INTRODUZIONE ALLA CRITTOGRAFIA

BY NANNI BASSETTI

WWW.NANNIBASSETTI.COM

https://github.com/nannib/crypto

CHI SONO

Mi occupo di digital forensics dal 2005/2006, project manager di CAINE (distro usata in tutto il mondo)

Fondatore di CFI (Computer Forensics Italy)

Scrittore di parecchi articoli scientifici e software free/open source per la D.F.

Membro fondatore di ONIF (Osservatorio Nazionale Informatica Forense).

Coinvolto in casi di rilevanza nazionale come:

* Consulente tecnico informatico di parte civile nel caso del transessuale "Brenda" (caso Brenda-Marrazzo).

* Consulente tecnico informatico di parte civile nel caso della scomparsa di Roberto Straccia

* Consulente tecnico informatico di parte civile nel caso della scomparsa della piccola Angela Celentano

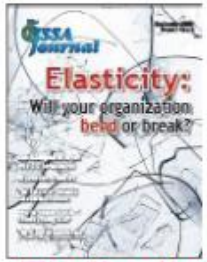
* Consulente tecnico di parte nel caso Bossetti/Yara Gambirasio.

CHI SONO

2005/2006

primi vagiti online sulla computer forensics

2007



Tanti articoli dal 2007 ad oggi

2008

Creazione di CFI e pubblicazione di SFDumper

2009 - 2014

Classificato nella DC3 Challenge

2009

Scripts4CF

Alcuni miei scripts per la forensics

2009

Sviluppo CAINE Gnu/Linux

2013

Nel comitato di redazione di S&G

2015

Segretario e membro fondatore ONIF

CHI SONO

CASI “MEDIATICI”

E TANTA FORMAZIONE

DEFINIZIONE

"*la crittografia è la disciplina che studia la trasformazione di dati allo scopo di nascondere il loro contenuto semantico, impedire il loro utilizzo non autorizzato, o impedire qualsiasi loro modifica non rilevabile*" - Bruce Schneier

La crittografia (dall'unione di due parole greche: κρυπτός (kryptós) che significa "nascosto", e γραφία (graphía) che significa "scrittura") è la branca della crittologia che tratta delle "scritture nascoste", ovvero dei metodi per rendere un messaggio "offuscato" in modo da non essere comprensibile/intelligibile a persone non autorizzate a leggerlo. - https://it.wikipedia.org/wiki/Crittografia

LA STORIA

Da sempre si sono ideati sistemi per trasmettere le informazioni senza che qualcun'altro potesse decifrarle:

ATABASH - Ebrei
SCITALA - Spartani
CESARE - Romani
DISCO CIFRANTE - L.B. Alberti - 1466 D.C.
VIGENERE - Vigenère - 1586
VERNAM - G. Vernam - 1918
ENIGMA - WWII - 1920-1945
DES - IBM/NSA 1977
RSA - Rivest Shamir Adleman - 1978/1982
IDEA - Xuejja Lai e James L. Massey - 1991
AES - Joan Daemen e Vincent Rijmen - 2001
CRITTOGRAFIA QUANTISTICA E CURVE ELLITTICHE....

ATABASH E CESARE

CIFRARIO DI ATABASH

A	B	C	D	E	F	G	H	I	J	K	L	M
Z	Y	X	W	V	U	T	S	R	Q	P	O	N

cifrario monoalfabetico

CIFRARIO DI CESARE (ROT3)

ABCDEFGHIJKLMNOPQRSTUVWXYZ
DEFGHIJKLMNOPQRSTUVWXYZABC
transforms "HELLO" to "KHOOR"

ALBERTI

DISCO CIFRANTE DI ALBERTI

polialfabetico

Si sceglie dal disco interno una lettera
e la si mette in corrispondenza della A del disco esterno
Se sceliamo la g ecco che appare l'alfabeto alternativo

disco stabile: ABCDEFGILMNOPQRSTVXZ1234

disco mobile: gklnprtuz&xysomqihfdbace

https://it.wikipedia.org/wiki/Disco_cifrante

VIGENÈRE

Polialfabetico

un cifrario del 1586 ossia il famoso cifrario di Vigenère (https://it.wikipedia.org/wiki/Cifrario_di_Vigen%C3%A8re), che utilizza una chiave alfabetica per creare il testo criptato, semplicemente spostando ogni lettera del testo in chiaro del numero di posti relativo alla lettera corrispondente della chiave di cifratura, che si ripete fino alla fine del testo in chiaro.

Esempio:
Testo in chiaro: ciaoailettori Chiave: pippo

Si deve posizionare la chiave sotto il testo, ripetendola fine alla fine dello stesso:

VIGENÈRE

C	I	A	O	A	I	L	E	T	T	O	R	I
P	I	P	P	O	P	I	P	P	O	P	I	P

Per cifrare il messaggio basta aggiungere al numero di posizione della lettera del testo in chiaro il numero di posizione della lettera della chiave e se supera il 26 (il numero di lettere dell'alfabeto) basta sottrarre 26 e ricavare il numero della lettera che finirà nel testo cifrato, infine si consideri che il conteggio delle posizioni inizia da 0 (zero) ossia la lettera A occupa la posizione 0, la B la posizione 1 e così via.

VIGENÈRE

ALFABETO:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

TESTO IN CHIARO	**A**	**B**	**C**
CHIAVE	**M**	**I**	**K**
TESTO CIFRATO	**M**	**J**	**M**

A=0 B=1 C=2

+

M=12 I=8 K=10

12 9 12

Ossia: MJM

Per decriptare:

TESTO IN CIFRATO	**M**	**J**	**M**
CHIAVE	**M**	**I**	**K**
TESTO IN CHIARO	**A**	**B**	**C**

M=12 J=9 M=12

-

M=12 I=8 K=10

0 1 2

Ossia: ABC

VIGENÈRE

Se da un’addizione (cifratura) fosse venuto un numero maggiore di 26, basta effettuare la sottrazione di 26 al quel numero per avere la lettera corrispondente, es. 32 - 26 = 6 ossia la lettera G.

Se da una sottrazione (decifratura) risultasse un numero negativo, basterà aggiungere 26 per riportralo nel range dei valori dell’alfabeto, es. V=21 - Z=25 = -4, quindi -4 + 26 = 22 che corrisponde alla lettera W.

Se la posizione della lettera di cifratura o di decifratura è 26 si “riavvolge” il nastro, nel senso che il 26-esimo elemento non esiste quindi diventa 0 ossia si riparte dalla lettera A.

VIGENÈRE

Si può usare ovviamente una funzione matematica per la criptazione e decriptazione; in entrambe useremo sempre le stesse lettere:

Numero prima lettera del cifrario (A) = 0

Numero ultima lettera del cifrario (Z) = 25

L = Lunghezza del cifrario = Numero elementi dell'insieme (26)

a = Numero della lettera della parola in Chiaro (0-25)

b = Numero della lettera della parola Chiave (0-25)

c = Numero della lettera della parola Criptata (0-25)

Per criptare: c= a + b

Per decriptare: n = c - b + 26

r [parte intera del numero] **= n / L**

F(x) = n - (L * r) = Numero della lettera della parola in Chiaro/Criptata (0-25)

La funzione si basa semplicemente sulla somma/sottrazione dei numeri delle lettere e dividere per la lunghezza del cifrario per ottenere il numero della lettera desiderata. Per ottenere SEMPRE un numero **n** positivo anche per la decriptazione, in quanto una sottrazione, basta ricorrere al semplice aritificio di aggiungere 26, in quanto verrà poi eliminato grazie ad r. (**https://it.wikipedia.org/wiki/Cifrario_di_Vigen%C3%A8re**)

VIGENÈRE

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

TAVOLA DI VIGENERE

Testo: guerrasia

chiave: CIAO

TC: iceftiswc

VEDERE CODICE:
VIGENERE_NB.py
https://github.com/nannib/crypto

HASH E SALT

Un codice hash è un codice a lunghezza fissa, a seconda dell'algoritmo utilizzato, generato da una funzione matematica non invertibile, nel senso che dal codice non si può ricavare l'oggetto al quale si riferisce, ossia se applichiamo la funzione di hash ad un file di 200Gb o alla parola "pippo", avremo sempre una sequenza di numeri e lettere di lunghezza fissa, che non contiene il file o il testo, insomma come se fosse la targa di un'automobile o l'impronta digitale di un essere umano.

Ma un esempio vale sempre mille parole:

hash("pippo")= ABC123

hash("l'estate sta finendo")= DFF872

Da qui possiamo capire che dal codice DFF872 non possiamo ricavare l'intera frase "l'estate sta finendo", perché non è una cifratura o una codifica in un altro alfabeto ma è un codice a lunghezza fissa, che identifica univocamente un qualcosa, basta cambiare anche un solo bit ed il codice cambia, per esempio:

hash("l'estate sta finend")=ADB634

HASH E SALT

Quindi come funzionano i database di password dei vari servizi online? Quando scegliamo la password, es. “pippo”, nel database viene memorizzato l’hash di “pippo”, ossia, per riprendere l’esempio precedente, “ABC123”, quindi ogni volta che inseriamo la password per accedere ad un servizio, l’applicazione calcola l’hash della password e lo confronta con quello presente nel database, se i due hash coincidono allora la nostra password è corretta.

Ma se i database contengono l’hash delle password e dall’hash è impossibile risalire alla password in chiaro, allora non rischiamo nulla?

HASH E SALT

Falso, rischiamo! Perché ci sono vari metodi per ricavare la password in chiaro, specialmente se molto semplice (parole di senso compiuto, sequenze famose, ecc.), ci sono le rainbow tables, i dizionari, i servizi online, cercate con Google “hash cracking online”, ecc..

Qui potete leggere un bell’articolo in inglese:
https://crackstation.net/hashing-security.htm

HASH E SALT

Come funzionano?

In sintesi e per semplificare se ho un dizionario di parole con tutti gli hash precalcolati, basta che cerco l'hash ABC123 a che parola corrisponde e trovo "pippo", chiaramente di funzioni di hash ce ne sono tante, ma le più utilizzate in molti sistemi sono l'MD5 e lo SHA1.

MD5 è una codifica a 128 bit, che genera un codice alfanumerico di 32 caratteri, mentre SHA1 è a 160 bit che genera un codice di 40 caratteri, se contiamo i caratteri dei codici presenti nel database possiamo intuire che tipo di algoritmo è stato utilizzato.

Chiaramente se la password è complessa e non è formata da parole di senso compiuto, difficilmente sarà presente in dizionari o tabelle varie, ma per essere più sicuri in genere le password vengono salvate con la codifica di hash + il sale!

HASH E SALT

Che c'entra l'arte culinaria adesso?
Niente! Infatti si chiama "salt" (sale), una sequenza casuale di caratteri che si aggiunge alla password da noi scelta, questa sequenza non la conosce nemmeno l'utente, ma la genera il sistema al quale accediamo:

hash("pippo" + "A@!763F") = DBB881

che è diverso da ABC123.
Quindi questo sale modifica l'hash della password, rendendo vano l'attacco a dizionario, ma se l'attaccante avesse anche il database del salt o sapesse l'algoritmo che genera i salt, potrebbe ricavare un sistema per provare tutte le combinazioni, ma facciamo un esempio semplice, se i miei salt sono combinazioni di numeri di massimo 4 cifre, diventa facile ad ogni parola del dizionario aggiungere tutte le 10.000 combinazioni (da 0000 a 9999) di numeri e calcolarne l'hash, per poi confrontarlo con quelli presenti nel database rubato.
Un altro sistema stupido per "salare", potrebbe essere quello di calcolare l'hash della password con lo username come salt:
hash("pippo" + ciccio@xxxxxx.com)

HASH E SALT

Nel caso Dropbox, questi hanno detto che metà delle password erano codificate con **bcrypt** (https://it.wikipedia.org/wiki/Bcrypt) ed altre con **SHA1 + salt**, quelle codificate con bcrypt sono più sicure, ma anche quelle con SHA1 + salt dovrebbero essere sicure perché "pare" che il database dei salt non sia stato preso, ma nel dubbio è meglio correre a cambiare la password.

Perchè con BCRYPT sono più sicure?

Perchè BCRYPT è un algoritmo più "lento".....

KDF E KEY STRETCHING

SCHEMA:

PASSWORD ---> KDF (funzione di derivazione della chiave) --> CHIAVE ---> TESTO CIFRATO/DECIFRATO

Capiamo la differenza tra password e chiave di criptazione

KDF E KEY STRETCHING

Quindi se possiamo generare 100.000 password al minuto possiamo generare 100.000 chiavi al minuto, ma questo sarebbe stupido, infatti il senso di derivare la chiave è anche nel rallentare l'attacco, ossia si fa in modo che la funzione di derivazione sia complessa in tal modo per ogni password provata/immessa, possiamo pensare che la funzione ci metta 1 secondo a creare la chiave, questo collo di bottiglia, abbasserebbe l'attacco all'immissione di 60 tentativi al minuto e non più 100.000, rendendo il tempo un fattore critico ed economico per l'attaccante.
La tecnica per fare questo si chiama Key Stretching https://en.wikipedia.org/wiki/Key_stretching , ossia effettuare una serie di elaborazioni tali da rallentare la generazione della chiave anche partendo da password deboli.

KDF E KEY STRETCHING

Per esempio, se una volta che immettiamo la password, la chiave di cifratura viene generata calcolando 500.000 volte l'hash della stessa + un salt:

VEDERE CODICE:
VIGENERE_stretched.py e sha256_500k.py
https://github.com/nannib/crypto

Hash("pluto")=ABD345
Hash("pluto" + 15BA) = FFC881

Vuol dire che il programma prima di comunicarci se abbiamo beccato la password o meno, ci mettera un tempo X per elaborare la chiave, così da rallentare il prossimo tentativo

```
C:\...>python sha256_500k.py
tempo iniziale: 2016-09-10 19:02:55.389000
tempo finale :  2016-09-10 19:02:56.985000
tempo totale:  0:00:01.596000
fine
b618b90015994f0bf0657648a469476aa96672eeaa17a9d609b0b98722406351
```

KDF E KEY STRETCHING

Possiamo quindi concludere che scegliere una password forte è importante per infastidire gli attacchi brute force e a dizionario ma che è altrettanto importante che la funzione di derivazione sia lenta, quindi non fidiamoci solo dei numeri altisonanti, “algoritmo a 128 bit!”, se non è implementato bene, la cifratura sarà fatta con una chiave a 128 bit ma se per generarla ci vuole meno di un decimo di secondo, non servirà a molto.

CRITTOANALISI

1883, La Criptographie Militaire

“la sicurezza di un crittosistema non deve dipendere dalla segretezza dell’algoritmo usato, ma solo dalla segretezza della chiave” (principio o legge di Kerckhoffs)

1949, Communication Theory of Secrecy Systems

“il nemico conosce il sistema”

(massima di Shannon)

CRITTOANALISI

Cipher Text Attack - si conoscono solo i crittogrammi

Known Plain-text Attack - si conoscono alcuni crittogrammi ed alcuni testi in chiaro che li hanno generati.

Chosen Plain-Text Attack - si conoscono alcuni testi in chiaro che passano per il canale cifrato

ed altri ancora....

CRITTOANALISI

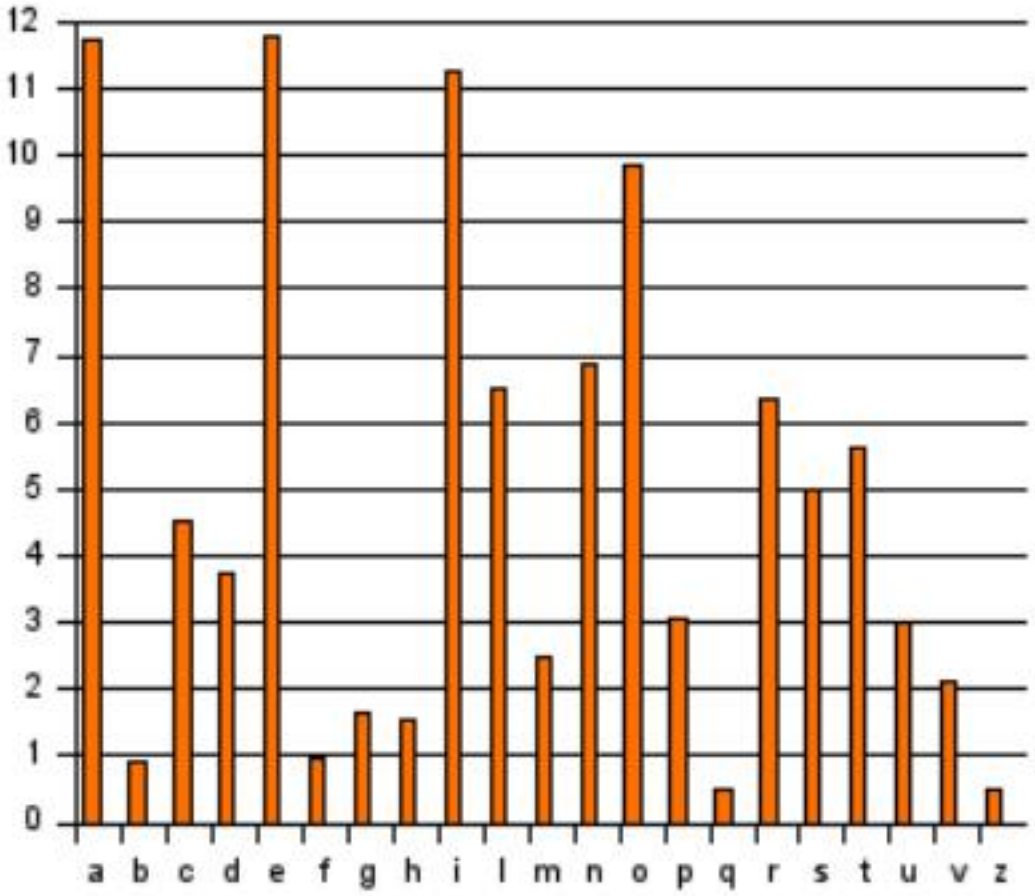
La crittoanalisi statistica

• Si utilizzano tecniche statistiche sulla frequenze dei caratteri o pattern del crittogramma e si ottengono informazioni utili sul testo in chiaro.

• Ad esempio con il cifrario di Cesare facendo un'analisi statistica delle lettere contenute nel testo cifrato e confrontando le frequenze con le frequenze dell’alfabeto italiano si può ricavare il testo in chiaro originale.

CRITTOANALISI

https://it.wikipedia.org/wiki/Analisi_delle_frequenze



Lettera	Frequenza
e	11.79%
a	11.74%
i	11.28%
o	9.83%
n	6.88%
l	6.51%
r	6.37%
t	5.62%
s	4.98%
c	4.50%
d	3.73%
p	3.05%
u	3.01%
m	2.51%
v	2.10%
g	1.64%
h	1.54%
f	0.95%
b	0.92%
q	0.51%
z	0.49%

CRITTOANALISI

Esempio di crittoanalisi statistica del cifrario di Cesare

Il testo cirato con CESARE (spostamento +3 caratteri)

OD FDSUD FDPSD

frequenze:

D(5/12), F(2/12), S(2/12), O(1/12), U(1/12), P(1/12)

D F S O U P

e a i o n l

OE AEINE AELSE

provo

D F S O U P

a c e l n i

LA CAENA CAIEA che se lo vedo così LA CAxxA CAxxA potrebbe essere: LA CAPRA CAMPA

CRITTOANALISI

Alcune tecniche:

• Brute-force - calcolo di tutte le possibili combinazioni di chiavi.

• Crittoanalisi differenziale - Si analizzano le “distanze” numeriche dei caratteri presenti nel testo (**vedi Kasiski** **https://it.wikipedia.org/wiki/Metodo_Kasiski**)
http://www.crittologia.eu/critto/kasiski.htm

• Man-in-the-middle - intercettazione delle chiavi sul canale di trasmissione-

CRITTOANALISI

Kasiski

python vigenere_nb.py

Vuoi criptare o decriptare? (c/d):c

Inserisci un testo da criptare:ilmondodelmarepiaceatuttiilmondodelcielononatutti

testo: ilmondodelmarepiaceatuttiilmondodelcielononatutti

Inserisci un chiave di criptazione:ciao

chiave: ciao

newkey= ciaociaociaociaociaociaociaociaociaociaociaoc

testo criptato: ktmcplorgtmotmpwckeovcthkqlaqvdcfmlqkmlcpwnovcthk

CRITTOANALISI

Esempio di uso del metodo Kasiski

fonte: https://it.wikipedia.org/wiki/Metodo_Kasiski

Quando una stessa porzione di chiave è usata per cifrare una ripetizione contenuta nel testo da cifrare, essa genera una ripetizione corrispondente nel testo cifrato. Questo equivale a dire che l'intervallo fra due gruppi di lettere cifrate uguali è un multiplo della lunghezza della chiave. L'analisi di questi intervalli può quindi fornire la lunghezza della chiave, cioè il numero di alfabeti usati per cifrare. È però necessario individuare e scartare le eventuali ripetizioni che siano semplicemente dovute al caso e non alla ripetizione della chiave (di solito, gruppi di lettere non più lunghi di due o tre caratteri). In sintesi si deve:

1. **Individuare i gruppi di lettere ripetuti ed elencarli indicando le posizioni in cui le loro lettere iniziali si trovano nel crittogramma**
2. **Calcolare gli intervalli fra le ripetizioni eseguendo la differenza fra i due numeri**
3. **Scomporre questi intervalli in tutti i loro sottomultipli**
4. **Individuare i fattori maggiormente ricorrenti, dando più peso a quelli relativi ai gruppi di lettere più lunghi, i quali più difficilmente saranno casuali.**

CRITTOANALISI

Kasiski

python vigenere_nb.py

Vuoi criptare o decriptare? (c/d):c

Inserisci un testo da criptare:ilmondodelmarepiaceatuttiilmondodelcielononatutti

testo: ilmondodelmarepiaceatuttiilmondodelcielononatutti

Inserisci un chiave di criptazione:ciao

chiave: ciao

newkey= ciaociaociaociaociaociaociaociaociaociaociaoc

testo criptato: ktmcplorgtmotmpwckeovcthkqlaqvdcfmlqkmlcpwnovcthk

CRITTOANALISI

divido per bigrammi e conto le occorrenze ripetute e le distanze (si conta da 0).

kt mc pl or gt mo tm pw ck eo vc th kq la qv dc fm lq km lc pw no vc th k

PW trovato a 14 e a 40 distanza 40-14=26
TH trovato a 22 e a 46 distanza 46-22= 24
VC trovato a 20 e a 44 distanza 44-20= 24
24 è divisibile per 2,3,4,6,8,12 (sottomultipli)
24 è divisibile per 2,3,4,6,8,12
26 è divisibile per 2,13
La lunghezza candidata ad esser quella giusta pare sia di 2 (MCD delle distanze) caratteri, ma è troppo breve, quindi 4,3,8,6,8,12 sono le lunghezze immeditamente successive.

https://www.it.uu.se/edu/course/homepage/security/vt08/labs/vigenere.html

CRITTOANALISI

Prendiamo la lunghezza 4, ogni 4 lettere vuol dire che è cifrata con la stessa lettera chiave.

ktmc plor gtmo tmpw ckeo vcth kqla qvdc fmlq kmlc pwno vcth k

Prima lettera: kpgtcvkqfkpvk

ktmc plor gtmo tmpw ckeo vcth kqla qvdc fmlq kmlc pwno vcth k

Seconda lettera: tltmkcqvmmwc

ktmc plor gtmo tmpw ckeo vcth kqla qvdc fmlq kmlc pwno vcth k

Terza lettera: mompetldllnt

ktmc plor gtmo tmpw ckeo vcth kqla qvdc fmlq kmlc pwno vcth k

Quarta lettera: crowohacqcoh

CRITTOANALISI

ANALISI DELLE FREQUENZE E SCORE

Le prime 6 (sei) lettere più frequenti nell'Italiano sono
eaionl

Le ultime 6 lettere meno frequenti sono:
ghfbqz

In una frase assegnamo un punto se tra le prime 5 lettere più frequenti appare una appartenente a EAION ed un punto se tra le ultime 5 lettere meno frequenti appare una appartenente a HFBQZ
ESEMPIO: IO AMO ANDARE AL MARE
I(1),O(2),A(5),M(2),N(1),D(1),R(2),E(2),L(1)
quindi la classifica è: AMEROI NDL nei primi 5 c'è A,E,O quindi lo score è 3!

Cifrario di Cesare con chiave singola lettera per ogni subkey individuata, cominciamo con la prima subkey

kpgtcvkqfkpv

```
python freq_nb.py
Inserisci un testo da decriptare:kpgtcvkqfkpv
testo: kpgtcvkqfkpvk
N. Lett. Decr.      OrdFreq.    Score
1  a   kpgtcvkqfkpvk   kpvcgfqt   1
2  b   jofsbujpejouj   joubefps   2
3  c   ineratiodinti   intaedor   7
4  d   hmdqzshnchmsh   hmscdnqz   1
5  e   glcpyrgmbglrg   glrcbmpy   2
6  f   fkboxqflafkqf   fkqablox   3
7  g   ejanwpekzejpe   ejpaknwz   3
8  h   dizmvodjydiod   diojmvyz   2
9  i   chyluncixchnc   chniluyx   3
10 j   bgxktmbhwbgmb   bgmhktwx   1
11 k   afwjslagvafla   aflgjswv   2
12 l   zevirkzfuzekz   zekfiruv   3
13 m   yduhqjyetydjy   ydjehqut   2
14 n   xctgpixdsxcix   xcidgpst   2
15 o   wbsfohwcrwbhw   wbhcfosr   2
16 p   varengvbqvagv   vagbenqr   4
17 q   uzqdmfuapuzfu   ufzadmqp   1
18 r   typcletzotyet   teyclopz   4
19 s   sxobkdsynsxds   sdxbkony   2
20 t   rwnajcrxmrwcr   rcwajmnx   3
21 u   qvmzibqwlqvbq   qbvimlwz   2
22 v   pulyhapvkpuap   pauhklvy   2
23 w   otkxgzoujotzo   otzgkjux   2
24 x   nsjwfyntinsyn   nsyfijtw   3
25 y   mrivexmshmrxm   mrxeihsv   3
26 z   lqhudwlrglqwl   lqwdghru   2
```

http://www.dcode.fr/caesar-cipher

CODICE:

freq_nb.py https://github.com/nannib/crypto

Cifrario di Cesare con chiave singola lettera per ogni subkey individuata, cominciamo con la prima subkey

tltmkcqvmmwc

python freq_nb.py

testo: tltmkcqvmmwc

N.	Lett.	Decr.	OrdFreq.	Scor
1	a	tltmkcqvmmwc	mctklqwv	2
2	b	sksljbpullvb	lbskjpuv	1
3	c	rjrkiaotkkua	karijout	5
4	d	qiqjhznsjjtz	jqzihnst	3
5	e	phpigymriisy	ipyghmsr	2
6	f	ogohfxlqhhrx	hoxgflqr	3
7	g	nfngewkpggqw	gnwefkqp	2
8	h	memfdvjoffpv	fmvedjop	2
9	i	ldlecuineeou	elucdion	5
10	j	kckdbthmddnt	dktcbhmn	2
11	k	jbjcasglccms	cjsabgml	2
12	l	iaibzrfkbblr	birafklz	4
13	m	hzhayqejaakq	ahqekjyz	2
14	n	gygzxpdizzjp	zgpdijyx	1
15	o	fxfywochyyio	yfocihwx	2
16	p	ewexvnbgxxhn	xenbghwv	2
17	q	dvdwumafwwgm	wdmagfuv	1
18	r	cucvtlzevvfl	vclefutz	3
19	s	btbuskyduuek	ubkedsty	2
20	t	asatrjxcttdj	tajcdsrx	3
21	u	zrzsqiwbssci	sizcbqrw	2
22	v	yqyrphvarrbh	rhyabqpv	2
23	w	xpxqoguzqqag	qgxaopuz	2
24	x	wowpnftyppzf	pfwontyz	3
25	y	vnvomesxooye	oevmnsyx	3
26	z	umunldrwnnxd	ndumlrwx	3

CRITTOANALISI

Abbiamo per la prima lettera: c

per la seconda: c,i

per la terza:a

per la quarta: d,o

combinazioni 1x2x1x2=4 (contro le 26^4=456976 se avessimo usato tutti i caratteri dell'alfabeto):

ccad

ccao

ciad

ciao

CHIARAMENTE è CIAO la chiave utilizzata.

RSA

OPERAZIONE MODULO

Tra i numeri interi è definita la funzione *modulo*, indicato con $\mathbf{mod}$, che dà come risultato il resto della divisione euclidea del primo numero per il secondo. Cioè dati $a, b \in \mathbb{Z}$, con $b \neq 0$ allora $a \bmod b$ dà come risultato il resto della divisione euclidea $\frac{a}{b}$.

Per esempio, si ha $13 \bmod 3 = 1$, perché $\lfloor 13/3 \rfloor = 4$, quindi $13 - (3 \cdot 4) = 1$ e dunque il resto è 1.

Se $b > a$, allora $a \bmod b = a$.

Ad esempio $3 \bmod 7 = 3$, perché $\lfloor 3/7 \rfloor = 0$, quindi $3 - (7 \cdot 0) = 3$ e dunque il resto è proprio 3.

RSA

La sua forza risiede nella difficcoltà computazionale di scomporre il numero n in due fattori primi.

RSA costruiamo le chiavi:

1. Scegliere due numeri primi molto grandi: **p**, **q** tali per cui **p*q** sia di almeno 1024 bit.
2. Calcolare **n = pq** e la funzione di Eulero **z = (p-1)(q-1)**
3. Scegliere **e** (con e < n) tale che non abbia fattori in comune con z, ossia z mod e > 0
4. Scegliere **d** tale che ed-1 sia esattamente divisibile per z, ossia: ed mod z = 1
5. Otteniamo la chiave pubblica **Kpub(e, n)** e la chiave privata **Kpriv(d, n)**

RSA

Cifratura

Testo in chiaro: m < n

Kpub(e, n)

c = m^e mod n

Decifratura

Testo cifrato: c

Kpriv(d, n)

m = c^d mod n

RSA

Dimostrazione:

m = (m^e mod n)^d mod n

(m^e mod n)^d mod n = m^(e*d)mod n= me^(d m o d (p - 1) (q - 1)) mod n = m^1 mod n = m

(perché abbiamo scelto che e e d siano divisibili per (p-1)(q-1) con resto 1)

RSA

ESEMPIO:

Lettera F nell'alfabeto è alla posizione 5 (A=0,B=1,ecc.)

p=5 q=11 n=55 z=(5-1)(11-1)=4*10=40

devo trovare e tale che z mod e>0 quindi 40 mod e>0

40 mod 1 = 0 40 mod 2 = 0 40 mod 3 = 1

Ok e=3

Trovo d tale che ed mod z=1 quindi 3*d mod 40=1

3*[3,5,7....] mod 40 = 1 trovo d=27 (z è pari per forza, d deve essere dispari, per far sì che e*d non sia divisibile per z)

QUINDI:

p=5,q=11,n=55,z=40,e=3,d=27

RSA

ALFABETO:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

CRIPTIAMO:

Lettera F posizione 5 quindi m=5 (m messaggio in chiaro)

c=m^e mod n = (5^3) mod 55 = 125 mod 55 = 15 = lettera P

DECRIPTIAMO

m=c^d mod n = (15^27) mod 55 = 5 = lettera F

VEDERE CODICE:

RSA_NB.py RSA_NB_dkey.py RSA_CRACK.py

https://github.com/nannib/crypto

RSA

Per questioni computazionali è meglio cifrare con RSA la chiave di sessione, ossia una chiave che cripta/decripta:

Documento criptato con KS --> RSA(KS)
Decriptare RSA(KS) --> KS --> Documento

FIRMA DIGITALE

VEDERE CODICE:
firmadigitale_nb.py
https://github.com/nannib/crypto

La firma digitale funziona diversamente.
Serve ad autenticare, identificare il mittente e garantire l'integrità del documento firmato.

Documento --> Hash(Doc) --> CriptChiavePrivata(Hash(Doc))

DecryptChiavePubblica(Hash(Doc))-->Hash(Doc)

Così otteniamo che il doc è conforme perchè se lo “hashiamo” avremo lo stesso hash presente nella firma e che lo ha mandato il mittente vero perchè siamo riusciti ad estrarlo usando la sua Kpub.

FIRMA DIGITALE

Ma se inviassi una chiave pubblica mia spacciandomi per un altro? Ecco che servono i CERTIFICATI emessi da un'Authority certa che crea un certificato firmando la Kpub del mittente con la propria Kpriv, così chiunque avendo la KPub dell'Autorità certificante, può verificare che il certificato è autentico e che la Kpub del mittente è quella vera.

TRUECRYPT MODIFICATO

VEDERE VIDEO:
https://www.youtube.com/watch?v=ss8WUecpRNs
http://downloads.volatilityfoundation.org/omfw/2013/OMFW2013_Ligh.pdf

PC ACCESO --> Volume TC montato --> dump della RAM --> Estrazione della master key dalla RAM

Creazione di un volume TC nuovo della stessa dimensione del TC di cui non conosciamo la PW

Sovrascrittura dei primi 512 bytes del NewTC sul TC

Adesso Truecrypt ci fa inserire la PW di NewTC su TC ma non lo monta, perchè la Master Key non corrisponde, allora ricompiliamo Truecrypt con la patch e gli facciamo leggere la MK da file esterno dove conserviamo quella presa dalla RAM....et voilà!