REPORT S3/L4

Crittografia

Traccia

Dato un messaggio cifrato cercare di trovare il testo in chiaro.

Messaggio cifrato:

1. **HSNFRGH**
2. **QWJhIHZ6b2VidHl2bmdyIHB1ciB6ciBhciBucHBiZXRi**

Svolgimento

1. Primo messaggio

Consideriamo il primo messaggio cifrato. Data l'assenza di caratteri speciali e la brevità della stringa possiamo ipotizzare che il messaggio sia stato criptato usando il **Cifrario di Cesare**, in cui ogni lettera è spostata di un certo numero di posti nell'alfabeto. Dato che non sappiamo di quanto sia lo spostamento possiamo andare per tentativi, oppure basarci sulla probabilità di apparizione di ogni lettera dell'alfabeto.

Secondo uno studio del 2024, per la lingua italiana, queste probabilità sono:

• E (12,7%) • T (9,5%) • A (8,2%) • O (7,5%) • I (6,9%) • N (6,4%) • S (6,1%) • R (5,7%) • C (5,4%) • U (4,8%) • M (4,5%) • L (4,2%) • P (3,9%)	• G (3,6%) • D (3,4%) • F (3,2%) • H (2,9%) • V (2,6%) • B (2,4%) • K (2,1%) • J (1,9%) • Q (1,6%) • X (1,4%) • Y (1,2%) • Z (0,9%)

Basandoci su queste ipotesi, teniamo conto che la lettera H corrisponda in realtà alla lettera E, dandoci pertanto come risultato dello spostamento quello di 3 lettere.

Arretrando tutte le successive lettere del messaggio cifrato di 3 posizioni nell'alfabeto, otteniamo le seguenti corrispondenza:

• H -> E • S -> P • N -> I • F -> C	• R -> O • G -> D • H -> E

Pertanto Il messaggio cifrato corrisponde alla parola "**EPICODE**".

2. Secondo messaggio

Per il secondo messaggio, dato il modo in cui è scritto, ci rendiamo conto che è stato scelto un altro metodo per codificarlo. In particolare notiamo che:

1. **Formato di stringa**: La stringa presenta un formato specifico, con caratteri tra A-Z, a-z, 0-9.
2. **Lunghezza**: La lunghezza della stringa è un multiplo di 4, in particolare sono 44 lettere.

Questo ci porta a pensare che il messaggio sia stato codificato in Base64. Infatti i caratteri citati nel punto 1, unitamente a +, / e = sono quelli scelti per rappresentare i messaggi in questo formato. In più la lunghezza del messaggio, come descritto nel punto 2 è multiplo di 4 poiché ogni 4 byte di dati binari sono rappresentati da 4 caratteri ASCII.

Ricordiamo, per dovere di cronaca che la crittografia Base64 non è propriamente una crittografia, ma una codifica di tipo "encoding" che trasforma dati binari in stringhe ASCII. La differenza tra encoding e crittografia è un concetto fondamentale nell'informatica e nella sicurezza dei dati. Entrambe le tecniche vengono utilizzate per trasformare i dati, ma hanno scopi e metodi diversi.

- **Encoding**: è un processo che trasforma i dati in un formato diverso per renderli utilizzabili da sistemi differenti o per ridurre le dimensioni dei dati. L'encoding non è una tecnica di sicurezza e non protegge i dati da accessi non autorizzati. Esempi di encoding sono la codifica dei caratteri (ad esempio, ASCII, UNICODE), la codifica dei dati in formato binario (ad esempio, Base64) e la compressione dei dati.

- **Crittografia**: è un processo che trasforma i dati in un formato illeggibile per proteggerli da accessi non autorizzati. La crittografia utilizza algoritmi e chiavi per trasformare i dati in un formato cifrato che può essere decrittografato solo con la chiave appropriata. La crittografia è una tecnica di sicurezza che protegge i dati da intercettazioni e accessi non autorizzati.

Premesso ciò, tornando al nostro messaggio usiamo un comodo tool online per decodificare il messaggio. Vediamo nello screen sotto il risultato:

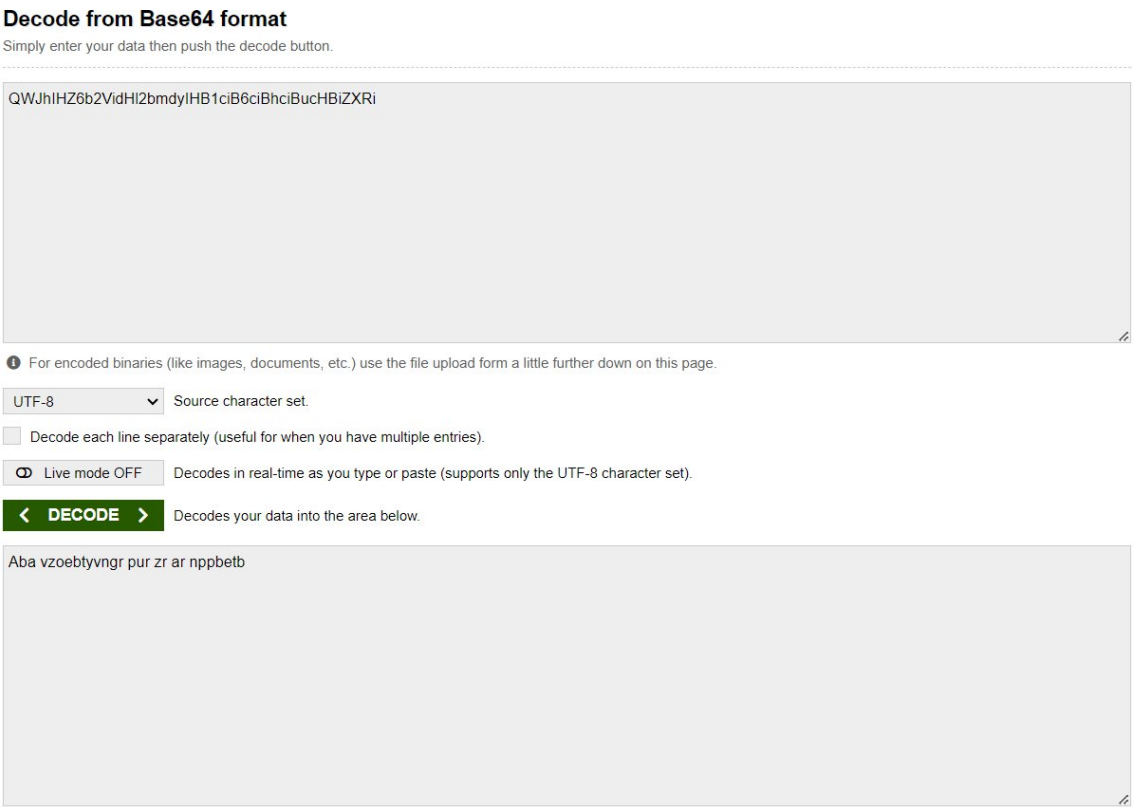

Figura 1. Decodifica in base64

Otteniamo una nuova stringa "**Aba vzoebtyvngr pur zr ar nppbetb**" che ovviamente non ha senso compiuto, ma rientra nella casistica del primo messaggio che abbiamo decriptato, pertanto anche stavolta proviamo ad usare il **cifrario di Cesare** e a stabilire di quanto ci dobbiamo muovere sull'alfabeto.

Questa volta poniamo l'attenzione sulla prima parola "Aba". É una parola palindroma di 3 lettere e in italiano non ne esistono molte di senso compiuto. In più la prima e la seconda lettera sono consecutive (a e b), pertanto lo saranno anche le lettere che compongono il messaggio in chiaro e questo riduce di molto l'elenco. La prima parola che ci viene in mente e che rispecchia i parametri descritti sopra (palindroma, di 3 lettere, con le prime due lettere consecutive) è "**non**". Il che ci porta a concludere che probabilmente la misura che dobbiamo usare per decifrare la frase è di 13 lettere. Anche stavolta, per velocizzare il processo usiamo un tool online che cripta/decripta usando il cifrario di Cesare. Di seguito lo screen del risultato:

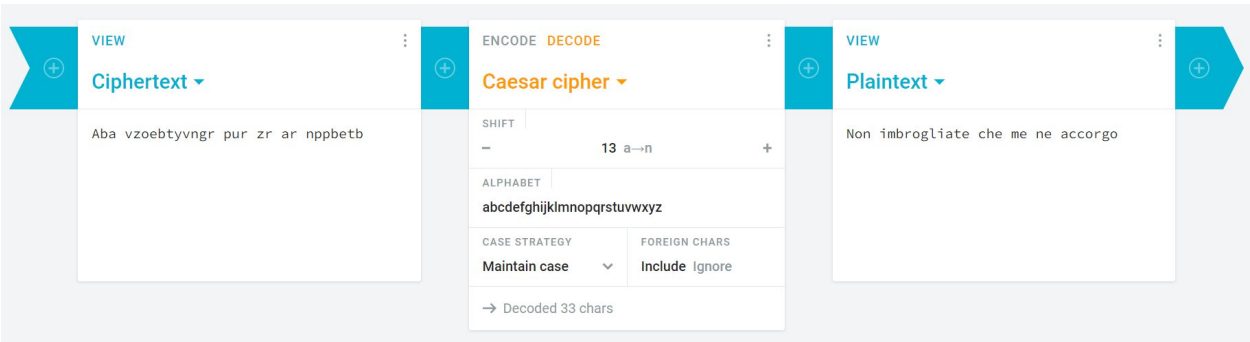

Figura 2. Decifratura usando il "cifrario di Cesare"

La frase in chiaro è "Non imbrogliate che me ne accorgo", nota frase spesso e volentieri ripetuta dal nostro bravissimo prof. Antonio Pozzi.

Traccia bonus

Criptazione e Firmatura con OpenSSL e Python. Obiettivi dell'esercizio:

- Generare chiavi RSA.
- Estrarre la chiave pubblica da chiave privata.
- Criptare e decriptare messaggi.
- Firmare e verificare messaggi.

Strumenti utilizzati:

- OpenSSL per la generazione delle chiavi.
- Libreria cryptography in Python.

Svolgimento

Si tratta di un esercizio guidato, di cui abbiamo i passaggi nelle slide di oggi, quindi proviamo a riprodurli e nel caso di eventuali problemi, vediamo come poterli risolvere.

Innanzitutto dobbiamo installare OpenSSL e la libreria python sulla macchina virtuale di kali, pertanto avviamo la macchina, apriamo il terminale ed eseguiamo i seguenti comandi:

1. sudo apt update
2. sudo apt install openssl

Questi ci permetteranno di aggiornare le dipendenze del sistema operativo e successivamenti di installare openssl.

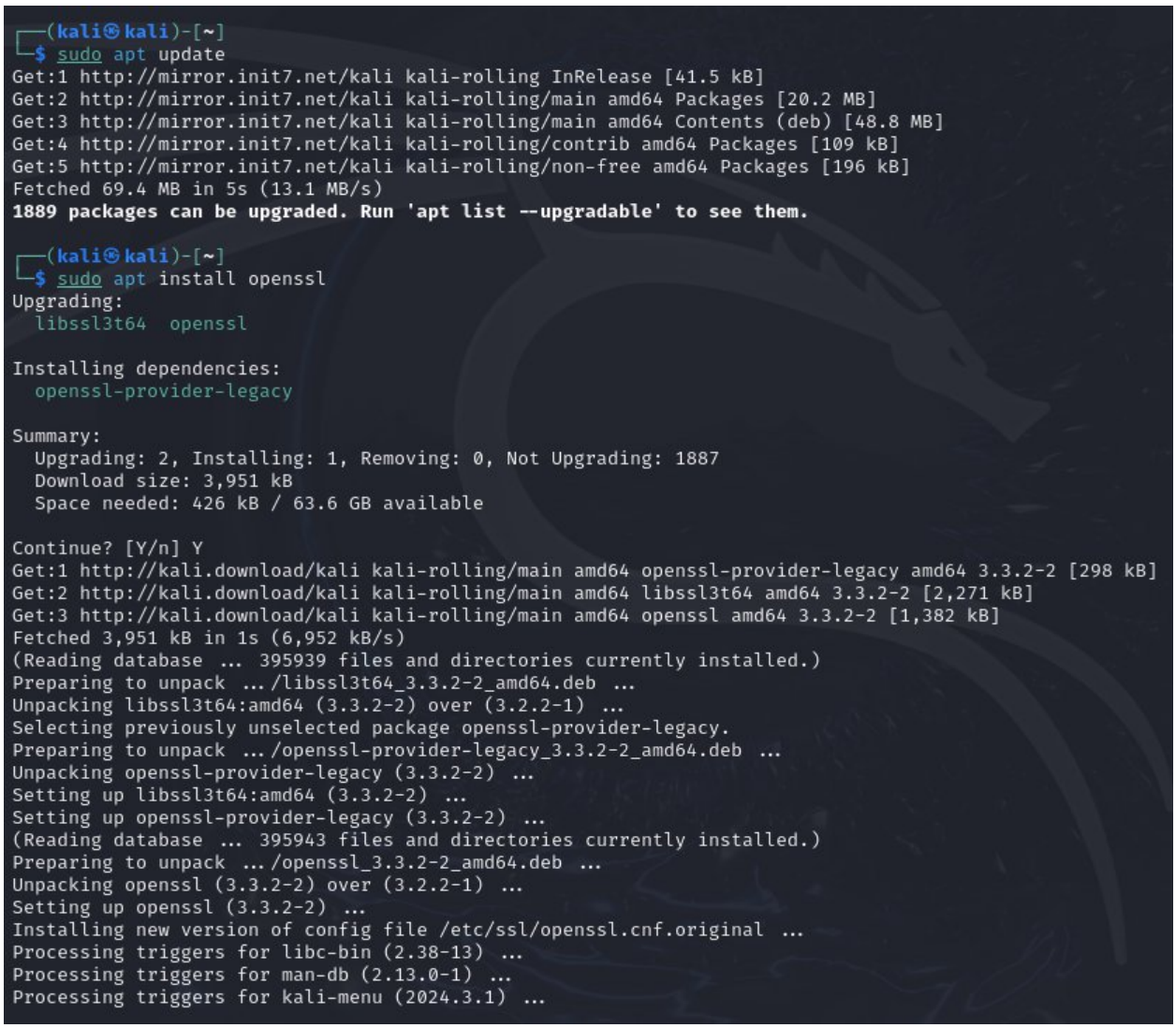

Figura 1. Aggiornamento e installazione OpenSSL

A questo punto installiamo la libreria cryptography per python tramite i comandi:

- sudo apt install python3-pip
- pip3 install cryptography

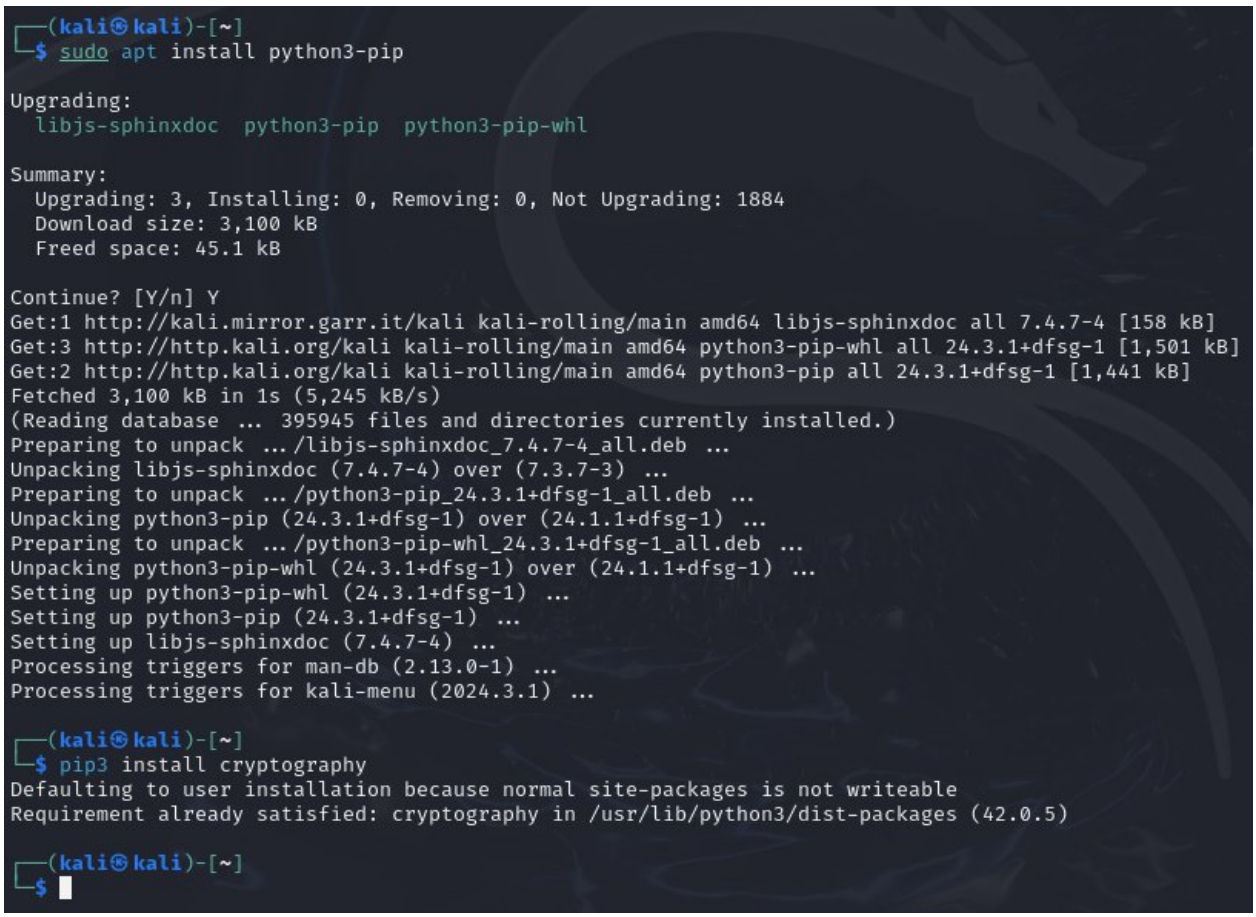

Figura 2. Installazione libreria python

Ora usiamo OpenSSl per generare la nostra chiave privata RSA e poi per estrarre la chiave pubblica, tramite i comandi:

- openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt rsa_keygen_bits:2048
- openssl rsa -pubout -in private_key.pem -out public_key.pem

Figura 3. Generazione chiave privata.

Figura 4. Estrazione chiave pubblica.

Creiamo il file: encdec.py che è un semplice programma python che tramite l'uso della libreria cryptography permette di cifrare e decifrare messaggi. Di seguito riportiamo lo snippet del codice e il suo risultato:

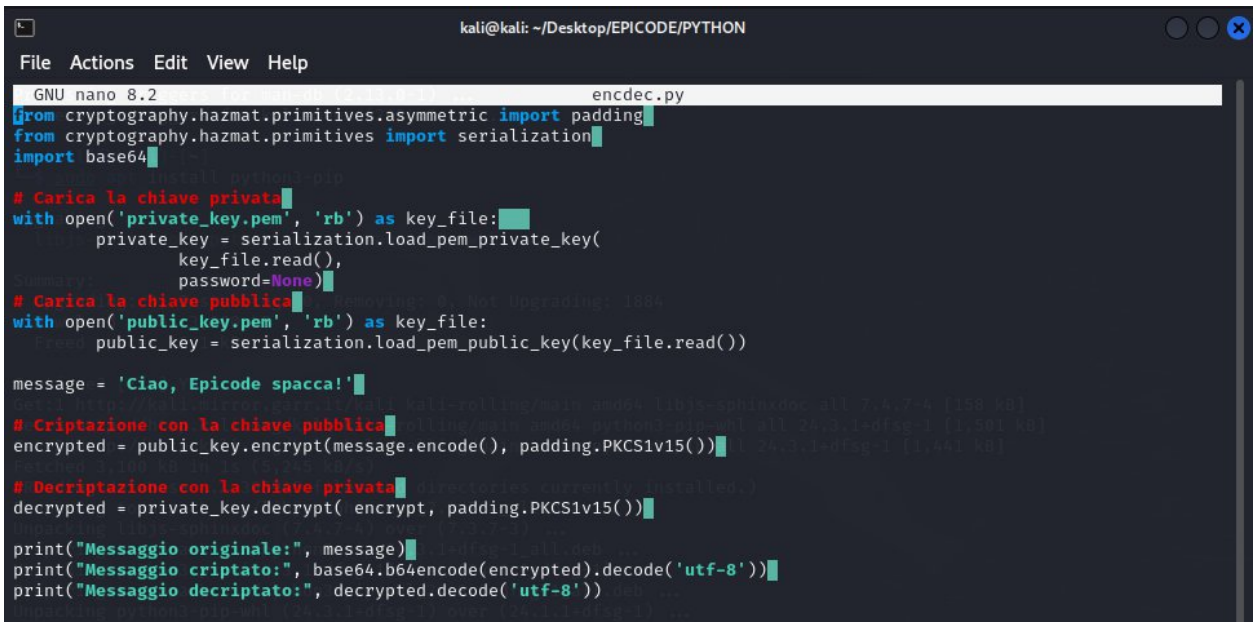

Figura 5. Snippet codice

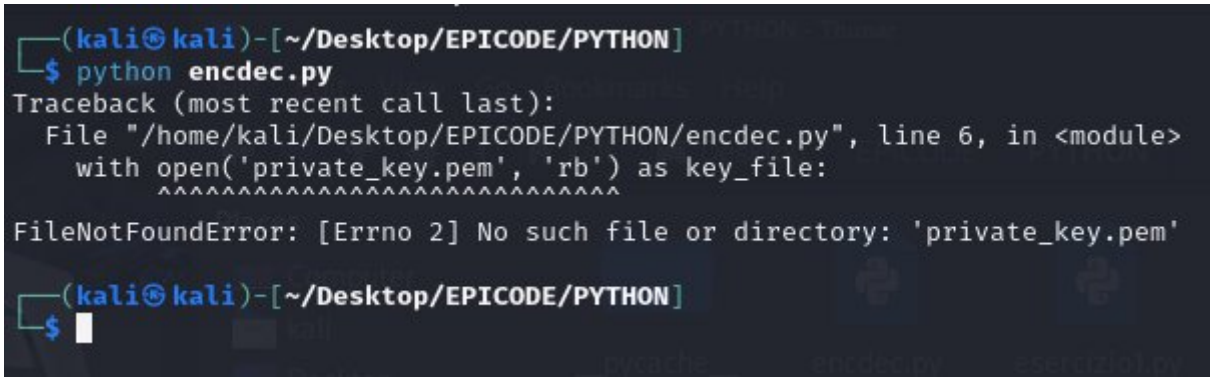

Figura 6. Primo risultato programma encdec.py

Notiamo che se avviamo il programma encdec.py subito dopo averlo scritto va in errore e ci fa notare che non riesce a trovare il file della chiave privata. Questo ci fa capire che il programma deve essere avviato nella cartella in cui queste chiavi sono state generate (cartella /home di default) oppure che bisogna inserire il path completo dei file delle chiavi all'interno del codice, così da poterlo eseguire in qualsiasi posizione. Correggiamo questo errore spostando, per semplicità, il file encdec.py nella cartella /home dove si trovano le chiavi pubblica e privata, riavviamo e vediamo il risultato:

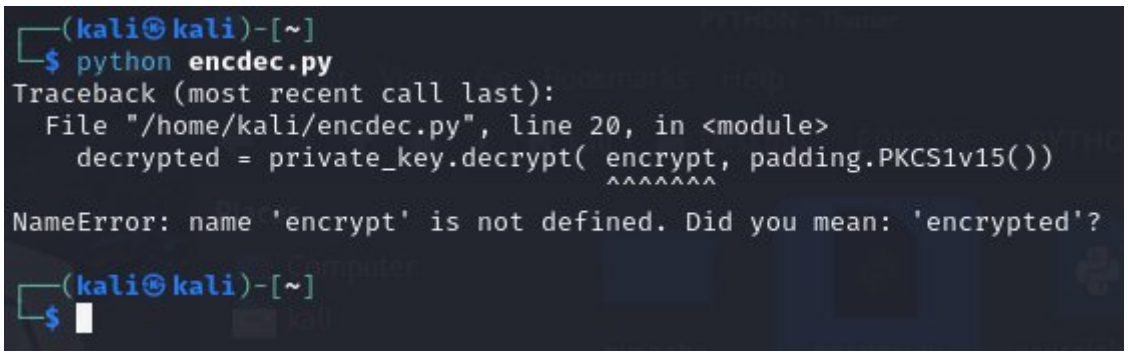

Figura 7. Nuovo avvio programma

Come suggerito nello screen in alto, c'è un errore nel codice che abbiamo trascritto dalle slide e dobbiamo correggere la parola encrypt con encrypted, quindi:

- **Codice errato**: decrypted = private_key.decrypt(**encrypt**, padding.PKCS1v15())
- **Codice corretto**: decrypted = private_key.decrypt(**encrypted**, padding.PKCS1v15())

Fatta questa correzione finalmente il programma funziona e nello screen seguente vediamo il risultato:

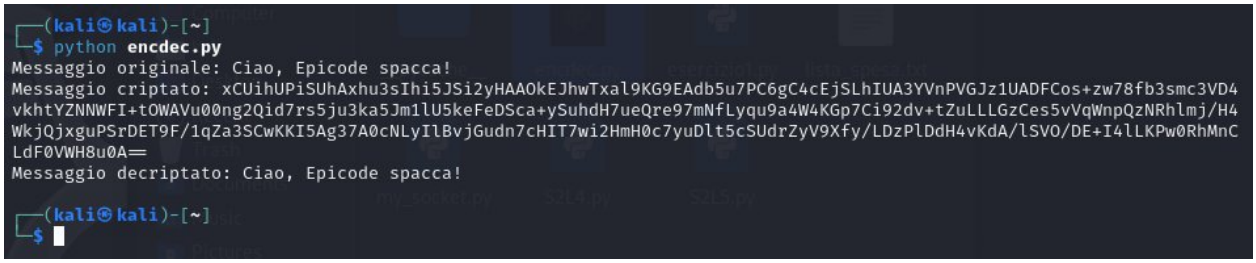

Figura 8. Programma funzionante

Per vedere la funzionalità del programma proviamo a decodificare in Base64 il messaggio criptato che ci è stato restituito dal programma ed otteniamo:

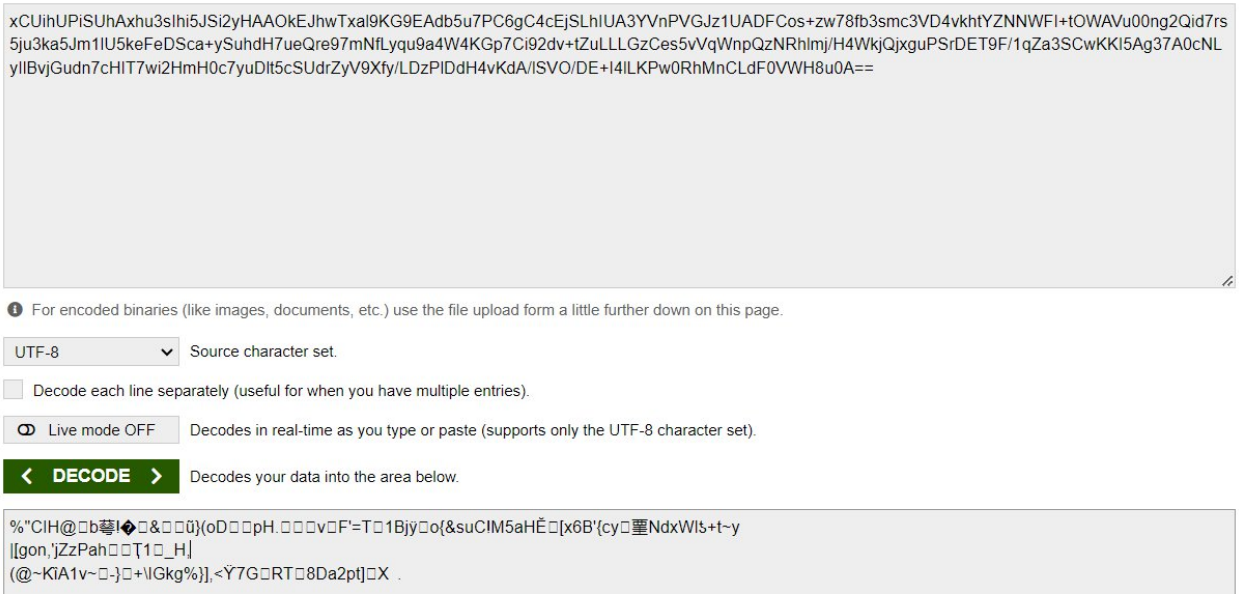

Figura 9. Decodifica messaggio cifrato

Questo risultato ci fa capire due cose in particolare, ovvero:

- **Come agisce il programma**: prima cripta con le chiavi che abbiamo generato e poi codifica il massaggio che può così essere trasmesso.
- **Importanza della criptazione**: se il messaggio non fosse stato criptato sarebbe stato semplice, una volta intercettato, decodificarlo e leggerlo in chiaro.