Cifrari Storici e One-Time Pad

Cifrari Storici e One-Time Pad
- Introduzione
- Shift Cipher
- Cifrario per permutazione
- One-Time Pad
  - Perfetta sicurezza
  - Cosa succede se utilizzo due volte la stessa chiave?
  - Considerazioni finali
- Teorema di Shannon (1)
  - Teorema di Shannon (2)
  - Conclusione

Introduzione

Si considerano cifrari storici tutti i cifrari ideati prima degli anni 70. Questi sono (quasi) tutti assolutamente insicuri. Quelli che vedremo sono i seguenti:

- **Shift Cipher**;
- **Cifrario per sostituzione**.

I passi per definire un cifrario sono i seguenti:

1. Bisogna definire lo spazio delle chiavi;
2. Si associa ogni chiave ad uno specifico simbolo;
3. I messaggi sono un insieme di simboli;
4. Si cifra utilizzando uno specifico cifrario.

Shift Cipher

Definiamo l'insieme delle chiavi:

$$K = \{0, 1, 2, \dots, 25\}$$

Ad ogni lettera viene associata una delle chiavi:

$$A = 0,\ B = 1,\ \dots$$

Per cifrare bisogna "spostare" ogni lettera secondo quanto indicato dalla chiave.

> **Curiosità:**
>
> Se la chiave è k = 3 allora il cifrario prende il nome di *cifrario di cesare*.

Esempio:

k = 3, m = dado

dado => 03 00 03 14

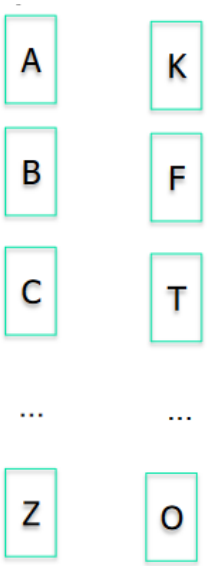
Applicando la chiave si ottiene:

Questo cifrario è facile da rompere perché basta provare in maniera random degli shift per trovare il messaggio originale. Quindi osserviamo che il numero delle chiavi deve essere sufficientemente grande per evitare di forzare la ricerca della chiave corretta.

Cifrario per permutazione

In questo cifrario K è l'insieme di tutte le possibili permutazioni dei simboli 0, 1, ..., 25.
Ogni parola viene cifrata sostituendo al simbolo iniziale, il simbolo indicato dalla chiave.

Supponiamo di avere due mazzi di carte: il primo lo disponiamo in maniera ordinata; il secondo viene mescolato e ogni carta posta accanto all'altra. Sotto viene riportato un esempio.



Dunque alla A “sostituiremo” la K, alla B la F, …

Esempio:
k associa: A → K, C → T, S→ D

CASA => TKDK

Analisi:

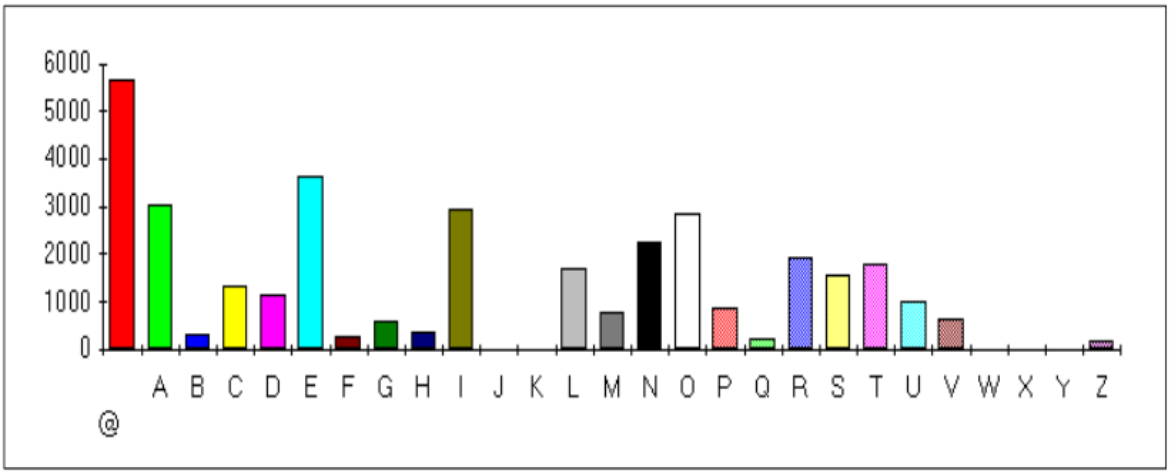
In questo caso il numero di chiavi possibili è molto alto.
Supponiamo di avere un alfabeto di 256 caratteri, allora il numero di chiavi è pari a 256!

$$256! > 128^{128} = 2^{7*128} = 2^{896} > 10^{224}$$

Supponiamo di avere un computer capace di provare 1000 miliardi di chiavi al secondo, ovvero 10^12 chiavi al secondo. In totale ci vorrebbero
$10^{212}\ secondi > 10^{200}\ giorni > 10^{195}\ anni > 10^{185}$ miliardi di anni.

Questo però non ci permette di affermare che il cifrario è sicuro, perché nei linguaggi naturali ci sono dei pattern che si ripetono, ovvero analizzando un testo e calcolando la frequenza delle varie lettere, si può notare che le vocali E, A, O sono le lettere più frequenti del dizionario italiano. A queste seguono L, N, R, S, T. Tra i bi-grammi più frequenti troviamo Q - U (seguiti da una vocale). La lettere H è spesso preceduta dalla lettera C o G per formare i trigrammi CHE, CHI, GHE, GHI. Le lettere straniere invece sono quasi assenti.

Esempio:



Questa statistica però non aiuta per testi piccoli e in quel caso sembra che il cifrario per sostituzione sia sicuro. In realtà è possibile trovare dei contro esempi come nel caso delle votazioni studentesche (esempio svolto in aula).

One-Time Pad

Definiamo l'insieme delle chiavi nel seguente modo:

$$K = \{0,1\}^l$$

Consideriamo solo stringhe che hanno esattamente *l* bit e quindi una chiave con la stessa dimensione. Quindi si ha $m = k = c$, dove c è il cripto-testo.

Una chiave k è costituita da elementi presi a caso con distribuzione uniforme da K.

$$k \xleftarrow{R} K$$

La chiave generata deve essere utilizzata UNA sola volta.

La **Secure Encryption (SE)** è definita dalla terna *(KeyGen, Enc, Dec)* e nel caso del One-Time Pad si ha:

- Enc(k, m) che permette di produrre il cripto-testo utilizzando l'operatore XOR: $c = m \oplus k$
- Dec(k, c) che permette di ottenere il testo in chiaro utilizzando l'operazione di XOR: $m = c \oplus k$

Si usa l'operazione di XOR perché:

- Facilmente invertibile;
- Locale;
- Efficiente.

Perfetta sicurezza

Supponiamo di avere un cifrario simmetrico, come il One-Time Pad.

$$\forall m_1, m_2 \in M \, e \, \forall c \in C$$

si ha:

$$P[Enc(k, m_1) = c] = P[Enc(k, m_2) = c]$$

Questo ci dice che il cripto-testo non contiene nessuna informazione sul messaggio.

Esempio 1:

Dimostrazione che il cifrario per sostituzione non soddisfa la definizione di perfetta sicurezza. Consideriamo due parole di 4 lettere:

$m_1 =$ CIAO e $m_2 =$ CASA

$c =$ TBKW

Consideriamo m_1:

C = T

I = B

A = K

O = W

$$P[Enc(k, m_1) = c] = \frac{22!}{26!} = \frac{1}{26 * 25 * 24 * 23}$$

Consideriamo m_2:

C = T

A = B

S = K

A = W

Le due A non possono essere sostituite da due caratteri diversi, quindi si ha:

$$P[Enc(k, m_2) = c] = \frac{0}{26!} = 0$$

Le due probabilità sono diverse quindi il cifrario non è sicuro.

Esempio 2:

Utilizziamo in questo secondo esempio il One-Time Pad e dimostriamo la perfetta sicurezza.

$$P[Enc(k, m_1) = c] = P[k \, \oplus \, m_1 = c] = \frac{1}{2^l}$$

k è l'incognita e si calcola nel seguente modo: $k = m_1 \, \oplus \, c$ dalla quale deriva proprio il risultato della probabilità dato che $M = C = \{0,1\}^l$

La stessa considerazione può essere fatta per il calcolo della seconda probabilità:

$$P[Enc(k, m_2) = c] = P[k \oplus m_2 = c] = \frac{1}{2^l}$$

Quindi il One-Time Pad è un cifrario che rispetta la perfetta sicurezza, anche quando la chiave contiene tutti zeri.

Esempio 3:

Consideriamo un ipotetico One-Time Pad +, per cui si ha:

$$K = \{0,1\}^l - \{0^l\}$$

Dimostriamo che non è perfettamente sicuro.

Supponiamo $l = 5$, $m_1 = 10010$, $m_2 = 10100$ e $c = 10100$.

Consideriamo m_1:

$$P[Enc(k, m_1) = c] = P[k \oplus m_1 = c] = \frac{1}{2^l - 1}$$

Consideriamo m_2:

$$P[Enc(k, m_2) = c] = P[k \oplus m_2 = c] = 0$$

Questa nuova versione non rispetta la perfetta sicurezza.

Cosa succede se utilizzo due volte la stessa chiave?

Supponiamo di prendere due messaggi m_1 e m_2 e di volerli cifrare con la stessa chiave k. Così facendo otteniamo due cripto-testi:

$$m_1 \oplus k = c_1$$
$$m_2 \oplus k = c_2$$

L'avversario è in grado di vedere entrambi i cripto-testi, perché abbiamo supposto che il canale non è sicuro. L'avversario può quindi combinare:

$$c_1 \oplus c_2 = (m_1 \oplus k) \oplus (m_2 \oplus k) = m_1 \oplus m_2$$

Dato che i messaggi sono noti, posso estrarre i messaggi nel seguente modo:

$$c_1 \oplus c_2 \oplus m_1 = m_2$$
$$c_1 \oplus c_2 \oplus m_2 = m_1$$

Dimostrazione

Supponiamo di concatenare i due messaggi:

$$m_1 || m_2$$

Ovvero se $m_1 = 10$ e $m_2 = 11$ ottengo che $m_1 || m_2 = 1011$

Quindi si ha che:

$$(m1||m2) \oplus (k||k) = c$$

Si sa che lo spazio dei messaggi è $M = \{0,1\}^{2l}$ e quello delle chiavi è $K = \{0,1\}^{2l}$.

Troviamo un controesempio per cui non valga la definizione di perfetta sicurezza.

Consideriamo:

- $m_1 = X||Y$ con $X, Y \in \{0,1\}^l$ e $X \neq Y$
- $m_2 = X||X$

Allora:

- $C = Z||Z$ con $Z \in \{0,1\}^l$

Consideriamo m_1:

$$P[Enc(k, m_1) = c] = P[(k||k) \oplus (X||Y) = Z||Z] \rightarrow k||k = (X \oplus Z)||(Y \oplus Z) = 0$$

Questo perché è impossibile con due stringhe diverse lo stesso cripto-testo.

Consideriamo m_2:

$$P[Enc(k, m_2) = c] = P[(k||k) \oplus (X||X) = Z||Z] \neq 0$$

In questo caso esiste una probabilità non nulla per cui uno stesso messaggio dia lo stesso cripto-testo.

E' stato dimostrato che utilizzare la stessa chiave due volte va contro la definizione di perfetta sicurezza.

Considerazioni finali

One-Time Pad sembra efficiente, ma il cambiamento della chiave lo rende inutilizzabile, perché per mandare 2TB di messaggi bisognerebbe avere 2TB di chiavi, questa considerazione deriva dal Teorema di Shannon.

Teorema di Shannon (1)

Sia (KeyGen, Enc, Dec) uno schema perfettamente sicuro (M spazio dei messaggi, K spazio delle chiavi) allora:

$$|K| \geq |M|$$

Il teorema ci dice che volendo ottenere perfetta sicurezza bisogna avere un numero di chiavi maggiore o uguale al numero di messaggi da inviare.

Dimostrazione

Supponiamo per assurdo che $|K| < |M|$ e $m_1, m_2 \in M$. Inoltre, supponiamo che $c \in C$ sia valido.

Allora costruisco:

$$M(c) = \{m : Dec(k, c) = m \text{ al variare di } k\}$$

Ogni volta che effettuo una decifrazione posso ottenere due risultati:

- **valido**, che viene inserito in $M(c)$;
- **non valido**, che non viene inserito in $M(c)$.

Dato che esiste almeno un $c \in C$ che sia valido, allora $M(c) \neq \Phi$ ovvero
$1 \leq |M(c)| \leq |K| \leq |M|$

Costruiamo il controesempio.

Supponiamo che:

$$\exists \tilde{m} \in M \land \tilde{m} \notin M(c)$$

Consideriamo $m_1 \in M$:

$$P[Enc(k, m_1) = c] \neq 0$$

Questo perché il messaggio appartiene all'insieme $M(c)$.

Consideriamo $m_2 = \tilde{m}$:

$$P[Enc(k, m_2) = c] = 0$$

Questo perché abbiamo assunto per assurdo $|K| < |M|$ e quindi deve valere che:

$$|K| \geq |M|$$

Shannon fece molto di più e definì una Condizione Necessaria e Sufficiente.

Teorema di Shannon (2)

$SE = (KeyGen, Enc, Dec)$, $|M| = |K| = |C|$, SE offre perfetta sicurezza se e soltanto se:

1. $\forall k \in K, k$ è scelta con probabilità $1/|K|$;
2. $\forall m \in M, \forall c \in C, \exists k \in K$, tale che $Enc(k, m) = c$.

Conclusione

Questi teoremi ci dicono che la **Perfetta Sicurezza** è difficile e infattibile da un punto di vista pratico. Quindi bisogna creare una crittografia alternativa, che fornisce sicurezza da un punto di vista pratico, tale che gli avversari non riescono a rompere il sistema a causa dei limiti computazionali.