

CIFRARI: INTRODUZIONE

Il **CIFRARIO** è un sistema che, mediante opposti algoritmi, permette di:

- CIFRARE
- DECIFRARE
- GENERARE & GESTIRE CHIAVI CRITTOGRAFICHE

} ovvero, permette di garantire la **CONFIDENZIALITÀ**

I CIFRARI si possono classificare in 2 grandi classi:

- **APERTI** → gli ALGORITMI sono PUBBLICI
- **CHIUSI** → gli ALGORITMI sono SEGRETI

■ Pro & Contro ■

Il mondo commerciale utilizza i **CIFRARI APERTI** poiché sono considerati **più sicuri**.

La ragione risiede nel fatto che nella CYBERSECURITY è molto facile sbagliare: si beneficia dell'aiuto di tutti gli studiosi ed esperti del settore per trovare fallo.

Il mondo governativo/militare utilizza i **CIFRARI CHIUSI** e ciò **non** beneficia dei vantaggi sopra descritti.

Questo "concetto" prende il nome di **PRINCIPIO DI KERKHOFFS**:

« Metodi e Algoritmi segreti prima o poi verranno conosciuti dall'avversario, il segreto »
dove essere concentrato nelle chiavi

Oggi però, si sente più spesso parlare di **SECURITY THROUGH OBSCURITY**:

« sicurezza attraverso SUPPERFUGI »

che tendono a mascondere le cose in modo disordinato e non sicuro... il che rende il tutto estremamente deleterio e non sicuro!

(es: soldi sotto un materasso)

I CIFRARI possono essere di 2 tipi:

- **CIFRARI SIMMETRICI** → im cui le CHIAVI per (DE)CIFRARE sono CONDIVISE cioè sono le stesse. Si parla analogamente di **CIFRARI CONVENZIONALI**.

↓
A CHIAVI CONDIVISE

Nei CIFRARI SIMMETRICI, che tratteremo in questa prima parte del corso, sia il mittente **A** che il destinatario **B** utilizzano una stessa CHIAVE CONDIVISA $K_{A,B}$:



Fase 1: **CIFRATURA (ENCRYPTION)** con K_{AB} del messaggio

Fase 2: **DECIFRATURA (DECRYPTION)** con K_{AB} del messaggio

- **CIFRARI ASIMMETRICI** → im cui le CHIAVI per (DE)CIFRARE sono DIVERSE

Parleremo di :

- **TESTO IN CHIARO (PLAINTEXT o CLEARTEXT)**: PRIMA della CIFRATURA
- **TESTO CIFRATO (CIPHERTEXT)**: DOPO la CIFRATURA

CIFRARI SIMMETRICI

In questa prima parte parliamo dei CIFRARI SIMMETRICI.

Parleremo di:

- **CIFRARI A
"PERMUTAZIONE"** → im cui gruppi di caratteri vengono spostati di posizione nel testo.
 - **CIFRARI A
"SOSTITUZIONE"** → im cui un gruppo di caratteri viene sostituito con un altro gruppo di caratteri
- **MONOALFABETICI
A N-LETTERE** → Im cui la SOSTITUZIONE è la STESSA:
ogni N-upla di lettere del testo im chiaro viene sostituita sempre dalla stessa sostituzione
- **POLIALFABETICI
A N-LETTERE** → Im cui la SOSTITUZIONE è DIVERSA:
ogni N-upla di lettere del testo im chiaro viene sostituita im modo diverso

PREMESSA: Inizialmente applichiamo questi concetti su CARATTERI (CHARACTER ORIENTED) ovvero in ottica pre-informatica ... ma, come vedremo, li si può applicare ai BIT (BIT ORIENTED)

CIFRARI A SOSTITUZIONE: MONOALFABETICI

Per cifrare un testo, la cosa più semplice a cui si può pensare è la SOSTITUZIONE di un carattere. Ad es: $a \rightarrow q$, $b \rightarrow z$, $c \rightarrow f$, ...

PROBLEMA 1: È necessario ricordarsi la permutazione che, nel caso dell'italiano è $21!$ (~ 51 Miliardi di Chiavi) Il che, non è molto semplice ...

SOLUZIONE

Nel passato (e nel corso degli anni), sono utilizzate delle approssimazioni di questo metodo, che permettono di avere una chiave più corta e facile da ricordare. È il caso, ad esempio, del **CIFRARIO DI CESARE**:

CIFRARIO DI CESARE

- **CHIAVE:** ognero chiave K tra $0 \xrightarrow{=} a$ e $20 \xrightarrow{=} z$ ← **VINCOLATO** molto
- **CIFRA:** sostituisce la lettera x con la lettera $(x + K) \% 21$
- **DECIFRA:** sostituisce la lettera x con la lettera $(x - K) \% 21$

Ad esempio con $K = 3$:

buongiorno

$$b = 1 \Rightarrow (1 + 3) \% 21 = 4 = e$$

... 

diventa

earqlnruqr

$$e = 4 \Rightarrow (4 - 3) \% 21 = 1 = b$$

... 

Il **CIFRARIO DI CESARE** è un esempio molto utile per fare alcune osservazioni ed approfondire alcuni concetti...

In generale CIFRARI A SOSTITUZIONE MONOALFABETICA sono deboli poiche' soggetti a:

PROBLEMA 1 : CRYPTO ANALISI STATISTICA

NEW
PROBLEM

- Preseenza di regolarita' nel testo.
- PROBLEMA 1.1: + il CIPHERTEXT e' lungo,
+ e' facile decifrarlo.

Frequenza delle lettere in Inglese

| | | | |
|---|-------|---|------|
| a | 7.25 | n | 7.75 |
| b | 1.25 | o | 7.5 |
| c | 3.5 | p | 2.75 |
| d | 4.25 | q | 0.5 |
| e | 12.75 | r | 8.5 |
| f | 3.0 | s | 6.0 |
| g | 2.0 | t | 9.25 |
| h | 3.5 | u | 3.0 |
| i | 7.75 | v | 1.5 |
| j | 0.25 | w | 1.5 |
| k | 0.5 | x | 0.5 |
| l | 3.75 | y | 2.25 |
| m | 1.25 | z | 0.25 |

ATTACCO:

- Confronto la frequenza delle lettere nel CIPHERTEXT con la frequenza delle lettere in inglese.

- Ipotizzo e verifico possibili corrispondenze tra lettere del CIPHERTEXT con quelle del PLAINTEXT

Itero

L'idea e' ridurre il numero di lettere non ancora decifrati, ovvero lo spazio di ricerca, su cui applicare un:

PROBLEMA 2 : ATTACCO DI FORZA BRUTA (BRUTE FORCE)

- E' un tipo di attacco relativo alla capacita' di calcolo di un elaboratore nel provare tutte le possibili alternative (non solo in ambito di chiavi)

Di seguito e' riportato un esempio pratico su PYTHON che ci consente di analizzare tutti questi concetti.

ESEMPIO PYTHON

Il mostro scopo e' decifrare la seguente frase:

```
str2 ="uzqso vuohx mopvg pozpe vsgzw szopf pesxu d  
bmet sxaiz \  
vueph zhndz shzow sfpap pdtsv pquzw ymxuz uhsx \  
epyepopdzszufpo mb zwp fupz hmdj ud tmohmq"
```

import pandas as pd
import numpy as np

per le matrici
per il calcolo numerico

- Importo alcune librerie standard

```
str2 ="uzqso vuohx mopvg pozpe vsgzw szopf pesxu d  
bmet sxaiz \  
vueph zhndz shzow sfpap pdtsv pquzw ymxuz uhsx \  
epyepopdzszufpo mb zwp fupz hmdj ud tmohmq"
```

```
ea = "abcdefghijklmnopqrstuvwxyz"  
eac = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"  
l = np.arange(0,25)  
letters = [letter for letter in ea]  
counts = [str2.count(letter) for letter in letters]  
la = pd.DataFrame(counts, index=letters)
```

- Definisco Alfabeto in mim & MAIUSCOLO
- Numeri da 0 a 25
- Elenco le lettere
- Conta la frequenza di ogni lettera
- Creo una tabella con tali dati... e la ordino

```
result = la.sort_values([0], inplace=True, ascending=False)
```

- Stampo tabella

la

| | | | | |
|----|----|---|---|---|
| 0 | 16 | 7 | 3 | 1 |
| p | z | d | t | n |
| 14 | 10 | e | y | r |
| 10 | 10 | x | a | l |
| 9 | 7 | v | b | k |
| 7 | | f | g | c |
| | | w | j | |

```

freql = "etrinoasd1hcFupygvwBmqXkzj"
freqlC = "ETRINOASDLHCFUPYGVWBMQXKZJ"
freqlist = [letter for letter in freqlC]
la[1] = freqlist

```

- Creo una tabella com frequenza teorica delle lettere in inglese

- Affiamco le due tab.

| | | |
|---|----|---|
| | 0 | 1 |
| p | 16 | E |
| z | 14 | T |
| u | 10 | R |
| s | 10 | I |
| o | 9 | N |
| h | 7 | O |

| | | |
|---|---|---|
| m | 7 | A |
| d | 6 | S |
| e | 6 | D |
| x | 5 | L |
| v | 5 | H |
| f | 4 | C |
| w | 4 | F |

| | | |
|---|---|---|
| q | 3 | U |
| t | 3 | P |
| y | 2 | Y |
| a | 2 | G |
| b | 2 | V |
| g | 2 | W |
| j | 1 | B |

In questo momento ho davanti agli occhi una possibile sostituzione che potrebbe andar bene.

Di seguito metto in atto delle sostituzioni...

per facilità di
visione im
maiuscolo

```
str3 = str2.replace("p", la.loc["p"][1])
```

```
str4 = str3.replace("z", la.loc["z"][1])
```

Comincia ad emergere im maiuscolo quello che potrebbe essere il TESTO IN CHIARO. Vado avanti così fino a quando non mi accorgo che sto facendo qualche errore:

```
str5 = str4.replace("u", la.loc["u"][1])
```

str5

RTqso vRohx moEvg EoTEe vsgTw sToEf EesxR dbmet
sxaiT vReEh ThndT shTow sfEaE Edtsv EqRTw ymxRT R
hsx eEyeEoEdTsTRfEo mb TwE fRET hmdj Rd tmohmq'

ERRORE!

messima frase
im inglese imizia
com "RT" ...

Provo a fare una sostituzione diversa: sostituisco la "u" con la "I"

```
str5 = str4.replace("u", "I")
```

la prossima
im ordine
di frequenza

Fatto ciò si può passare anche a sostituire coppie di lettere frequentemente usate. Ad esempio un digrafo molto usato è il "th":

```
str5
```

```
'ITqso vIohx moEvg EoTEe vsgTw sToEf EesxI dbmet  
sxaiT vIeEh ThndT shTow sfEaE Edtsv EqITw ymxIT I  
hsx eEyeEoEdTsTIfEo mb TwE fIET hmdj Id tmohmq'
```

Osservando un po' il testo ed usando un po' di astuzia, è possibile ipotizzare che la "w" sia "H"

```
str6 = str5.replace("w", "H")
```

E continuo così ...

Questo al fine di ridurre il numero di lettere che ancora non ho decifrato, su cui effettuare un ATTACCO DI FORZA BRUTA

ad esempio 16!
anziche' 26!

Esercizio Completo (moodle): *StatisticalAnalysisExample.py*

PYTHON è perfetto a questo scopo poiché permette di generare in maniera automatica ed esaustiva un ATTACCO DI FORZA BRUTA.

NATURAL

LANGUAGE - NLP :

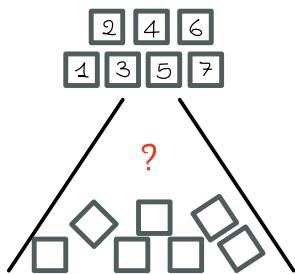
PROCESSING

L'elaborazione del linguaggio naturale è una sottobranca di linguistica, intelligenza artificiale ecc.

Grazie all'utilizzo di un **NLP** è in grado di capire se le parole generate sono effettivamente dei vocaboli o meno.

Volendo riassumere: il **CIFRARIO DI CESARE** e' molto debole poiche' soggetto ad **ATTACCHI DI FORZA BRUTA & STATISTICI**.

Dobbiamo quindi "spezzare" qualsiasi regolarita' nel testo. Si deve cioe' garantire il cosi detto **EFFETTO VALANGA**:



I dati sono ordinati sulla vetta della montagna.

La valanga porta tutto a valle ... mescolando il tutto in modo imprevedibile (e quindi rovinando la **STATISTICA**)

Approfondiremo questo concetto con i **CIFRARI MODERNI**... Ora invece, ci concentriamo sul come migliorare il mostro **CIFRARIO A SOSTITUZIONE MONOALFABETICA**:

CIFRARIO DI CESARE → 21 CHIAVI

SOSTITUZIONE A 1-LETTERA → 21! CHIAVI 😕

Una prima idea potrebbe essere sostituire coppie di lettere anziche' singole lettere:

aa → ...

az

ba

bz

za

zz

SOSTITUZIONE A 2-LETTERE → (21^2) ! CHIAVI



Questa semplice modifica rende

😊 IMPOSSIBILE l'ATTACCO DI FORZA BRUTA

MA e ancora possibile

😕 l'ANALISI STATISTICA

Un esempio di **CIFRARIO A SOSTITUZIONE A 2-LETTERE** e' il cifrario di Playfair ... che pera' non approfondiremo nel corso.

Non possiamo essere soddisfatti di questo piccolo miglioramento! Dobbiamo osservare pera' che fino ad ora, la sostituzione e' sempre stata la stessa, per ogni riconvenzione.

L'**ANALISI STATISTICA** e' resa molto piu' difficile con un **CIFRARIO POLIALFABETICO**, ove una lettera viene sostituita ogni volta in modo diverso a seconda della posizione del testo.

CIFRARI A SOSTITUZIONE: POLIALFABETICI

CIFRARIO DI VIGENERE'

- CHIAVE: numero chiave $K = K_0, K_1 \dots K_{n-1}$, dove ogni sotto-chiave K_i consiste in un numero tra 0 e 20

- CIFRA: sostituisce la lettera T_j con la lettera:

$$(T_j + K_{(j \% n)}) \% 21$$

► DECIFRA: segno —

Detto in altre parole:

- prendo la lettera T_j
- applico il CIFRARIO DI CESARE con chiave $K_{(j \% n)}$

è molto probabile
che la chiave sia
più corta del testo

Ad esempio:

$$n = 5 \curvearrowright$$

$$K = K_0 K_1 \dots K_4 = 10, 3, 1, 20, 0$$

Testo = b u o n a g i o r n a t a
10 3 1 20 0 10 3 1 20 0 10 3 1
→ n a p m a s n p q n m z b

Quando la
chiave finisce?
La ripeto!

In questo specifico esempio ho 21^5 possibili chiavi.
Numero che può essere reso arbitrariamente grande.

Contro: C'è ancora dell'ANALISI STATISTICA poiché si usa la stessa chiave su lettere che distanziemo n posizioni nel testo.

Quindi, anche questo CIFRARIO DI VIGENERE' è imperfetto ... a meno che la chiave non abbia la stessa lunghezza del testo.
Si parla in questo caso di **ONE TIME PAD** e non c'è alcuna ANALISI STATISTICA possibile:

TOTALMENTE SICURO MA NON UTILE a livello pratico

Tanto vale scambiarsi in modo sicuro il messaggio,
invece della chiave!

OR-ESCLUSIVO (XOR)

Come abbiamo accennato all'inizio, possiamo dimenticarci delle sostituzioni alfabetiche.

Per passare da caratteri a bit, dobbiamo introdurre l'operatore **OR-ESCLUSIVO** \oplus :

- Trasforma un bit X in qualsiasi altro bit Y , utilizzando una opportuna chiave K
- Analogamente, trasforma 8 bit X in qualsiasi altra sequenza di 8 bit Y , utilizzando una opportuna chiave K di 8 bit

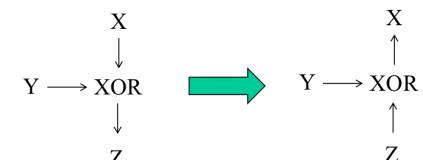
Noi lo utilizzeremo moltissimo in questo corso (così come è utilizzato molto in ambito crittografico/di sicurezza informatica), poiché grazie alla sua struttura:

| A | B | C |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

- **INVERTIBILE** Per capire tale concetto basti pensare ad A come il TESTO IN CHIARO e B come CHIAVE
- **NO RIORTO**
- $A \oplus B = C \Rightarrow A = B \oplus C$

Dimostrazione:

Se $X \text{ xor } Y = Z$ allora $Z \text{ xor } Y = X$
Infatti $X \text{ xor } Y \text{ xor } Y = Z \text{ xor } Y$, quindi
 $X \text{ xor } 0 = Z \text{ xor } Y$, e pertanto $X = Z \text{ xor } Y$



CIFRARIO DI VERNAM

La versione bit-oriented del **CIFRARIO DI VIGENERE**:

- **CHIAVE**: genera chiave binaria $K = K_0, K_1 \dots K_{n-1}$
- **CIFRA DECIFRA**: sostituisce il bit T_j con il bit $(T_j \oplus K_{(j \% n)})$

Ad esempio:

$n = 5$

$$K = K_0 K_1 \dots K_4 = 0, 0, 1, 0, 1$$

Testo = 1 1 0 0 1 0 1 0 1 0 0

0 0 1 0 1 0 0 1 0 1 0

\rightarrow 1 1 1 0 0 0 1 1 1 1 0

Detto in altre parole:

- prendo il bit T_j
- applico lo **XOR** con chiave $K_{(j \% n)}$

Valgono le stesse e identiche osservazioni viste im **VIGENERE**

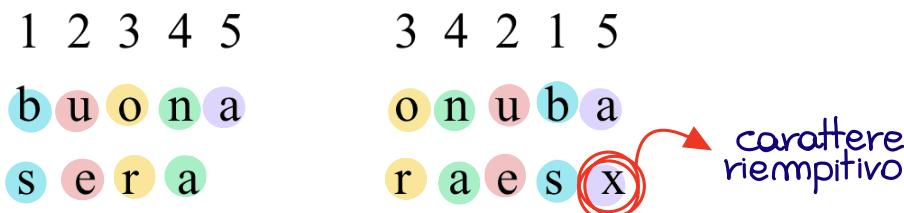
CIFRARI A PERMUTAZIONE

In questa configurazione le lettere vengono scambiate di posizione, non sostituite.

- CHIAVE: genero una permutazione segreta di N elementi
- CIFRA:
 - sistemo il testo su N colonne
 - scambio le lettere invertendo le colonne secondo una permutazione segreta di N elementi

Ad esempio:

Buona sera diventa onubaraesx



$$K = 3 \ 4 \ 2 \ 1 \ 5$$

Contro: C'è ancora dell'**ANALISI STATISTICA** poiché è possibile elaborare successivi raffinamenti di ipotesi di permutazione basandosi sulla frequenza di digrafi e trigrafi, o sulla presenza di testo fisso o probabile

CONCLUSIONI

Abbiamo visto:

- CIFRARI A SOSTITUZIONE MONOALFABETICA A n -LETTERE
- CIFRARI A SOSTITUZIONE POLIALFABETICA A n -LETTERE
- CIFRARI A PERMUTAZIONE

NON possiamo ritenerci soddisfatti al 100%

Tutta questa panoramica su cifrari pressoché inutili, ci serve per capire i **CIFRARI MODERNI**.

La strategia vincente infatti è combinare tutti questi concetti: PERMUTAZIONI, SOSTITUZIONI e ROUD

→ fasi