Esercizio 3.2 – Cifrario di Hill

Il seguente esercizio è stato svolto usando il linguaggio Python (versione 3.7.10) e sono state usate le **librerie** *Numpy*, *String* e *Sympy*.

Istruzioni per il funzionamento: scaricare le 3 librerie necessarie al progetto ed eseguire il file *main.py*

Per questo esercizio, sono state implementate le seguenti **funzioni**:

- letterToNumber(letter): Trasforma le lettere nella corrispettiva posizione nell' alfabeto (a è 0, b è 1, …)
- numberToLetter(number): Funzione inversa di letterToNumber()
- encrypt(message, key): Funzione che cripta il messaggio data la chiave
- decrypt(encryption, key): Funzione che decripta la cifratura data la chiave
- attack_Hill(plaintext, ciphertext, m): Funzione che prova l'attacco al cifrario di Hill

Viene descritto adesso il **funzionamento** del programma.

Una volta inserita la chiave, definita come una matrice quadrata invertibile, si decide un plaintext da trasmettere e questo viene salvato in una lista trasformato in numeri (eliminando eventuali spazi vuoti nel testo).

Se il modulo tra la lunghezza del messaggio e le righe della chiave fosse diverso da 0, si andrebbe ad aggiungere delle lettere al messaggio iniziale (per non dare errore nella successiva moltiplicazione in quanto la dimensione del messaggio deve essere uguale al numero di righe della chiave).

Cifratura → Si esegue la funzione encrypt(message, key) : il messaggio, da lista, viene trasformato in ndarray e poi, considerando m lettere alla volta, si esegue la moltiplicazione matriciale con il metodo numpy.matmul() tra la matrice *encryption* e la chiave *key*, eseguendone il modulo 26. Si ritorna *encryption* come ndarray.

Decifratura → Si esegue la funzione decrypt(encryption, key) : si calcola la matrice inversa della chiave *key* modulo 26 con il metodo Matrix(key).inv_mod(26) della libreria *Sympy.* A questo punto, sempre considerando m lettere alla volta, si esegue la moltiplicazione matriciale tra la matrice *decryption* e la chiave inversa *inverse_key*, eseguendone il modulo 26. Calcolato *decryption*, si provvede a trasformarlo in lettere col metodo numberToLetter() e si ritorna la stringa *decrypted_message*.

Attacco di Hill → Si esegue la funzione attack_Hill(plaintext, ciphertext, m) : si passa come argomenti il plaintext come numeri (*message)* in formato ndarray, *encryption* come numeri in formato ndarray e *m* corrisponde al numero di righe della chiave. Si suddividono queste matrici in sottomatrici (vedere esempio 3.1 a pagina 19 delle Note come riferimento) e si creano P^* e C^* . Si calcola poi la matrice inversa di P^* (P^{*-1}), se ne esegue il modulo 26 e la chiave *key* risulta essere il prodotto matriciale $C^* \; x \; P^{*-1}$. Si ritorna *key* come ndarray.