

Implementazione di un simulatore di macchina Enigma in Python

Matteo Ghera

Università degli Studi di Firenze

7 Settembre 2020

- La macchina Enigma era un dispositivo, simile ad una macchina da scrivere, utilizzato dai tedeschi durante la Seconda Guerra Mondiale per criptare e decriptare un messaggio;
- Esistevano diverse versioni di Enigma e ogni comparto dell'esercito utilizzava una diversa versione;
- La scoperta di un procedimento che consentisse di decriptare i messaggi nemici da parte degli Alleati era una sfida importante per le sorti della guerra;
- Secondo alcuni storici, tale scoperta contribuì a ridurre la durata della guerra di due anni;
- Lo scopo del mio progetto è quello di implementare un simulatore di una macchina Enigma in Python.



Figure 1: Foto di una macchina Enigma esposta presso il Museo nazionale della scienza e della tecnologia “L. Da Vinci” di Milano

La struttura di Enigma

- Enigma può essere considerata come un'estensione del metodo del cifrario di Vigenère munita di dischi cifrati.
- Gli elementi elettromeccanici che componevano una macchina Enigma erano i seguenti:
 - ① una tastiera (*keyboard*);
 - ② un pannello luminoso (*lampboard*);
 - ③ tre *rotori* connessi a cascata;
 - ④ un *riflettore*;
 - ⑤ un pannello a più prese (*plugboard*).

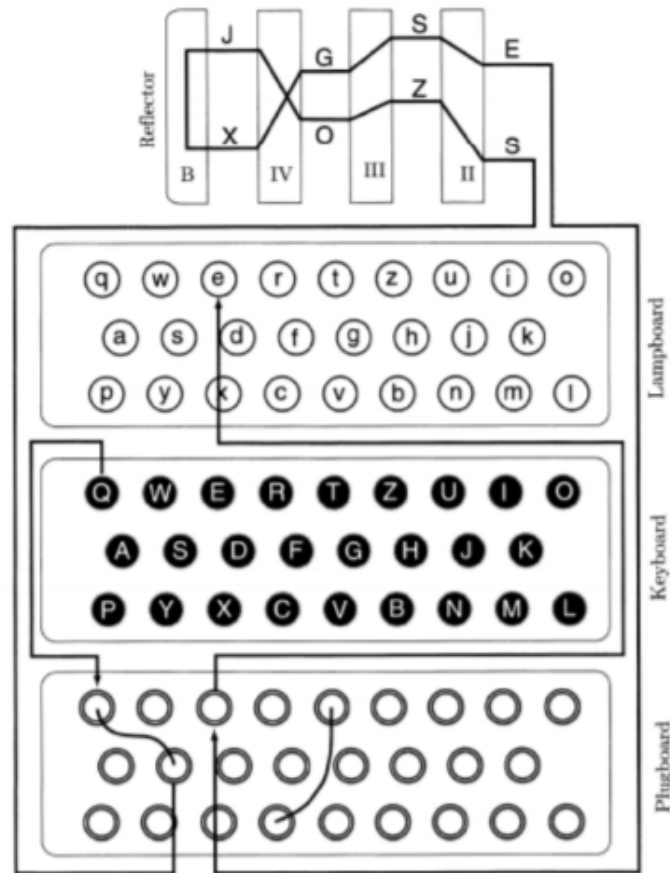


Figure 2: Schema che rappresenta il funzionamento di una macchina Enigma tratto da [6]

- Si consideri una generica macchina Enigma;
- Sia S l'insieme fissato dei rotori, dove ciascun rotore realizza una permutazione nota fra le lettere, allora una *chiave* di Enigma consiste di:
 - una tripla $(R_1, R_2, R_3) \in S$ di rotori;
 - una tripla $(i, j, k) \in \mathbb{Z}_{26}^3$ che specifica la posizione iniziale di ciascuno dei tre rotori;
 - una permutazione σ dell'alfabeto con esattamente sei punti fissi che specifica le connessioni della plugboard.
- (R_1, R_2, R_3, i, j, k) rappresenta lo *stato* della macchina enigma in un certo momento.

Numero di possibili chiavi

Il numero di possibili chiavi K è dato da:

$$\begin{aligned} K &= (5 \cdot 4 \cdot 3) \cdot 26^3 \cdot \frac{20!}{(20-10)!} \cdot \frac{1}{2^{10}} \cdot \binom{26}{20} \\ &= (5 \cdot 4 \cdot 3) \cdot 26^3 \cdot \frac{20!}{10!} \cdot \frac{1}{2^{10}} \cdot \frac{26!}{20! \cdot 6!} \\ &= (5 \cdot 4 \cdot 3) \cdot 26^3 \cdot \frac{26!}{(6! \cdot 10! \cdot 2^{10})} \approx 10^{20} \text{ oppure } \approx 2^{67.1} \end{aligned}$$

Funzioni di Encryption e Decryption

- Si supponga che
 - a. $\alpha, \beta, \gamma, \pi$ e σ siano le trasformazioni applicate alle lettere dal rotore sinistro, dal rotore centrale, dal rotore destro dal riflettore e dalla plugboard;
 - b. $\alpha^{-1}, \beta^{-1}, \gamma^{-1}$ e σ^{-1} siano le trasformazioni all'indietro applicate alle lettere dalle diverse componenti.
- La funzione di Encryption alla fase iniziale è data da:

$$E(x) = \sigma^{-1} \circ \alpha^{-1} \circ \beta^{-1} \circ \gamma^{-1} \circ \pi \circ \gamma \circ \beta \circ \alpha \circ \sigma(x)$$

- Se si indica con $\mu^{-n}R\mu^n$ la rotazione del generico rotore R di n posizioni, si ottiene la funzione di Encryption generica:

$$E(x) = \sigma^{-1} \circ \left(\mu^{-i} \alpha^{-1} \mu^i \right) \circ \left(\mu^{-j} \beta^{-1} \mu^j \right) \circ \left(\mu^{-k} \gamma^{-1} \mu^k \right) \circ \pi \\ \circ \left(\mu^{-k} \gamma \mu^k \right) \circ \left(\mu^{-j} \beta \mu^j \right) \circ \left(\mu^{-i} \alpha \mu^i \right) \circ \sigma(x)$$

- σ e π sono involuzioni;
- σ ha esattamente sei punti fissi mentre π non ha punti fissi,

Proposizione 1

Sia $\rho = \sigma^{-1} \circ \tilde{\tau} \circ \sigma$, dove $\tilde{\tau} = \alpha^{-1} \circ \beta^{-1} \circ \gamma^{-1} \circ \pi \circ \gamma \circ \beta \circ \alpha$, una permutazione allora ρ è un'involuzione e non ha punti fissi.

Implementazione del simulatore in Python

POSIZIONE ROTORI DENTRO LA MACCHINA				
<i>Riflettore</i>	<i>Rotore sinistro</i>	<i>Rotore al centro</i>	<i>Rotore destro</i>	<i>Iniziale</i>
T U	E O	E L---	---I Y	Q
E I	Q A	Y Q	J X	W
V O	D S	U W	U C	E
---J A---	---L D	F E	H V	R
C S	B F	H R	X B	T
L D	C G	X T	Z N	Z
X F	K H	Z Z	M M	U
Z G	J J	M U	N L	I
R H	G K	N I	A Q	O
---A J---	---V P	J O	V W	A
Q K	P Y	G A---	---O E	S
N P	U X	O S	E R	D
B Y	R C	P D	Y T	F
F X	W V---	---A F	F Z	G
S C	N B	Q G	W U	H
O V	X N	I H	L I---	---J---
Y B	A M	R J	D O---	---K---
P N	H L---	---L K	Q A	P
W M	S Q	D P	C S	Y
D L	T W	T Y	B D	X
K Q	I E	W X	S F	C
M W	O R	V C	P G	V
I E	F T	K V	T H	B
H R	M Z	S B	K J	N
U T	Y U	B N	R K	M
G Z	Z I	C M	G P	L

Figure 3: Tabella delle permutazioni ottenute con la configurazione UOLY

La classe *EnigmaRotor*

`__init__(self, entrata, uscita, rotore_succ=None, flag=True)`

- `entrata` e `uscita` definiscono la permutazione realizzata dal rotore corrente;
- `rotore_succ` indica il rotore successivo (se esiste) a cui sarà inviato il segnale di rotazione una volta completato il giro;
- `flag` indica se il rotore corrente può essere ruotato.

`impostaPosizione(self, elemento)`

- Imposta la configurazione iniziale del rotore corrente alla lettera indicata da `elemento`

posizioneSinistra(self, posizione)

- Ricerca la lettera nel vettore uscita corrispondente alla lettera presente alla posizione indicata nel vettore entrata

posizioneDestra(self, posizione)

- Analogo al precedente







muovi(self)

- Se è consentito, muove il rotore di una posizione verso l'alto e invia un segnale al rotore successivo (se esiste)

La classe *EnigmaMachine*

- La classe Python *EnigmaMachine* simula il comportamento di una macchina Enigma;
- Il costruttore di tale classe riceve in ingresso quattro oggetti di tipo *EnigmaRotor* (*riflettore*, *rotoreSinistro*, *rotoreCentrale* e *rotoreDestro*) e due liste (*alfabeto* e *plugboard*);
- Il metodo *impostaEnigma* sarà utilizzato per impostare le posizioni dei rotori e del riflettore;
- Il metodo *esegui* cripta/decripta il messaggio in input.
- L'alfabeto scelto è il QWERTZ;
- La lista *plugboard* è una sequenza di numeri da 0 a 25 che realizzano lo scambio tra coppie di lettere

Bibliografia

-  Enigma (crittografia), 2019.
[https://it.wikipedia.org/wiki/Enigma_\(crittografia\)](https://it.wikipedia.org/wiki/Enigma_(crittografia)).
-  Una macchina enigma virtuale, 2019.
<http://www.crittologia.eu/critto/js/enigma.html>.
-  M. Boreale - *Corso di data security e privacy, esercizi su crittografia a chiave condivisa*, 2018. <https://e-l.unifi.it/pluginfile.php/780653/course/section/87266/Esercizi>.
-  S. Keegan, John - *Intelligence in warfare* - New York: Alfred A. Knopf, 2003
-  M. Rejewski - *An application of the theory of permutations in breaking the enigma cipher* - *Applicationes Mathematicae*
-  D. Salomon - *Data Privacy and Security* - Springer, 2003