \subseteq Decidibili$$

4.4.6 co-P & co-NP

Chapter 5

Esempi di algoritmi numerici

5.1 Euclide

$$a, b \in \mathbb{Z}, a \geq b, a > 0$$

$$MCD(a, b) = \begin{cases} a & \text{se } b = 0 \\ MCD(b, a \pmod b) & \text{se } b \neq 0) \end{cases}$$

Numero di passi: $O(log(a))$

5.1.1 Dimostrazione

$$a \pmod b \leq \frac{a}{2}$$

Questo perché

$$\begin{aligned} a & \\ & = \\ & qb + a \pmod b \\ (q = \frac{a}{b} \wedge a \geq b) \geq & \\ & b + a \pmod b \\ b > a \pmod b > & \\ & 2a \pmod b \end{aligned}$$

Costo di MCD: $O(n^3)$

5.2 Test di primalità (inefficiente)

Algorithm 2: Primo(N)

```
for i = 2; i < √N; i++ do √N: Max divisore se N non e' primo
    if N%i == 0 then
        return false;
return true;
```