SECURITY REPORT

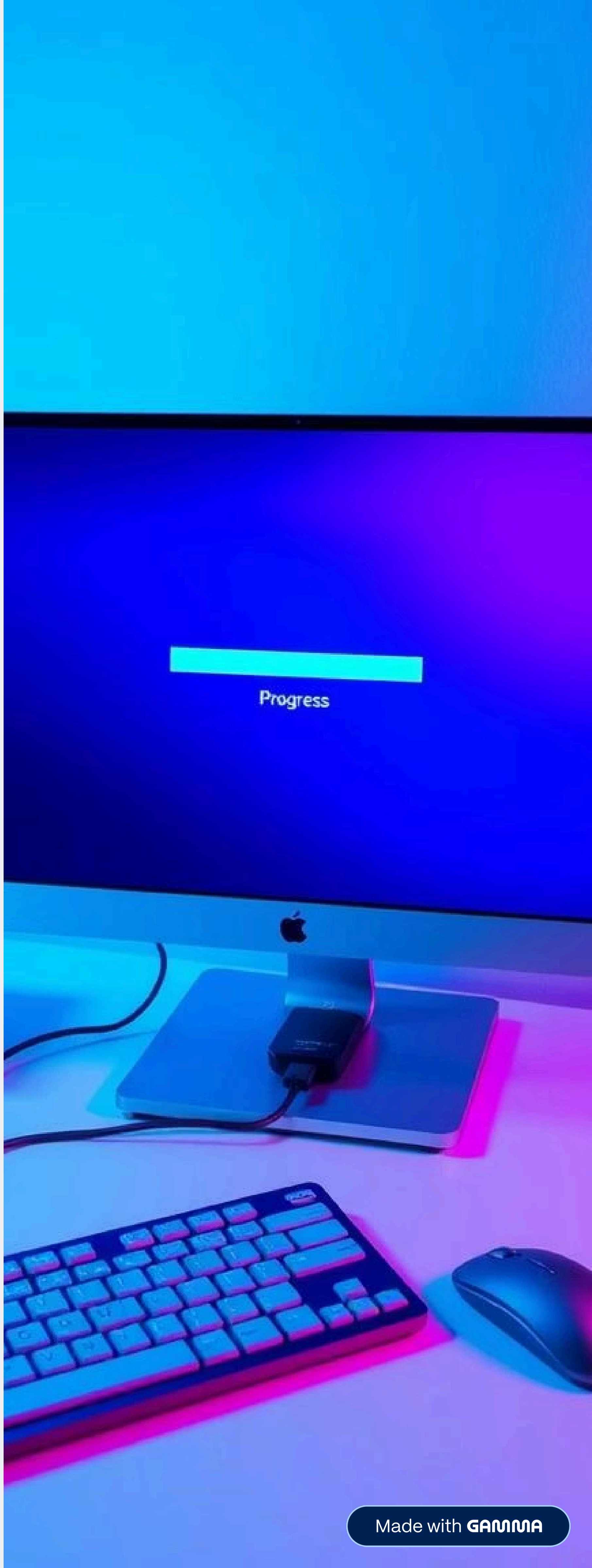
PRESENTED BY :

Yazan Aqtash

PRESENTED TO :

DR.SAMER KHASAWNEH

Hashemite University



Introduction

This project demonstrates three fundamental information security mechanisms using Python programming. Each mechanism represents a vital concept in the field of cryptography and secure communication. The user is prompted to select one of the three core functionalities:

- **A. Data Confidentiality via Symmetric Encryption (RC4)**
- **B. Message Authentication Assurance**
- **C. Data Confidentiality via Hybrid Encryption (Digital Envelope)**

A. Data Confidentiality via Symmetric Encryption (RC4)

This module uses the **RC4 stream cipher** to demonstrate symmetric encryption. RC4 is a simple, fast, and educationally effective algorithm for byte-wise data processing.

Process Overview:

- **Pseudorandom Generator Initialization:**
  The program initializes the key scheduling and pseudorandom generation phases of the RC4 algorithm to generate a keystream.
- **User Interaction:**
  The user chooses either encryption or decryption mode.
- **Encryption/Decryption Operation:**
  The input (plaintext or ciphertext) is processed **byte by byte**, where each byte is XORed with the corresponding byte in the keystream.
- **S-box Debug Output:**
  After processing each byte, the current state of the key matrix (S-box) is printed, providing a visual of the algorithm's internal transformations.

B. Message Authentication Assurance

This module implements **message authentication** to verify message integrity and authenticity, using **hashing combined with RSA encryption**.

Process Overview:

- **Hashing the Message:**
  A cryptographic hash function is used to generate a digest of the message.
- **Encrypting the Hash (MAC):**
  The digest is then encrypted using RSA-1024, producing a Message Authentication Code (MAC). This ensures the message originated from an authenticated sender and has not been tampered with.
- **Concatenating Message and MAC:**
  The program appends the MAC to the original message, preparing it for transmission.
- **Replay Attack Mitigation:**
  To prevent replay attacks, a **unique cryptographic parameter** (e.g., timestamp or nonce) is embedded in the message. This allows the receiver to detect duplicate or old messages.
- **Verification Mode:**
  Users can verify an incoming message with a precomputed MAC by decrypting the MAC and comparing it to a freshly hashed message digest.

C. Data Confidentiality Assurance by Hybrid Encryption (Digital Envelope)

This module uses **hybrid encryption** to provide a secure method of transmitting encrypted messages by combining symmetric and asymmetric techniques.

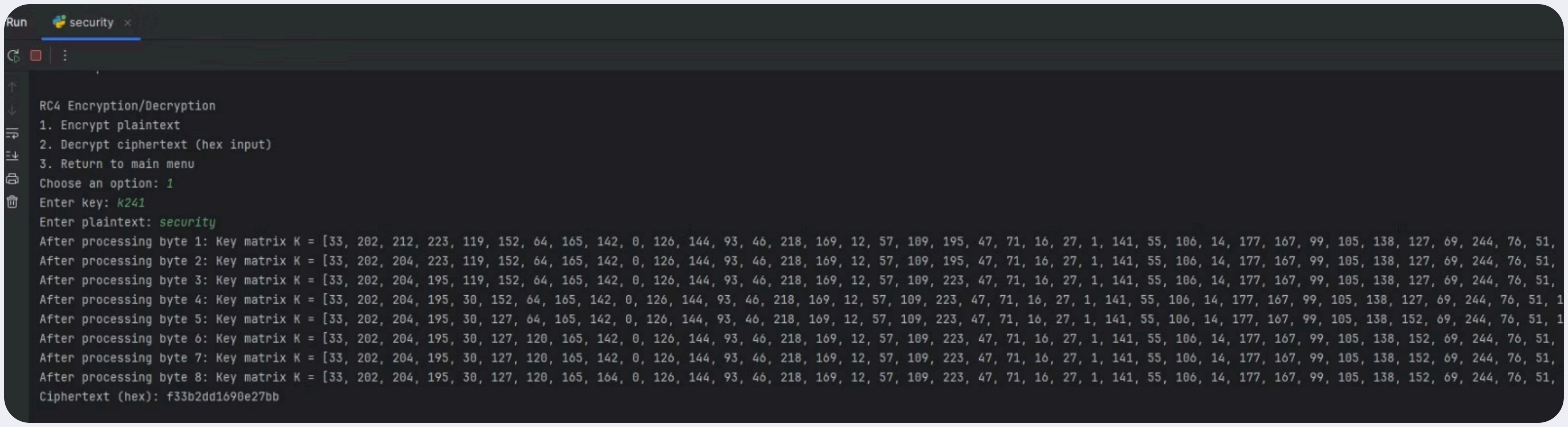
Process Overview:

- **Symmetric Encryption (DES-CTR):**
  The plaintext is encrypted using a randomly generated symmetric key with **DES in Counter (CTR) mode**, offering both confidentiality and efficiency.
- **Asymmetric Encryption (RSA-1024):**
  The symmetric key is encrypted using the receiver's RSA-1024 public key. This ensures that only the intended recipient can decrypt the key using their private RSA key.
- **Forming the Digital Envelope:**
  The final output is a digital envelope that consists of:
  - The encrypted message
  - The encrypted symmetric key
- **Decryption Process:**
  The reverse process is supported:
  - Decrypt the symmetric key using the RSA private key
  - Use the decrypted key to decrypt the DES-encrypted message and retrieve the original plaintext
- **Key Display:**
  All cryptographic keys generated during this process (symmetric and RSA key pairs) are displayed for transparency and educational insight.

Implementation & Testing

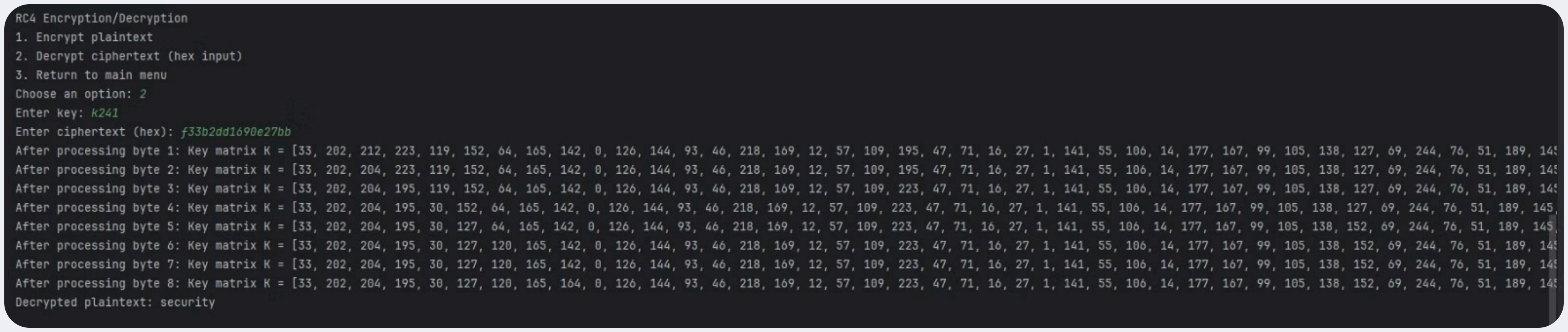
A. RC4

A.1 RC4 Encrypt



```
RC4 Encryption/Decryption
1. Encrypt plaintext
2. Decrypt ciphertext (hex input)
3. Return to main menu
Choose an option: 1
Enter key: k241
Enter plaintext: security
After processing byte 1: Key matrix K = [33, 202, 212, 223, 119, 152, 64, 165, 142, 0, 126, 144, 93, 46, 218, 169, 12, 57, 109, 195, 47, 71, 16, 27, 1, 141, 55, 106, 14, 177, 167, 99, 105, 138, 127, 69, 244, 76, 51,
After processing byte 2: Key matrix K = [33, 202, 204, 223, 119, 152, 64, 165, 142, 0, 126, 144, 93, 46, 218, 169, 12, 57, 109, 195, 47, 71, 16, 27, 1, 141, 55, 106, 14, 177, 167, 99, 105, 138, 127, 69, 244, 76, 51,
After processing byte 3: Key matrix K = [33, 202, 204, 195, 119, 152, 64, 165, 142, 0, 126, 144, 93, 46, 218, 169, 12, 57, 109, 223, 47, 71, 16, 27, 1, 141, 55, 106, 14, 177, 167, 99, 105, 138, 127, 69, 244, 76, 51,
After processing byte 4: Key matrix K = [33, 202, 204, 195, 30, 152, 64, 165, 142, 0, 126, 144, 93, 46, 218, 169, 12, 57, 109, 223, 47, 71, 16, 27, 1, 141, 55, 106, 14, 177, 167, 99, 105, 138, 127, 69, 244, 76, 51, 1
After processing byte 5: Key matrix K = [33, 202, 204, 195, 30, 127, 64, 165, 142, 0, 126, 144, 93, 46, 218, 169, 12, 57, 109, 223, 47, 71, 16, 27, 1, 141, 55, 106, 14, 177, 167, 99, 105, 138, 152, 69, 244, 76, 51, 1
After processing byte 6: Key matrix K = [33, 202, 204, 195, 30, 127, 120, 165, 142, 0, 126, 144, 93, 46, 218, 169, 12, 57, 109, 223, 47, 71, 16, 27, 1, 141, 55, 106, 14, 177, 167, 99, 105, 138, 152, 69, 244, 76, 51,
After processing byte 7: Key matrix K = [33, 202, 204, 195, 30, 127, 120, 165, 142, 0, 126, 144, 93, 46, 218, 169, 12, 57, 109, 223, 47, 71, 16, 27, 1, 141, 55, 106, 14, 177, 167, 99, 105, 138, 152, 69, 244, 76, 51,
After processing byte 8: Key matrix K = [33, 202, 204, 195, 30, 127, 120, 165, 164, 0, 126, 144, 93, 46, 218, 169, 12, 57, 109, 223, 47, 71, 16, 27, 1, 141, 55, 106, 14, 177, 167, 99, 105, 138, 152, 69, 244, 76, 51,
Ciphertext (hex): f33b2dd1690e27bb
```

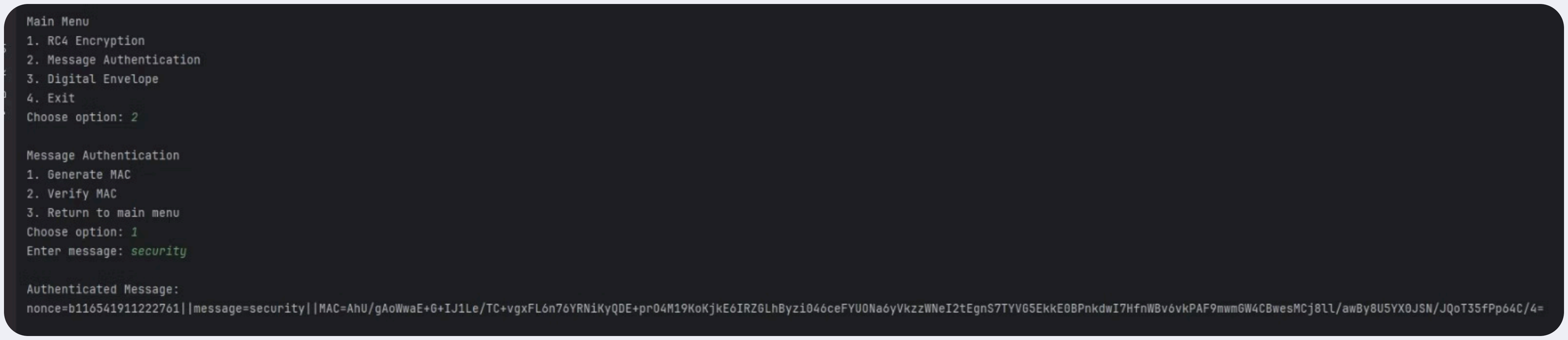
A.2 RC4 Decrypt



```
RC4 Encryption/Decryption
1. Encrypt plaintext
2. Decrypt ciphertext (hex input)
3. Return to main menu
Choose an option: 2
Enter key: k241
Enter ciphertext (hex): f33b2dd1690e27bb
After processing byte 1: Key matrix K = [33, 202, 212, 223, 119, 152, 64, 165, 142, 0, 126, 144, 93, 46, 218, 169, 12, 57, 109, 195, 47, 71, 16, 27, 1, 141, 55, 106, 14, 177, 167, 99, 105, 138, 127, 69, 244, 76, 51, 189, 14
After processing byte 2: Key matrix K = [33, 202, 204, 223, 119, 152, 64, 165, 142, 0, 126, 144, 93, 46, 218, 169, 12, 57, 109, 195, 47, 71, 16, 27, 1, 141, 55, 106, 14, 177, 167, 99, 105, 138, 127, 69, 244, 76, 51, 189, 14
After processing byte 3: Key matrix K = [33, 202, 204, 195, 119, 152, 64, 165, 142, 0, 126, 144, 93, 46, 218, 169, 12, 57, 109, 223, 47, 71, 16, 27, 1, 141, 55, 106, 14, 177, 167, 99, 105, 138, 127, 69, 244, 76, 51, 189, 14
After processing byte 4: Key matrix K = [33, 202, 204, 195, 30, 152, 64, 165, 142, 0, 126, 144, 93, 46, 218, 169, 12, 57, 109, 223, 47, 71, 16, 27, 1, 141, 55, 106, 14, 177, 167, 99, 105, 138, 127, 69, 244, 76, 51, 189, 145,
After processing byte 5: Key matrix K = [33, 202, 204, 195, 30, 127, 64, 165, 142, 0, 126, 144, 93, 46, 218, 169, 12, 57, 109, 223, 47, 71, 16, 27, 1, 141, 55, 106, 14, 177, 167, 99, 105, 138, 152, 69, 244, 76, 51, 189, 145,
After processing byte 6: Key matrix K = [33, 202, 204, 195, 30, 127, 120, 165, 142, 0, 126, 144, 93, 46, 218, 169, 12, 57, 109, 223, 47, 71, 16, 27, 1, 141, 55, 106, 14, 177, 167, 99, 105, 138, 152, 69, 244, 76, 51, 189, 14
After processing byte 7: Key matrix K = [33, 202, 204, 195, 30, 127, 120, 165, 142, 0, 126, 144, 93, 46, 218, 169, 12, 57, 109, 223, 47, 71, 16, 27, 1, 141, 55, 106, 14, 177, 167, 99, 105, 138, 152, 69, 244, 76, 51, 189, 14
After processing byte 8: Key matrix K = [33, 202, 204, 195, 30, 127, 120, 165, 164, 0, 126, 144, 93, 46, 218, 169, 12, 57, 109, 223, 47, 71, 16, 27, 1, 141, 55, 106, 14, 177, 167, 99, 105, 138, 152, 69, 244, 76, 51, 189, 14
Decrypted plaintext: security
```

B. Message Authentication Assurance

B.1 Message Authentication (Generate MAC)

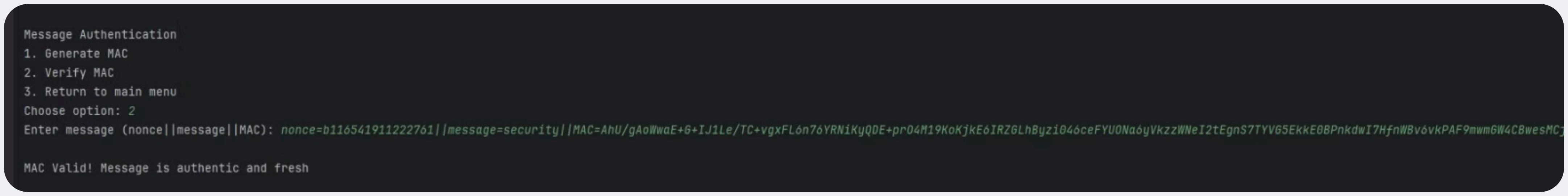


```
Main Menu
1. RC4 Encryption
2. Message Authentication
3. Digital Envelope
4. Exit
Choose option: 2

Message Authentication
1. Generate MAC
2. Verify MAC
3. Return to main menu
Choose option: 1
Enter message: security

Authenticated Message:
nonce=b116541911222761||message=security||MAC=AhU/gAoWwaE+G+IJ1Le/TC+vgxFL6n76YRNiKyQDE+pr04M19KoKjkE6IRZGLhByzi046ceFYUONa6yVkzzWNeI2tEgnS7TYVG5EkkE0BPnkdwI7HfnWBv6vkPAF9mwmGW4CBwesMCj8ll/awBy8U5YX0JSN/JQoT35fPp64C/4=
```

B.2 Message Authentication (Verify MAC)

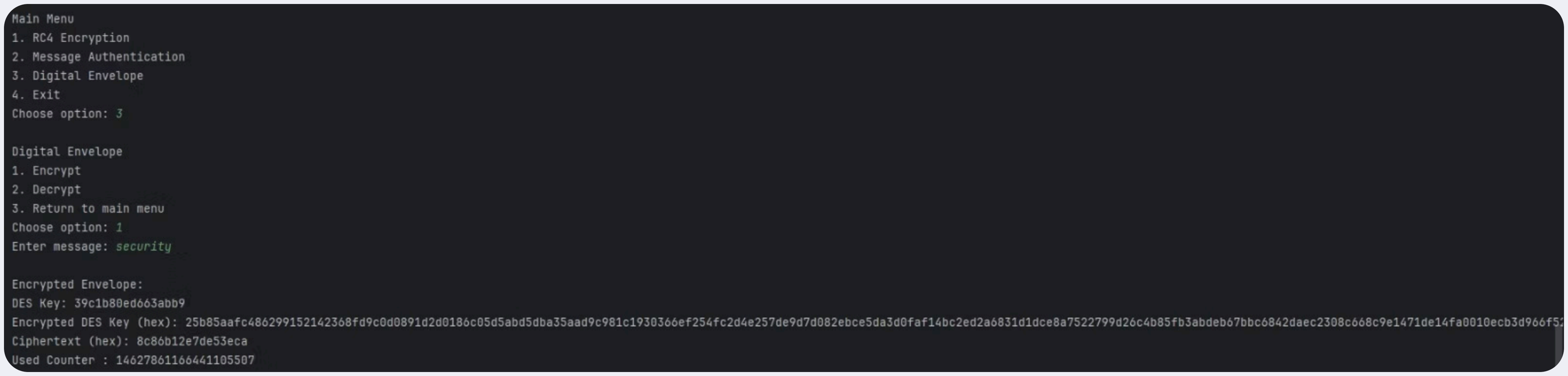


```
Message Authentication
1. Generate MAC
2. Verify MAC
3. Return to main menu
Choose option: 2
Enter message (nonce||message||MAC): nonce=b116541911222761||message=security||MAC=AhU/gAoWwaE+G+IJ1Le/TC+vgxFL6n76YRNiKyQDE+pr04M19KoKjkE6IRZGLhByzi046ceFYUONa6yVkzzWNeI2tEgnS7TYVG5EkkE0BPnkdwI7HfnWBv6vkPAF9mwmGW4CBwesMCj

MAC Valid! Message is authentic and fresh
```

C. Digital Envelope

C.1 Digital Envelope Encrypt

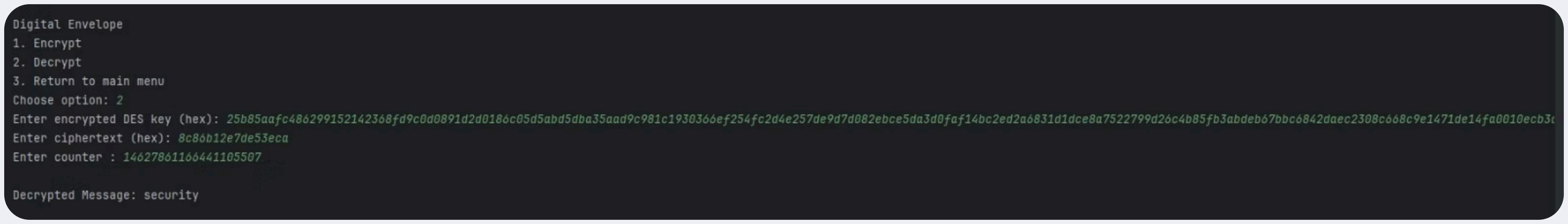


```
Main Menu
1. RC4 Encryption
2. Message Authentication
3. Digital Envelope
4. Exit
Choose option: 3

Digital Envelope
1. Encrypt
2. Decrypt
3. Return to main menu
Choose option: 1
Enter message: security

Encrypted Envelope:
DES Key: 39c1b80ed663abb9
Encrypted DES Key (hex): 25b85aafc486299152142368fd9c0d0891d2d0186c05d5abd5dba35aad9c981c1930366ef254fc2d4e257de9d7d082ebce5da3d0faf14bc2ed2a6831d1dce8a7522799d26c4b85fb3abdeb67bbc6842daec2308c668c9e1471de14fa0010ecb3d966f52
Ciphertext (hex): 8c86b12e7de53eca
Used Counter : 14627861166441105507
```

C.2 Digital Envelope Decrypt



```
Digital Envelope
1. Encrypt
2. Decrypt
3. Return to main menu
Choose option: 2
Enter encrypted DES key (hex): 25b85aafc486299152142368fd9c0d0891d2d0186c05d5abd5dba35aad9c981c1930366ef254fc2d4e257de9d7d082ebce5da3d0faf14bc2ed2a6831d1dce8a7522799d26c4b85fb3abdeb67bbc6842daec2308c668c9e1471de14fa0010ecb3
Enter ciphertext (hex): 8c86b12e7de53eca
Enter counter : 14627861166441105507

Decrypted Message: security
```