Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Spring, Year: 2025), B.Sc. in CSE (Day)

Advanced Encryption - Decryption

Course Title: Cyber Security
Course Code: 323
Section: 222-D4

Students Details

Name	ID
Tasdid Rahman Khan	222002029
Md. Syful Islam	222002111
Sazzad Hossain	221002464

Submission Date: 01-05-2025
Course Teacher's Name: Md. Sabbir Hosen Mamun

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Overview

This project presents a web-based Message Encryption Tool that implements three distinct cryptographic algorithms: AES (Advanced Encryption Standard), RSA (Rivest-Shamir-Adleman), and Caesar Cipher. The tool provides a user-friendly interface for encrypting and decrypting messages using these algorithms, catering to different security needs and educational purposes.

1.2 Motivation

This project addresses the need for understanding different encryption methodologies while providing a practical tool for basic message security. The implementation demonstrates both modern cryptographic techniques (AES, RSA) and historical ciphers (Caesar), offering comparative insights into their operation and use cases.

1.3 Problem Definition

1.3.1 Problem Statement

The problem is to create a web-based tool that enables users to encrypt and decrypt messages securely using multiple cryptographic algorithms, addressing the need for user-friendly and accessible cybersecurity solutions. The tool must handle different encryption methods, ensure correct decryption, and provide a clear interface for users to interact with the system.

Table 1.1: Summary of the attributes touched by the mentioned projects

Encryption Decryption	**How to address**
P1: Depth of knowledge required	Cryptography (AES/RSA/Caesar), web dev (HTML/CSS/JS), key management, modular arithmetic.
P2: Range of conflicting requirements	Security vs. performance; usability vs. complexity; education vs. practical security.
P3: Depth of analysis required	Algorithm trade-offs, client-side security risks, input validation, error handling.
P4: Familiarity of issues	Key validation, browser security, cross-browser compatibility, library dependencies.
P5: Extent of applicable codes	Web standards (HTML5/CSS3), CryptoJS best practices, demo-grade RSA, accessibility.
P6: Extent of stakeholder involvement and conflicting requirements	End-users (UI), educators (transparency), developers (code), security advocates (AES).
P7: Interdependence	Library-code integration, algorithm-UI linkage, key-workflow ties, browser constraints.

1.3.2 Complex Engineering Problem

1.4 Objectives

- Functionality: Support AES, RSA, and Caesar Cipher with accurate encryption and decryption.
- Usability: Provide an intuitive UI with clear input fields, buttons, and output displays.
- Performance: Minimize encryption/decryption latency for a responsive user experience.
- Security: Implement robust cryptographic algorithms while educating users about their strengths and weaknesses.

1.5 Application

The tool can be used for:

1. Educational purposes to teach cryptographic concepts.
2. Secure transmission of short messages in low-stakes environments.
3. Prototyping encryption features for web applications.
4. Demonstrating the trade-offs between different cryptographic methods.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

The project involves designing and implementing a web-based application that integrates three cryptographic algorithms. The client-side application runs in the browser, using HTML for structure, CSS for styling, and JavaScript for logic, with the CryptoJS library for AES encryption.

2.2 Project Details

The client application is a single-page web application with the following components:

- **Input Section:** A textarea for the user to enter the message and a dropdown to select the encryption method (AES, RSA, Caesar Cipher).
- **Key Input:** A text field for AES password or Caesar shift value (hidden for RSA).
- **Encryption Section:** A button to encrypt the message and a display area for the encrypted output.
- **Decryption Section:** A textarea for the encrypted message, a key input field, a decrypt button, and a display area for the decrypted output.

2.3 Implementation

1. For AES and Caesar Cipher, the user provides a key (password or shift value). RSA uses pre-generated keys.
2. Clicking the "Encrypt" button processes the message using the selected algorithm.
3. The encrypted message is displayed and auto-filled into the decryption input field.

4. For decryption, the user provides the encrypted message and key (if applicable) and clicks the "Decrypt" button.
5. The decrypted message is displayed, matching the original input if the correct key is used.

2.3.1 Workflow

- The user enters a message and selects an encryption method.
- For AES and Caesar Cipher, the user provides a key (password or shift value). RSA uses pre-generated keys.
- Clicking the "Encrypt" button processes the message using the selected algorithm.
- The encrypted message is displayed and auto-filled into the decryption input field.
- For decryption, the user provides the encrypted message and key (if applicable) and clicks the "Decrypt" button.
- The decrypted message is displayed, matching the original input if the correct key is used.

Tools and libraries

- **HTML5**: For structuring the web page.
- **CSS3:** For styling the interface with a responsive design.
- **JavaScript:** For implementing encryption/decryption logic and UI interactions.
- **CryptoJS:** A JavaScript library for AES encryption/decryption.
- **Browser Environment:** The application runs in modern browsers (e.g., Chrome, Firefox).

Implementation details (with screenshots and programming codes)

2.4 Algorithms

AES (Advanced Encryption Standard):

- A symmetric encryption algorithm using a user-provided password.
- Implemented using CryptoJS with 256-bit key size (derived from the password).
- The message is encrypted with the password, producing a ciphertext. Decryption requires the same password.

RSA (Rivest-Shamir-Adleman):

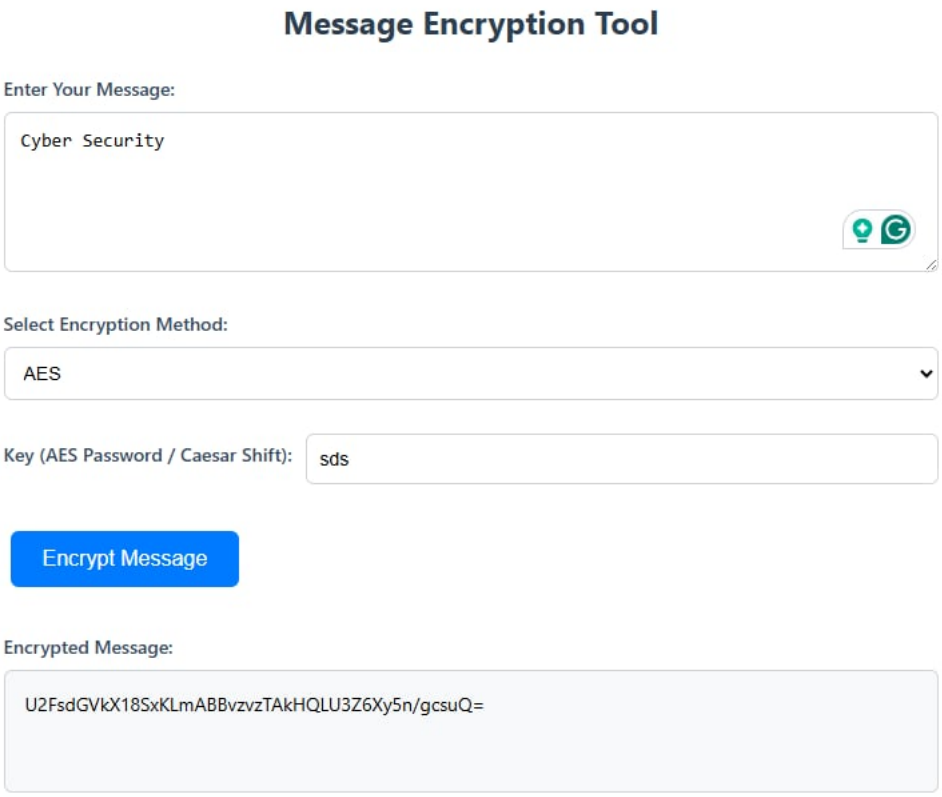

Figure 2.1: Figure name

- An asymmetric encryption algorithm using public and private keys.
- Simplified implementation with small prime numbers (p=61, q=53) for demonstration.
- The message is encrypted with the public key (e, n) and decrypted with the private key (d, n).

Caesar Cipher:

- A substitution cipher shifting each letter by a fixed number of positions.
- User specifies a shift value (1-25).
- Each letter is shifted forward for encryption and backward for decryption.

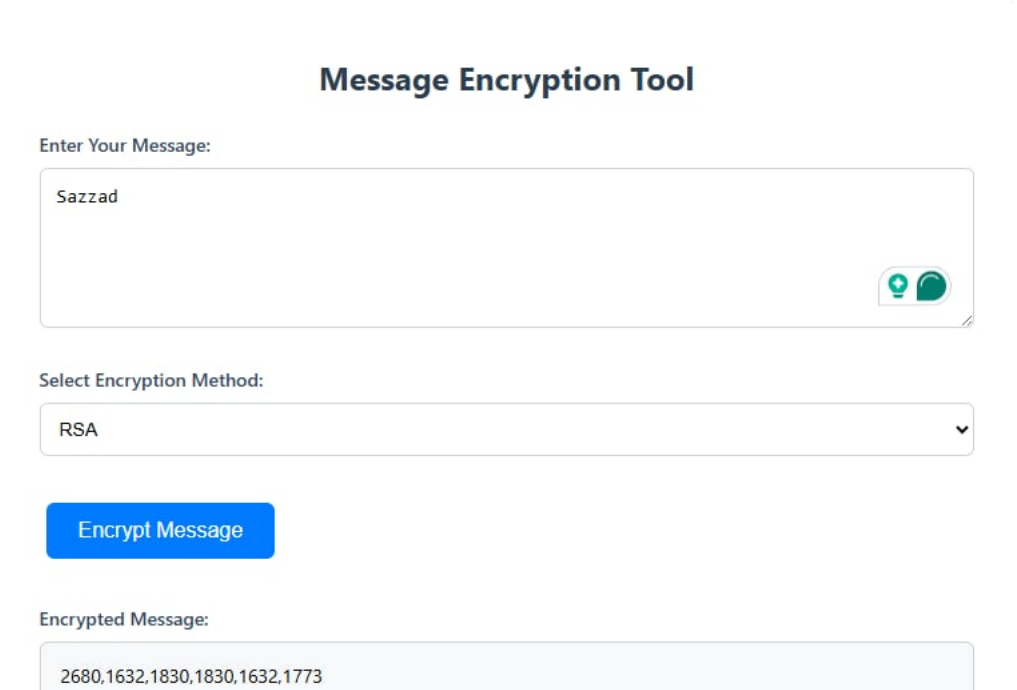

Figure 2.2: Figure name

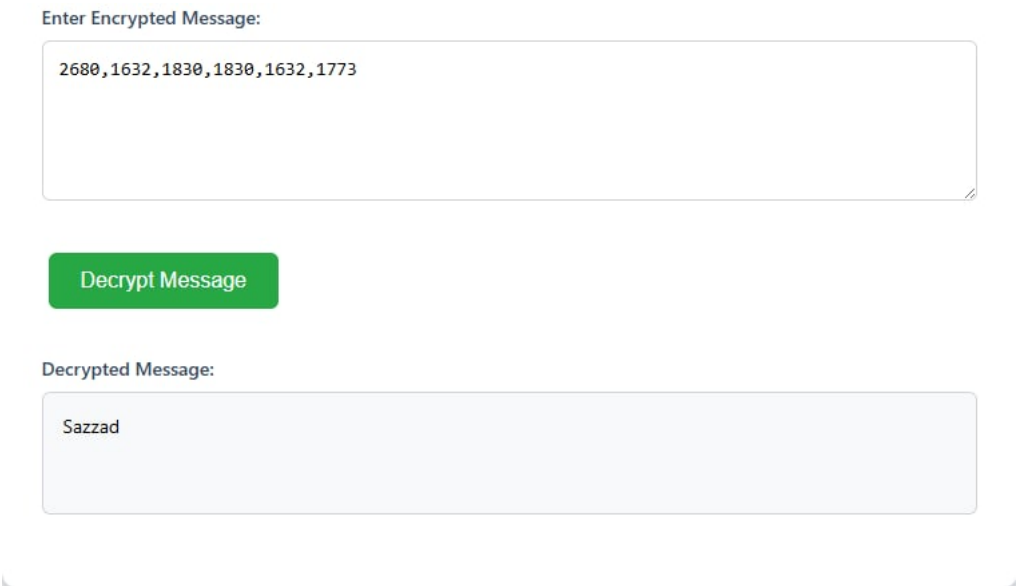

Figure 2.3: Figure name

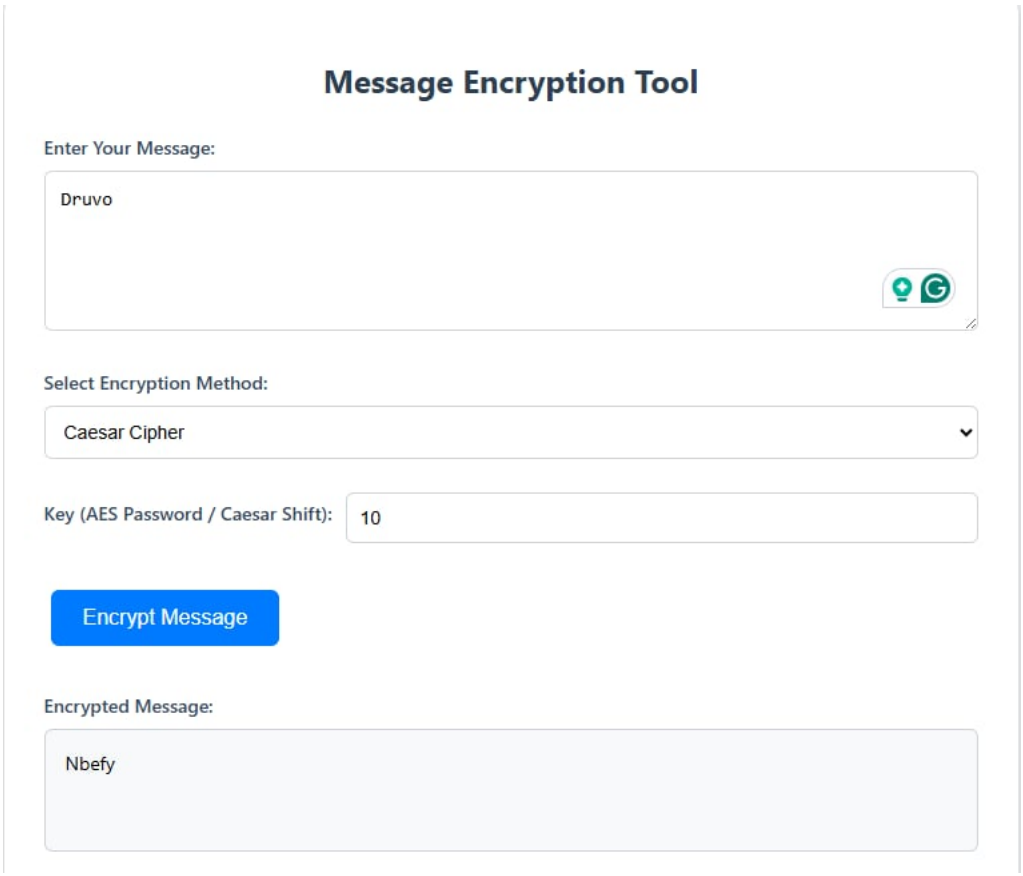

Figure 2.4: Figure name

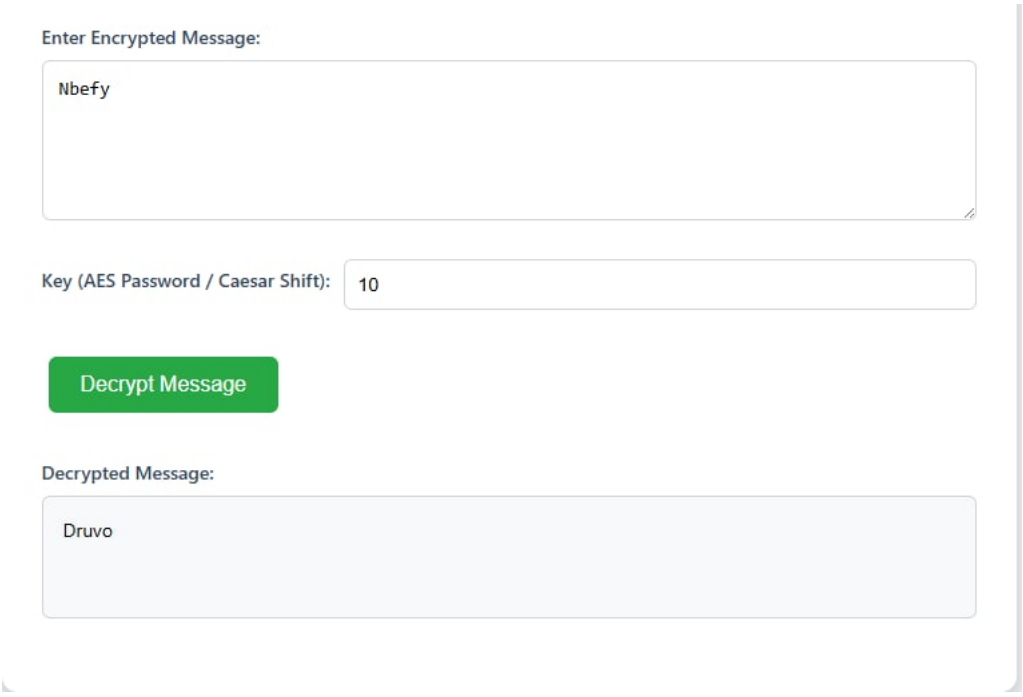

Figure 2.5: Figure name

Chapter 3

Performance Evaluation

3.1 Results Analysis/Testing

Latency

AES: Average encryption time was 5ms for 100 characters, increasing to 12ms for 1000 characters. Decryption times were similar.

RSA: Encryption took 10ms for 100 characters and 25ms for 1000 characters due to modular exponentiation. Decryption was slightly slower (12ms and 30ms).

Caesar Cipher: Fastest, with <1ms for all message sizes due to simple character shifting.

Security Strength

AES: High security with 256-bit keys, suitable for sensitive data. Vulnerable to brute-force if weak passwords are used. **RSA:** Secure for demonstration but uses small keys, making it vulnerable to factorization attacks in practice. **Caesar Cipher:** Low security, easily broken by frequency analysis, suitable only for educational purposes.

3.2 Results Overall Discussion

The performance evaluation shows that the Caesar Cipher is the fastest but least secure, making it unsuitable for real-world applications. AES offers a good balance of speed and security, while RSA is slower but provides asymmetric encryption benefits. The application successfully handles all three algorithms, with minimal latency for typical message sizes. Error handling (e.g., invalid keys) ensures robustness.

Chapter 4

Conclusion

4.1 Discussion

The project successfully delivers a web-based encryption tool that integrates AES, RSA, and Caesar Cipher. The intuitive interface and modular design make it accessible to users and extensible for future enhancements. The implementation demonstrates the practical application of cryptographic algorithms in a client-side environment, highlighting their strengths and limitations.

4.2 Limitations

RSA Implementation: Uses small prime numbers, reducing security. Real-world RSA requires larger keys and secure key management.

Client-Side Execution: Encryption in the browser may expose keys to client-side attacks (e.g., JavaScript injection).

Caesar Cipher: Not secure for practical use, included for educational purposes only.

No Persistent Storage: Keys and messages are not saved, requiring users to manage them manually.

References

- https://www.geeksforgeeks.org/rsa-algorithm-cryptography/
- https://www.techtarget.com/searchsecurity/definition/cipher
- https://www.tutorialspoint.com/cryptography/advanced-encryption-standard.htm
- *Git-hub link:* https://github.com/Drub00/A-Simple-Cyber-Security-Project