UNIVERSITI TEKNOLOGI MARA

COLLEGE OF COMPUTING, INFORMATICS AND MATHEMATICS

CDCS230: BACHELOR OF COMPUTER SCIENCE (HONS.)

CSC662: COMPUTER SECURITY

GROUP PROJECT: SHIFT CIPHER ALGORITHM

NO.	NAME	STUDENT ID
1.	AMIR ZAKARIAH	2023500147
2.	MUHAMMAD AFIQ BIN MOHD ASRI	2023185115
3.	MUHAMMAD FAIZ HAKIMI BIN MUHD SHEPUAN	2023184661
4.	MUHAMMAD IQBAL NAJMUDDIN BIN IDI AMIN	2023513853

PREPARED FOR:

DR. FERAS ZEN ALDEN

TABLE OF CONTENTS

PROJECT SCOPE

The project involves designing and improving a cryptographic system based on the shift cipher algorithm. The tasks include developing a functional implementation of the basic shift cipher, enhancing the algorithm to improve its security and usability, and documenting the process in a comprehensive report. The final deliverable will consist of a working system prototype, an improved shift cipher algorithm, and a report detailing the system design, algorithm implementation, testing, and analysis of improvements.

PROJECT GOALS

1. **Understand the Shift Cipher**:
    - Research and analyze the working principles, strengths, and weaknesses of the classical shift cipher algorithm.
2. **Develop the System**:
    - Create a prototype system that demonstrates the basic functionality of the shift cipher for encryption and decryption of text.
3. **Improve the Algorithm**:
    - Enhance the shift cipher algorithm to address its vulnerabilities (e.g., susceptibility to brute force attacks and frequency analysis).
    - Introduce additional features such as dynamic shifts, key integration, or multi-layer encryption.
4. **Validate and Test**:
    - Test the system with various datasets to ensure reliability, efficiency, and improved security.
    - Compare the performance and security of the basic and improved algorithms.
5. **Document the Process**:

- Create a detailed report covering the system design, algorithm logic, implementation, improvements, testing results, and overall project analysis.
- Provide recommendations for further enhancements or alternative cryptographic approaches.

1. WHAT IS A SHIFT CIPHER?

A shift cipher is an encryption technique which involves replacing each letter in the message by a letter that is some fixed number of positions further along in the alphabet. The fixed number of positions that determine the encryption key is called the encryption key. It is just the length of the shift we are using. For example, upon encrypting the message “cookie” using a shift cipher with encryption key 3, the encoded message or what called as ciphertext will be “FRRNLH”

2. HOW DOES SHIFT CIPHER WORKS?

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**
0	1	2	3	4	5	6	7	8	9	10	11	12
N	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
13	14	15	16	17	18	19	20	21	22	23	24	25

Table 1 Initial letter and its index

Table 1 represents the initial representation of the letter of each index. As stated before the example taken is the word ‘cookie’ which represented by the index key 2,4,8,10, and 14 as displayed in the *Table 2*

Letter	C	O	O	K	I	E
Key	2	14	14	10	8	4

Table 2 Original message with initial key

The example in this case use the value ‘3’ as the encryption key in which will change the message key being encrypt as shown in *Table 3*

Letter	F	R	R	N	L	H
Key	2+3 = **5**	14+3 = **17**	14+3 = **17**	10+3 = **13**	8+3 = **11**	4+3 = **7**

Table 3 Ciphertext when encryption key = 3

3. CODING IMPLEMENTATION OF BASIC SHIFT CIPHER

The below Python code demonstrates a basic implementation of the Shift Cipher algorithm:

```
alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

def create_substitution_table(shift):
    substitution_table = str.maketrans(alphabet,
alphabet[shift:] + alphabet[:shift])
    return substitution_table

def encrypt(plaintext, substitution_table):
    return plaintext.translate(substitution_table)

def decrypt(ciphertext, substitution_table):
    return ciphertext.translate(substitution_table)

plaintext = input("\nEnter plaintext: ")
shift = int(input("\nEnter shift: "))

substitution_table = create_substitution_table(shift)
```

```
ciphertext = encrypt(plaintext, substitution_table)

print(f'\nPlaintext: {plaintext}')
print(f'\nCiphertext: {ciphertext}')

substitution_table = create_substitution_table(-shift)
plaintext = decrypt(ciphertext, substitution_table)

print(f'\nDecrypted text: {plaintext}\n')
```

Below is a step-by-step explanation of how the code works:

3.1 Defining the Alphabet

```
alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

- The `alphabet` variable contains the characters used in the cipher. For this implementation, the alphabet is uppercase English letters (A-Z).

3.2 Creating the Substitution Table

```
def create_substitution_table(shift):
    substitution_table = str.maketrans(alphabet,
alphabet[shift:] + alphabet[:shift])
    return substitution_table
```

- This function creates a substitution table, which maps each letter of the alphabet to a new letter based on the specified `shift`.
- `alphabet[shift:] + alphabet[:shift]` shifts the characters cyclically:
    - For example, with a shift of `3`, the alphabet becomes "`DEFGHIJKLMNOPQRSTUVWXYZABC`".
- `str.maketrans()` generates the mapping between the original alphabet and the shifted version.

3.3 Encrypting the Plaintext

```
def encrypt(plaintext, substitution_table):
    return plaintext.translate(substitution_table)
```

- The `encrypt` function uses the substitution table to replace each character in the `plaintext` with its corresponding character from the shifted alphabet.
- The `translate()` method applies the substitution table to the input text.

3.4 Decrypting the Ciphertext

```
def decrypt(ciphertext, substitution_table):
    return ciphertext.translate(substitution_table)
```

- The `decrypt` function is similar to the `encrypt` function. It uses the substitution table created with the negative shift (`-shift`) to reverse the encryption process and restore the original `plaintext`.

3.5 Taking User Input

```
plaintext = input("\nEnter plaintext: ")
shift = int(input("\nEnter shift: "))
```

- The user is prompted to enter the `plaintext` (text to be encrypted) and the `shift` (number of positions to shift the alphabet).

3.6 Encryption Process

```
substitution_table = create_substitution_table(shift)
ciphertext = encrypt(plaintext, substitution_table)
```

```
print(f'\nPlaintext: {plaintext}')
print(f'\nCiphertext: {ciphertext}')
```

- A substitution table is created based on the user-provided `shift`.
- The `encrypt` function is called to generate the `ciphertext`, which is the encrypted version of the `plaintext`.
- Both the original `plaintext` and the resulting `ciphertext` are displayed.

3.7 Decryption Process

```
substitution_table = create_substitution_table(-shift)
plaintext = decrypt(ciphertext, substitution_table)

print(f'\nDecrypted text: {plaintext}\n')
```

- A new substitution table is created using the negative shift (`-shift`).
- The `decrypt` function is called to reverse the encryption process and retrieve the original `plaintext`.
- The decrypted text is displayed, demonstrating that the algorithm successfully restores the original input.

3.8 Summary of the Algorithm's Flow

1. The user inputs the plaintext and shift value.
2. The encryption process shifts the alphabet and substitutes the plaintext characters using the substitution table.
3. The decryption process reverses the shift and substitutes the ciphertext characters back to their original values.

3.9 Example of Encryption and Decryption Processes

Example 1:

```
Enter plaintext: COMPUTER SECURITY

Enter shift: 3

Plaintext: COMPUTER SECURITY

Ciphertext: FRPSXWHU VHFXULWB

Decrypted text: COMPUTER SECURITY
```

Example 2:

```
Enter plaintext: SHIFT CIPHER ALGORITHM

Enter shift: 14

Plaintext: SHIFT CIPHER ALGORITHM

Ciphertext: GVWTH QWDVSF OZUCFWHVA

Decrypted text: SHIFT CIPHER ALGORITHM
```

Example 3:

```
Enter plaintext: LOREM IPSUM DOLOR SIT AMET

Enter shift: 21

Plaintext: LOREM IPSUM DOLOR SIT AMET

Ciphertext: GJMZH DKNPH YJGJM NDO VHZO

Decrypted text: LOREM IPSUM DOLOR SIT AMET
```

4. STRENGTHS AND WEAKNESSES OF SHIFT CIPHER

4.1 Strengths

- Simplicity and Ease of Implementation

  One of the primary strengths of the shift cipher is its simplicity. As has been explained, the algorithm operates by shifting letters in the plaintext by a fixed number of positions in the alphabet, making it very straightforward to understand and implement. This characteristic makes it an excellent educational tool for beginners learning about encryption methods. The ease of use also extends to its implementation; it can be executed with minimal computational resources, requiring only basic arithmetic operations to encrypt and decrypt messages. This accessibility allows individuals without extensive cryptographic knowledge to grasp fundamental concepts of encryption quickly.

- Flexibility in Modifications

  Another strength of the shift cipher is its flexibility. While the basic version uses a single fixed shift, it can be easily modified to enhance security. For instance, variations such as using multiple shifts or incorporating keywords can create more complex encryption schemes. These modifications can increase the number of potential keys significantly beyond the basic 25 (for a standard English alphabet), thereby improving security against brute-force attacks. Such adaptability allows users to tailor the cipher to their specific needs while still retaining its foundational simplicity.

4.2 Weaknesses

- Vulnerability to Brute-Force Attacks

Despite its strengths, the shift cipher has significant weaknesses, particularly its vulnerability to brute-force attacks. With only 25 possible keys or shifts (for a standard English alphabet), an attacker can systematically try each one until they find the correct shift value. This makes the cipher easy to crack, especially with modern computational capabilities that allow for rapid testing of all possible keys. As a result, the shift cipher is not suitable for securing sensitive information in modern applications where stronger encryption methods are required.

- Lack of Security Against Frequency Analysis

  Another critical weakness lies in its vulnerability to frequency analysis. Since each letter in the plaintext is consistently replaced by another letter based on a fixed shift, patterns in letter frequency remain intact in the ciphertext. Attackers can exploit these patterns by analyzing how often certain letters appear, which is particularly effective against longer texts where letter distribution is more clear. This vulnerability compromises the confidentiality of messages encrypted with a shift cipher, making it ineffective for serious cryptographic purposes.

5. COMBINING SHIFT CIPHER WITH RSA ENCRYPTION

To address the weakness and strengthen the shift cipher, RSA encryption was adopted as an additional encryption method put on top of the shift cipher. RSA is an asymmetric encryption algorithm that uses a public-private key pair to encrypt and decrypt data. The encryption process follows these steps:

1. **Shift Cipher encryption -** The plaintext submitted is first encrypted with the selected shift value.
2. **RSA encryption -** The ciphertext is further encrypted using RSA.
3. **Decryption -** The ciphertext is first decrypted using RSA, followed by the shift cipher decryption.

This dual encryption approach provides another layer of security atop the existing shift cipher algorithm, mitigating the inherent weakness of the shift cipher algorithm.

6. IMPLEMENTATION AND TESTING

6.1 Coding Implementation

The system was implemented in JavaScript and utilizes Web Cryptography APIs for RSA encryption. The key implementation details are:

- Shift Cipher Encryption

```
// Step 1: Shift Cipher encryption
String substitutionTable = createSubstitutionTable(shift);
String shiftEncryptedText = shiftCipherEncrypt(plaintext.toUpperCase(), substitutionTable);
```

The plaintext inserted by the user will be encrypted by the method `shiftCipherEncrypt()`, which takes two arguments, the plaintext and the substitution table. In this method, the shift cipher encryption will be executed and return a ciphertext string.

- RSA encryption

```
// Step 2: RSA encryption
// Generate RSA key pair
KeyPair keyPair = generateKeyPair();
PublicKey publicKey = keyPair.getPublic();
PrivateKey privateKey = keyPair.getPrivate();

// Encrypt the Shift Cipher encrypted text using RSA public key
String rsaEncryptedText = rsaEncrypt(shiftEncryptedText, publicKey);

System.out.println("\nEncrypted (RSA + Shift Cipher) Message: " + rsaEncryptedText);
```

The ciphertext generated from the encryption of shift cipher will be used by `rsaEncrypt()` method, which takes the ciphertext and the generated public key. A ciphertext from RSA encryption will be generated.

- RSA and Shift Cipher decryption

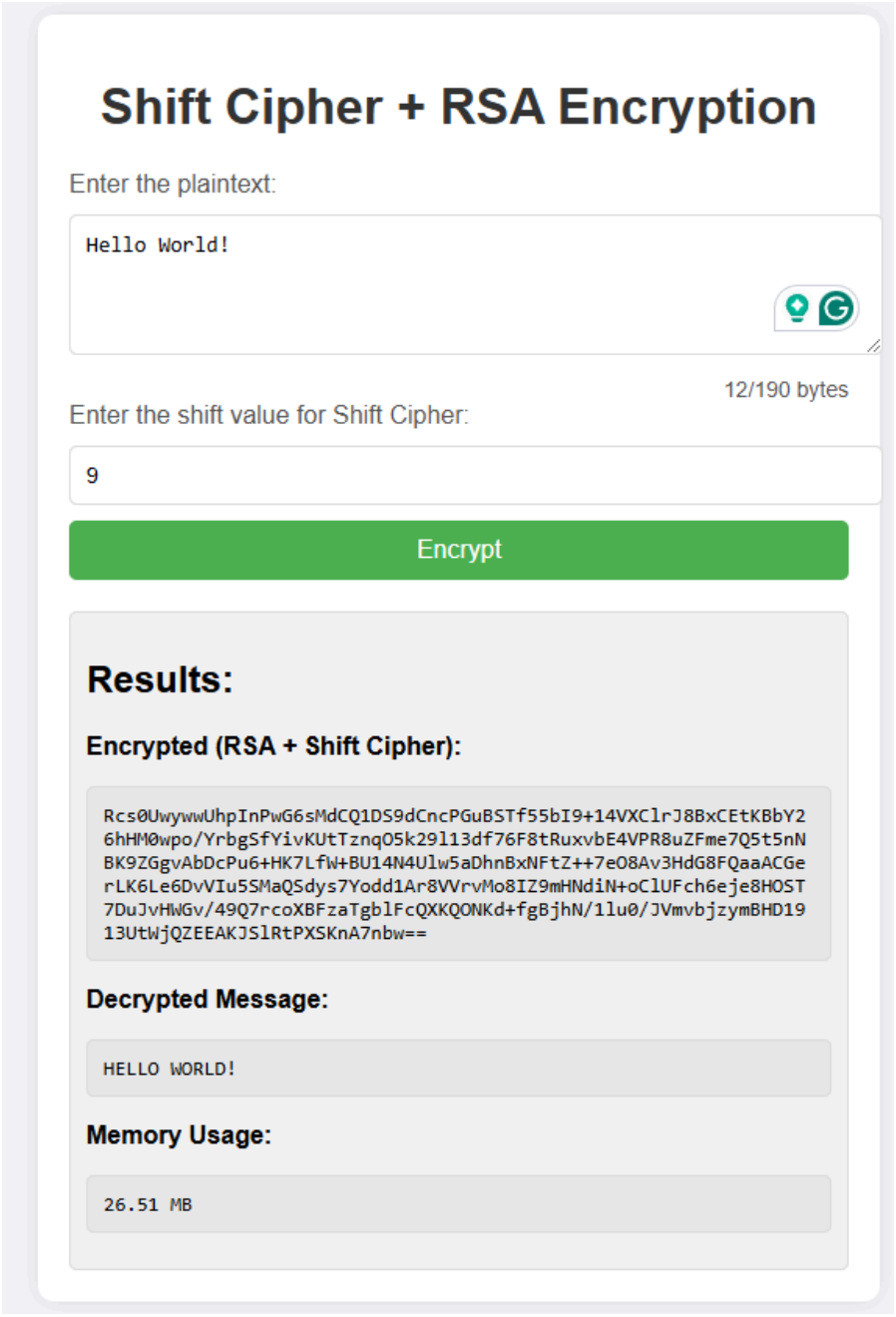
```
// Step 3: Decrypt the message

// Decrypt the RSA encrypted message using the RSA private key
String rsaDecryptedText = rsaDecrypt(rsaEncryptedText, privateKey);

// Decrypt the result using Shift Cipher
String finalDecryptedText = shiftCipherDecrypt(rsaDecryptedText, substitutionTable);

System.out.println("\nDecrypted Message: " + finalDecryptedText);
```

For decrypting the RSA + Shift Cipher ciphertext, `rsaDecrypt()` method will be called first and generate a decrypted plaintext. This method takes the ciphertext and private key for decryption. The decrypted plaintext will then be used by the `shiftCipherDecrypt()` method, which takes the decrypted plaintext from `rsaDecrypt()` and the substitution table to decrypt and generate the final decrypted plaintext.

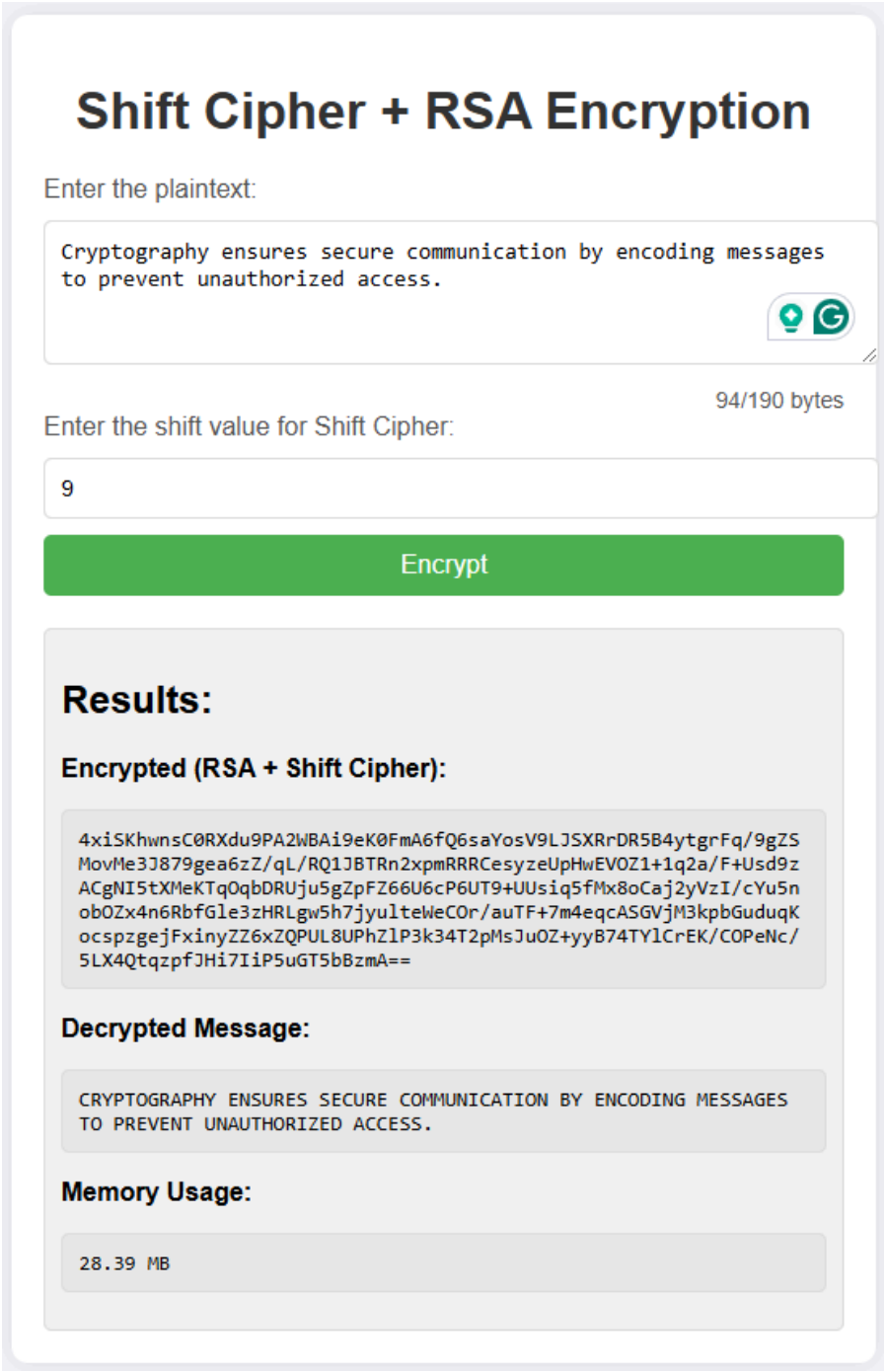
6.2 Testing

Three different test scenarios were conducted to validate the system's encryption strength and reliability. All the test cases have a constraint of not exceeding more than 190 bytes due to the limitation of RSA encryption ability.

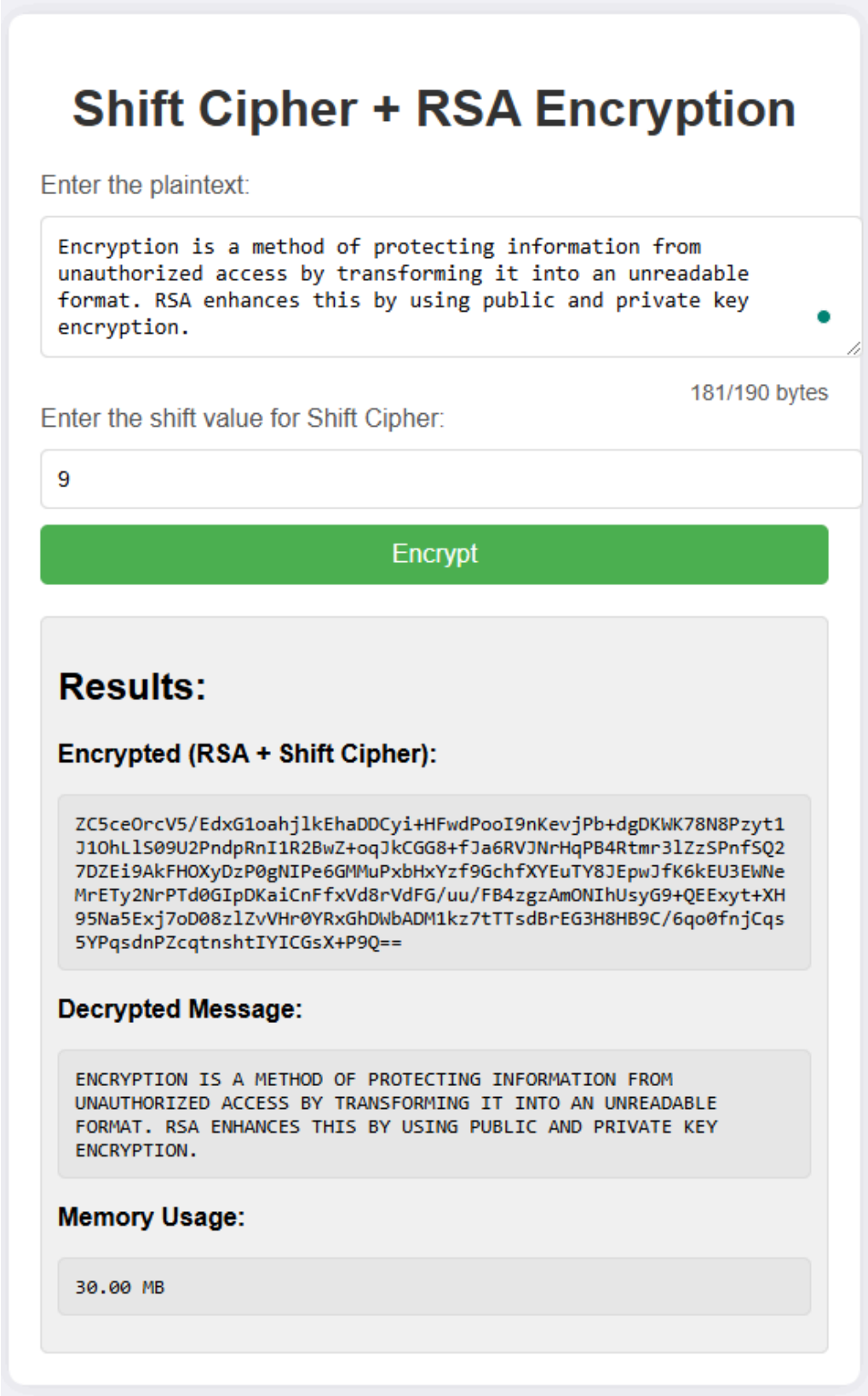
1. Short text


Shift Cipher + RSA Encryption
Enter the plaintext:
Hello World!
12/190 bytes
Enter the shift value for Shift Cipher:
9
Encrypt
Results:
Encrypted (RSA + Shift Cipher):
Rcs0UwywwUhpInPwG6sMdCQ1DS9dCncPGuBSTf55bI9+14VXClrJ8BxCEtKBbY2
6hHM0wpo/YrbgSfYivKUtTznqO5k29l13df76F8tRuxvbE4VPR8uZFme7Q5t5nN
BK9ZGgvAbDcPu6+HK7LfW+BU14N4Ulw5aDhnBxNFtZ++7eO8Av3HdG8FQaaACGe
rLK6Le6DvVIu5SMaQSdys7Yodd1Ar8VVrvMo8IZ9mHNdiN+oClUFch6eje8HOST
7DuJvHWGv/49Q7rcoXBFzaTgblFcQXKQONKd+fgBjhN/1lu0/JVmvbjzymBHD19
13UtWjQZEEAKJSlRtPXSKnA7nbw==
Decrypted Message:
HELLO WORLD!
Memory Usage:
26.51 MB

2. Medium text



3. Long text


Shift Cipher + RSA Encryption
Enter the plaintext:
Encryption is a method of protecting information from unauthorized access by transforming it into an unreadable format. RSA enhances this by using public and private key encryption.
181/190 bytes
Enter the shift value for Shift Cipher:
9
Encrypt
Results:
Encrypted (RSA + Shift Cipher):
ZC5ceOrcV5/EdxG1oahjlkEhaDDCyi+HFwdPooI9nKevjPb+dgDKWK78N8Pzyt1
J1OhLlS09U2PndpRnI1R2BwZ+oqJkCGG8+fJa6RVJNrHqPB4Rtmr3lZzSPnfSQ2
7DZEi9AkFHOXyDzP0gNIPe6GMMuPxbHxYzf9GchfXYEuTY8JEpwJfK6kEU3EWNe
MrETy2NrPTd0GIpDKaiCnFfxVd8rVdFG/uu/FB4zgzAmONIhUsyG9+QEExyt+XH
95Na5Exj7oD08zlZvVHr0YRxGhDWbADM1kz7tTTsdBrEG3H8HB9C/6qo0fnjCqs
5YPqsdnPZcqtnshtIYICGsX+P9Q==
Decrypted Message:
ENCRYPTION IS A METHOD OF PROTECTING INFORMATION FROM UNAUTHORIZED ACCESS BY TRANSFORMING IT INTO AN UNREADABLE FORMAT. RSA ENHANCES THIS BY USING PUBLIC AND PRIVATE KEY ENCRYPTION.
Memory Usage:
30.00 MB

As shown by the tests conducted, all plaintext encryption and decryption work as expected. A slight increase in memory usage as the text size increases is also within the expected parameter. The decrypted message for all encrypted plaintext is displayed correctly and retains its original plaintext.

7. HOW THIS IMPROVES SHIFT CIPHER

If used on its own, shift cipher is vulnerable to brute force attacks and frequency analysis due to its nature of shifting with a set value and having set patterns when encrypting since it only shifts the letters but does not change or distribute the letters during encryption. This can make it fairly easy for a computer or even a human to decipher the message that has been encrypted using the shift cipher. The simple nature of the shift cipher itself makes it very easy to decrypt, even without the usage of a computer.

However, when used in conjunction with the RSA encryption, the previously mentioned weaknesses become obsolete. RSA on its own is capable of withstanding a brute force attack, since the generation of the public and private keys depend heavily on the difficulty of factoring the product of two typically large prime numbers. If the key size increases, the cost in computation size will also increase, therefore even though a brute force on a message encrypted by the RSA is possible, it will take an unreasonable amount of time to do so, since bigger keys leads to more possibilities and the current technology nowadays, even supercomputers, are still incapable of performing brute force attacks on the RSA encryption within a reasonable time frame.

In terms of frequency analysis, RSA will not be affected by it either since after encryption, there will be no patterns that relate with the original message. RSA encryption operates on blocks of data, which will be converted into ciphertext using a modular exponentiation. If frequency analysis was to be done on an RSA encrypted message, it would be impossible to detect a pattern from the coded message, making it impossible to deduce what the message is by checking the frequency.

8. CONCLUSION

The shift cipher, while an excellent introductory tool for understanding cryptographic principles, has notable vulnerabilities. Its small key space makes it highly susceptible to brute force attacks, and its preservation of plaintext letter frequencies leaves it exposed to frequency analysis. These limitations highlight its unsuitability for modern applications requiring robust data security. However, enhancements such as dynamic shifts and multi-layered encryption can improve its resistance to attacks, albeit marginally.

When combined with RSA encryption, the inherent weaknesses of the shift cipher are effectively mitigated. RSA's reliance on the computational difficulty of prime factorization makes it resistant to brute force attacks, even with current technological capabilities. Additionally, RSA eliminates patterns in the ciphertext, nullifying the effectiveness of frequency analysis. This dual-encryption approach demonstrates how classical ciphers can be bolstered with modern cryptographic techniques, significantly enhancing overall security.

REFERENCES

1. https://devcodef1.com/news/1323649/shift-cipher-implementation
2. https://www.101computing.net/caesar-shift-substitution-cipher/
3. https://crypto.interactive-maths.com/caesar-shift-cipher.html
4. https://math.asu.edu/sites/default/files/shift.pdf
5. https://en.wikipedia.org/wiki/RSA_(cryptosystem)
6. https://www.geeksforgeeks.org/rsa-algorithm-cryptography/
7. https://www.techtarget.com/searchsecurity/definition/RSA
8. https://www.securew2.com/blog/what-is-rsa-asymmetric-encryption