CipherCraft Documentation

by

Islam Abdulhakeem

ID:224179

Email:islam.abdulhakeem@msa.edu.eg

&&

Youssef Omar

ID:222659

Email:youssef.omar@msa.edu.eg

Bachelor of computer science (SE programme)

in the
Faculty of Computer Science
of the
October University for Modern Sciences and Arts (MSA), EGYPT

Project advisor:
Dr. Ali Somaie

Abstract

CipherCraft is a multi-layer encryption web application that enables users to chain an unlimited number of cryptographic algorithms—AES, Autokey, Rail Fence, Shift, and Vigenère—in any order for enhanced security. Built with modern web technologies, the platform offers real-time encryption, decryption, and cryptanalysis while visualizing every intermediate stage. Designed to be both educational and functional, CipherCraft lets learners experiment freely with layered ciphers through an interactive interface with arrow buttons for reordering and dropdown menus for algorithm selection. This paper outlines the system's design, implementation, cryptanalysis features, and avenues for future enhancement.

1 Introduction

Modern security systems overwhelmingly rely on robust, standardized algorithms such as AES and RSA, while classical ciphers persist mainly in coursework and historical studies [1]. Existing educational tools—CrypTool, assorted web demonstrators, and many university lab sheets—tend to present each cipher in isolation [2]. Few resources illustrate how combining algorithms can boost (or, if done poorly, weaken) overall security, offer security through diversity, or provide tools for cryptanalysis to evaluate their weaknesses[**?**].

1.1 Related Work

CrypTool remains the de-facto desktop reference for cipher education, providing interactive modules for individual algorithms [3]. Numerous online simulators show Vigenère or Caesar ciphers, while hands-on AES visualizers focus on a single mode such as ECB or CBC [4]. None of these platforms, however, integrate real-time, multi-layer encryption with cryptanalysis inside a modern web application with a responsive UI [5].

1.2 Our Contribution

CipherCraft fills that gap by:

1. Offering an interface to configure an unlimited number of encryption layers, with algorithm selection via dropdown menus and reordering via arrow buttons.
2. Displaying intermediate plaintext/ciphertext after each layer, clarifying how every algorithm transforms the data.
3. Including a self-contained educational AES implementation with flexible key sizes (128, 192, or 256 bits) to keep the cryptographic logic transparent.
4. Marrying classical and contemporary techniques in one coherent user experience, encouraging "defence-in-depth" thinking.
5. Providing cryptanalysis tools to evaluate the strengths and weaknesses of each algorithm.

2 Key Features

- **Five algorithms**: AES (CBC, with 128/192/256-bit keys), Autokey, Rail Fence, Shift, Vigenère.
- **Infinite layer sequencing**: Add unlimited layers, select algorithms via dropdown, and reorder using up/down arrow buttons.
- **Real-time preview**: Instant encryption/decryption and step-through visualization.
- **Cryptanalysis tools**: Analyze vulnerabilities and strengths of each algorithm.
- **Parameter control**: User-defined keys, block sizes, alphabets, and IVs for AES.
- **Responsive UI**: React 18 + Tailwind CSS for mobile and desktop.

3 Technical Architecture

3.1 Technology Stack

Technology	Purpose
React 18	Frontend framework for building the user interface
TypeScript	Type-safe JavaScript for improved development experience
Tailwind CSS	Utility-first CSS framework for styling
Vite	Next-generation frontend build tool

Table 1: Core technologies powering CipherCraft.

3.2 Technology Stack

The chosen technologies are detailed in Table 1 and are selected to align with industry best practices for frontend development.

- **React 18:** As the foundation of the frontend, React 18 provides a robust and component-based architecture for building interactive user interfaces. Its declarative nature simplifies UI development, while the latest features such as concurrent rendering enhance performance and responsiveness.
- **TypeScript:** By adding static typing to JavaScript, TypeScript improves code quality and maintainability. It allows for better tooling support, early error detection, and clearer documentation, which is especially beneficial in large-scale or long-term projects.
- **Tailwind CSS:** This utility-first CSS framework enables rapid UI development with a consistent design system.
- **Vite:** Vite is a modern frontend build tool that significantly improves development speed through instant hot module replacement and optimized bundling.oice for building React applications.

3.3 Project Structure

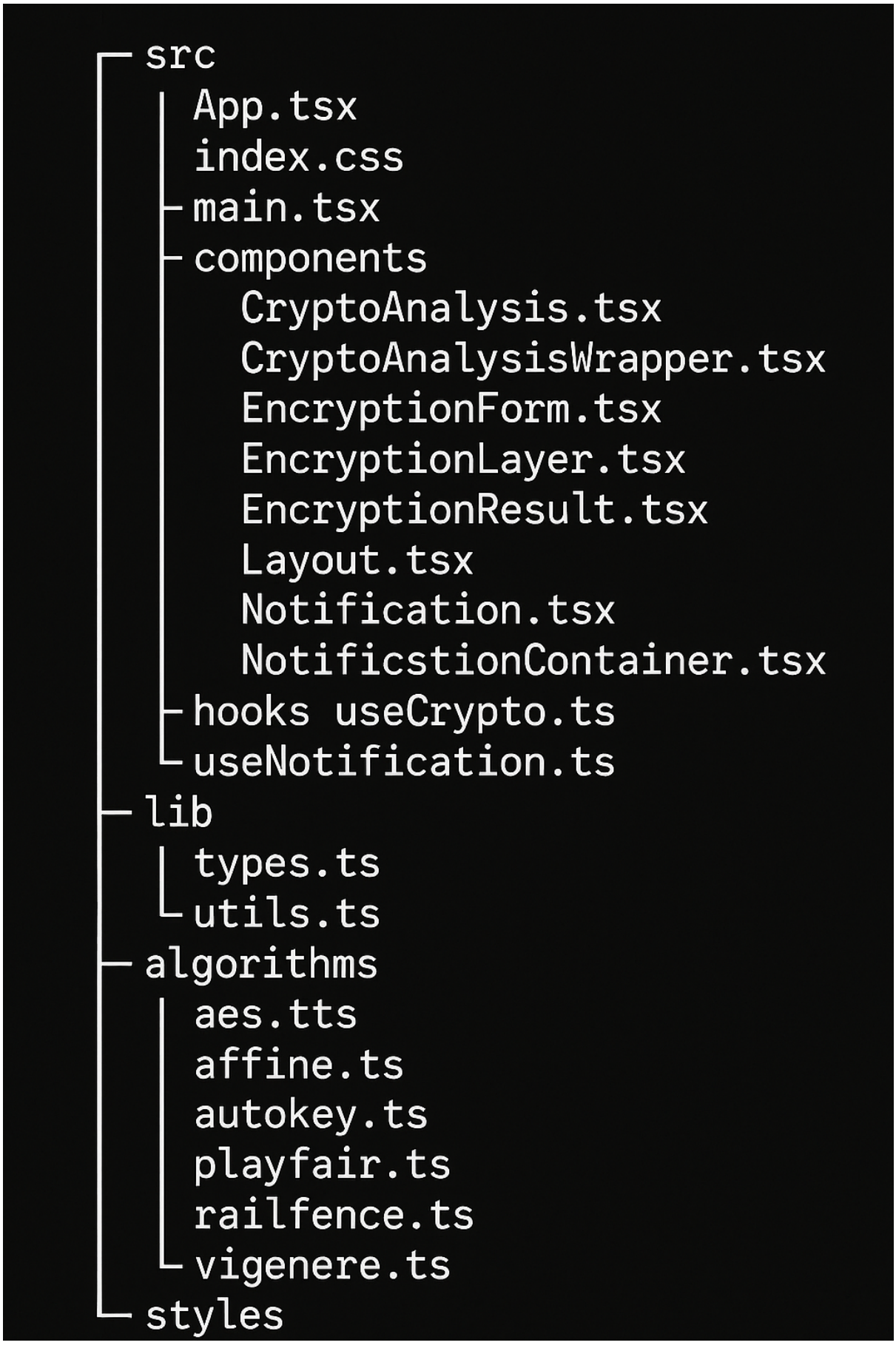

Figure 1: Repository layout highlighting separation of concerns.

3.4 Architecture Diagram

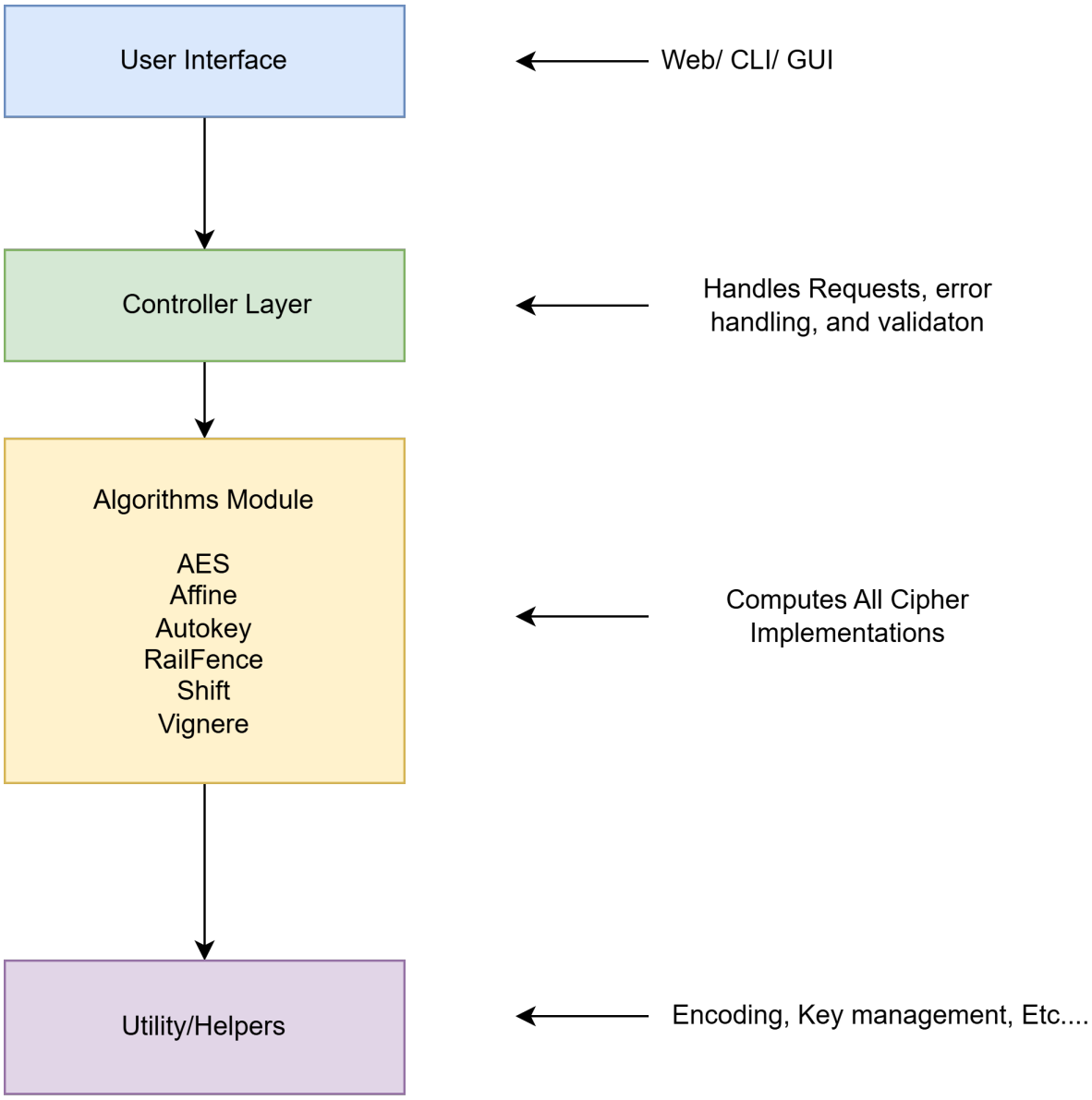

Figure 2: High-level component and data-flow architecture.

4 Encryption Algorithms

4.1 AES (Advanced Encryption Standard) [6]

4.1.1 Overview

CipherCraft includes ateaching-oriented implementation of AES, a symmetric-key block cipher operating on 128-bit blocks with flexible key sizes.

4.1.2 Implementation Details

- **Key size**: 128, 192, or 256 bits (16, 24, or 32 bytes)

- **Block size**: 128 bits
- **Rounds**: 10 (128-bit key), 12 (192-bit key), or 14 (256-bit key)
- **Key expansion**: Generates a round key per round
- **Primitives**: SubBytes, ShiftRows, MixColumns, AddRoundKey
- **Mode**: Cipher Block Chaining (CBC)
- **Initialization Vector**: 128-bit random IV for each encryption

4.1.3 Mathematical Formulation

AES in CBC mode operates iteratively over 10, 12, or 14 rounds per block, depending on key size. For the i-th 128-bit plaintext block P_i, previous ciphertext block C_{i-1} (or IV for $i = 1$), and round key K_r, the encryption is:

$$C_i = \text{AddRoundKey}(\text{MixColumns}(\text{ShiftRows}(\text{SubBytes}(P_i \oplus C_{i-1})))) \oplus K_r, \quad C_0 = IV, \tag{1}$$

$$P_i = \text{SubBytes}^{-1}(\text{ShiftRows}^{-1}(\text{MixColumns}^{-1}(\text{AddRoundKey}^{-1}(C_i \oplus K_r)))) \oplus C_{i-1}, \tag{2}$$

where C_i is the i-th ciphertext block, and the process repeats for each round with a new K_r derived from key expansion. Decryption reverses the process, using the previous ciphertext block (or IV) for XOR.

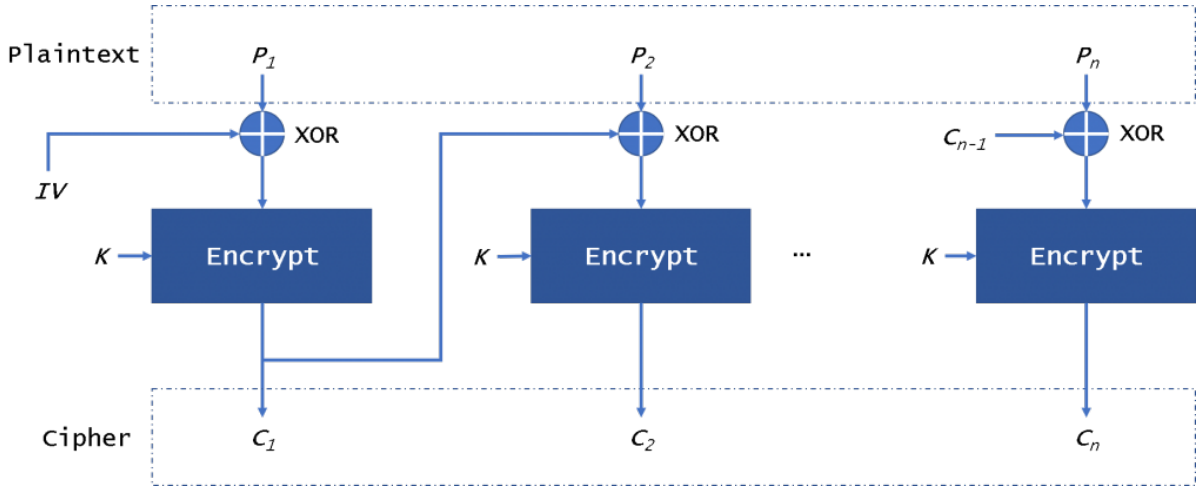

Figure 3: AES code-flow diagram inside CipherCraft.

4.2 Autokey Cipher

4.2.1 Overview

Autokey augments the initial keyword with the plaintext itself, yielding a dynamic keystream.

4.2.2 Implementation Details

- **Key type**: Alphabetic (A-Z)
- **Alphabet**: Uppercase Latin A-Z
- **Keystream**: Initial keyword + plaintext (encryption) / initial keyword + decrypted text (decryption)
- **Encrypt**: Modular addition (mod 26)
- **Decrypt**: Modular subtraction (mod 26)

4.2.3 Mathematical Formulation

Let K be the initial keyword, P the plaintext, and K_{stream} the keystream where $K_{\text{stream}} = K + P$ (after the keyword). Then:

$$C_i = (P_i + K_{\text{stream},i}) \mod 26, \tag{3}$$

$$P_i = (C_i - K_{\text{stream},i} + 26) \mod 26, \tag{4}$$

where C_i and P_i are the i-th ciphertext and plaintext characters, and $K_{\text{stream},i}$ is the i-th character of the keystream.

4.3 Rail Fence Cipher

4.3.1 Overview

The Rail Fence cipher is a transposition cipher that rearranges plaintext characters in a zigzag pattern.

4.3.2 Implementation Details

- **Key type**: Number of rails
- **Encrypt**: Arrange characters in a zigzag pattern and read off row by row
- **Decrypt**: Reconstruct the zigzag pattern based on rail count

4.3.3 Mathematical Formulation

For a plaintext P of length n and r rails, the position pos_i of the i-th character in the ciphertext is determined by:

$$\text{pos}_i = i \cdot (r-1) - 2 \cdot \left\lfloor \frac{i \mod (2r-2)}{r-1} \right\rfloor \cdot (r-1), \tag{5}$$

where the zigzag pattern cycles every $2r - 2$ positions, and decryption reverses this mapping.

4.4 Shift Cipher

4.4.1 Overview

The Shift cipher is a simple substitution cipher where each letter is shifted by a fixed number of positions.

4.4.2 Implementation Details

- **Key type**: Shift value (0-25)
- **Alphabet**: Uppercase Latin A-Z
- **Encrypt**: Modular addition (mod 26)
- **Decrypt**: Modular subtraction (mod 26)

4.4.3 Mathematical Formulation

Given a shift key k (0 to 25):

$$C_i = (P_i + k) \mod 26, \tag{6}$$
$$P_i = (C_i - k + 26) \mod 26, \tag{7}$$

where P_i and C_i are the i-th plaintext and ciphertext characters, mapped to 0-25 (A=0, B=1, ..., Z=25).

4.5 Vigenère Cipher

4.5.1 Overview

The Vigenère cipher is a poly-alphabetic substitution cipher whose keystream is a repeating keyword.

4.5.2 Implementation Details

- **Key type**: Alphabetic (A-Z)
- **Alphabet**: Uppercase Latin A-Z
- **Encrypt**: Modular addition (mod 26)
- **Decrypt**: Modular subtraction (mod 26)

4.5.3 Mathematical Formulation

Let K be the keyword of length m, repeated to match plaintext length:

$$C_i = (P_i + K_{i \mod m}) \mod 26, \tag{8}$$
$$P_i = (C_i - K_{i \mod m} + 26) \mod 26, \tag{9}$$

where P_i and C_i are the i-th plaintext and ciphertext characters, and $K_{i \mod m}$ is the corresponding key character.

5 Cryptanalysis

CipherCraft provides tools to analyze the cryptographic strength and vulnerabilities of each algorithm, offering insights into their practical security.

5.1 AES (CBC Mode)

5.1.1 Vulnerabilities

- **IV Reuse**: Reusing the same IV with the same key can leak information about the plaintext.
- **Padding Oracle Attacks**: Improper padding handling can expose plaintext in certain scenarios.

5.1.2 Analysis Tools

- Visualizes block chaining to demonstrate dependency on prior blocks.
- Checks for IV randomness and detects reuse.

5.2 Autokey Cipher

5.2.1 Vulnerabilities

- **Keystream Dependency**: The keystream incorporates the plaintext, making it susceptible to known-plaintext attacks.
- **Frequency Analysis**: Once the initial keyword is guessed, frequency analysis can reveal the rest of the keystream.

5.2.2 Analysis Tools

- Performs frequency analysis on the ciphertext to detect patterns.
- Simulates known-plaintext attacks to recover the initial keyword.

5.3 Rail Fence Cipher

5.3.1 Vulnerabilities

- **Limited Key Space**: The number of rails is typically small, making brute-force attacks feasible.
- **Pattern Recognition**: The zigzag pattern can be reconstructed with enough ciphertext.

5.3.2 Analysis Tools

- Brute-forces possible rail counts to decrypt the ciphertext.
- Displays character distribution to identify the zigzag pattern.

5.4 Shift Cipher

5.4.1 Vulnerabilities

- **Small Key Space**: Only 25 possible shifts (excluding 0), easily brute-forced.

- **Frequency Analysis**: Preserves letter frequencies, making it vulnerable to statistical attacks.

5.4.2 Analysis Tools

- Brute-forces all possible shifts and ranks results by English letter frequency.
- Plots letter frequency distribution against expected English frequencies.

5.5 Vigenère Cipher

5.5.1 Vulnerabilities

- **Key Repetition**: The repeating keyword creates periodic patterns in the ciphertext.
- **Kasiski Examination**: Repeated sequences in the ciphertext can reveal the key length.
- **Frequency Analysis**: Once the key length is known, the cipher reduces to multiple Shift ciphers.

5.5.2 Analysis Tools

- Applies Kasiski examination to estimate key length.
- Performs frequency analysis on each key position to deduce the keyword.

6 User Interface Components

6.1 Main Layout

1. **Header**: App name and branding.
2. **Mode Toggle**: Switch between encrypt/decrypt.
3. **Input Section**: Textarea for the message.
4. **Algorithm Section**: List of layers, each with a dropdown for algorithm selection, arrow buttons for reordering, and a button to add new layers.
5. **Results Section**: Final and intermediate outputs.
6. **Cryptanalysis Section**: Tools and visualizations for analyzing cipher weaknesses.

6.2 Interactive Features

- Dropdown menus to select algorithms (AES, Autokey, Rail Fence, Shift, Vigenère) for each layer.
- Up/down arrow buttons to reorder layers.
- Button to add new layers, supporting unlimited layers with optional algorithm repetition.

- Toggle switches to enable/disable a cipher layer.
- Expandable cards for per-cipher settings (e.g., key size for AES, rail count for Rail Fence).
- Cryptanalysis visualizations (e.g., frequency charts, pattern detection).
- One-click copy-to-clipboard for any result.
- Optional display of intermediate transformations.

7 Security Considerations

7.1 Strengths

1. Multi-layer encryption with unlimited layers increases defence in depth.
2. Algorithm diversity: Combines modern and classical schemes.
3. Transparency: Code is educational and auditable.
4. Cryptanalysis tools: Enable users to understand and evaluate cipher weaknesses.

7.2 Limitations

1. CBC mode requires secure IV management to prevent reuse.
2. Classical ciphers are trivially breakable today.

7.3 Recommendations

1. Off-load sensitive operations to a server.
2. Introduce secure key-management practices.

8 Performance Analysis

Algorithm	**Time Complexity**	**Space Complexity**
AES	O(n)	O(n)
Autokey	O(n)	O(n)
Rail Fence	O(n)	O(n)
Shift	O(n)	O(n)
Vigenère	O(n)	O(n)

Table 2: Algorithmic complexity of the five ciphers.

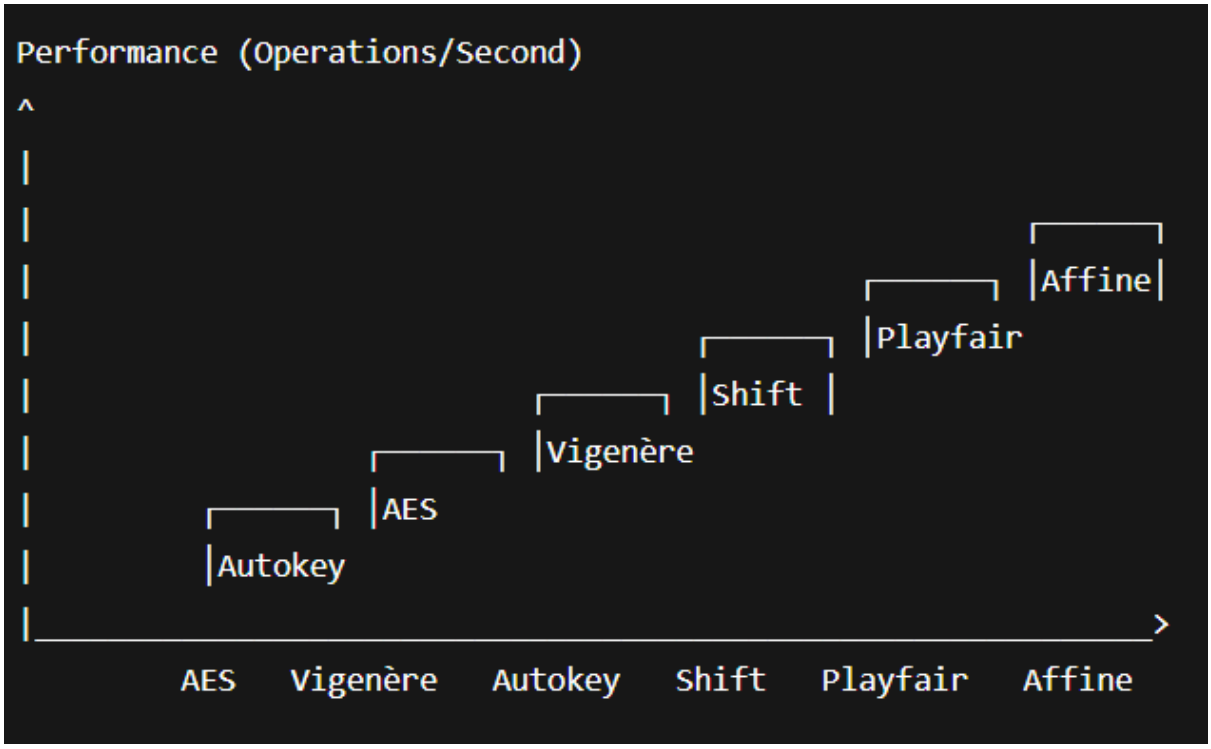

Figure 4: Relative throughput (operations per second).

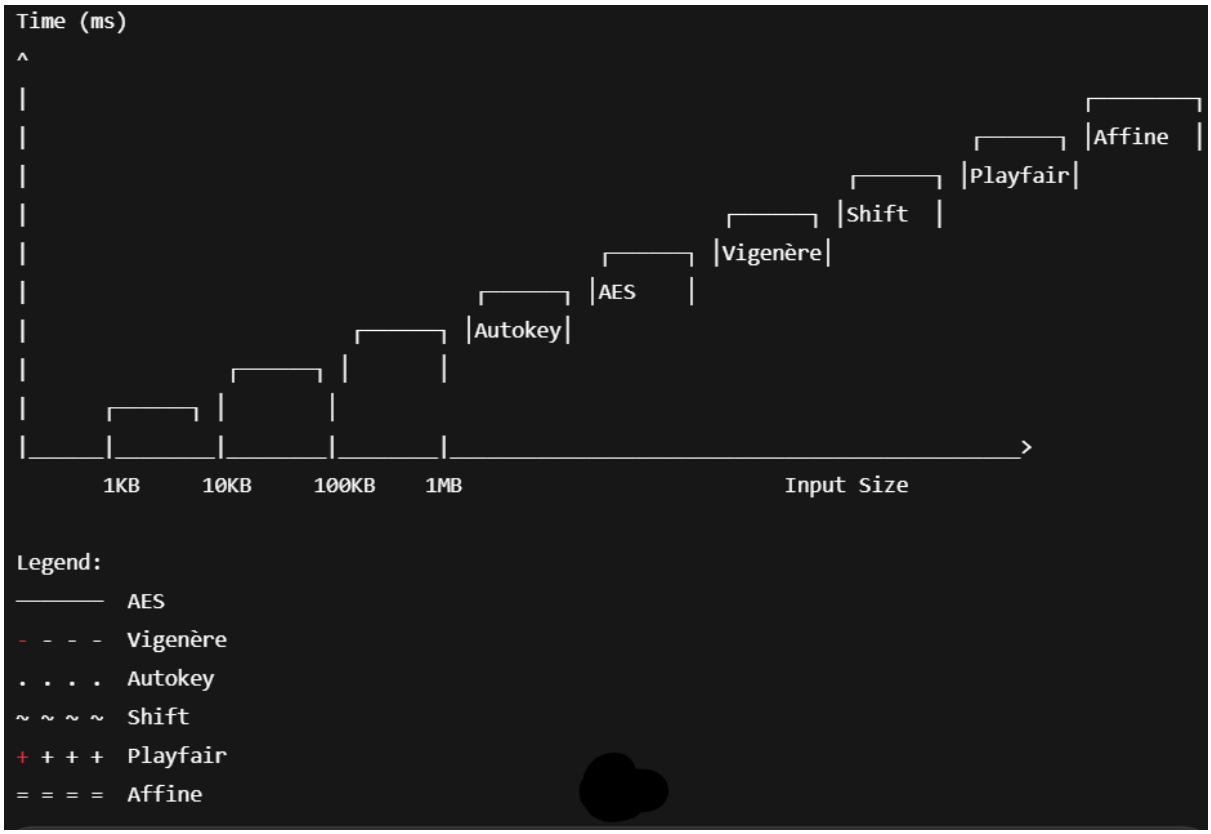

Figure 5: Encryption/decryption time versus input size.

9 Future Enhancements

9.1 Short-term

- Add RSA, DES, Blowfish modules.
- Support file encryption.
- User presets for favourite cipher stacks.

9.2 Long-term

- Hybrid (public-key + symmetric) workflows.
- User authentication and secure key storage.
- Enhanced cryptanalysis tools.
- Companion mobile app.
- Server-side API for secure operations.

10 Conclusion

CipherCraft shows how a modern web app can blend classical and contemporary ciphers with an unlimited number of configurable layers, giving users live insight into each transformational step and their cryptographic strengths and weaknesses. Although the current prototype is not ready for production security, its extensible design paves the way for stronger modes, server-side hardening, advanced cryptanalysis, and additional algorithms.

References

[1] Z. Z. Mammeri, *Cryptography: Algorithms, Protocols, and Standards for Computer Security*, John Wiley & Sons, 2024.https://onlinelibrary.com/9781394207510

[2] Y. A. Younis, K. Kifayat, Q. Shi, E. Matthews, G. Griffiths, and R. Lambertse, "Teaching cryptography using Cypher (interactive cryptographic protocol teaching and learning), 2020. https://doi.org/10.1145/3410352.3410742

[3] CrypTool Project, https://www.cryptool.org.

[4] A. S. Al-Bayati, *Enhancing Performance of Hybrid AES, RSA and Quantum Encryption Algorithm*, Ph.D. dissertation, Anglia Ruskin Research Online (ARRO), 2023Al-Bayati, Anwer S. (2023).https://hdl.handle.net/10779/aru.23768127.v1.

[5] D. Cevallos-Salas, J. Estrada-Jiménez, and D. S. Guamán, "Application layer security for internet communications: A comprehensive review, challenges, and future trends,2024, Elsevier.10.1016/j.compeleceng.2024.109498

[6] NIST, FIPS 1997 - Advanced Encryption Standard (AES).https://doi.org/10.6028/NIST.FIPS.197-upd1