Enhanced Rail Fence Cipher (ERFC): A Modern Approach to Secure Emoji-Based Encryption

Mohid Arshad, Mohammad Umar
School of Electrical Engineering and Computer Science, NUST, Islamabad, Pakistan
{marshad.bsds23seecs, mraza.bsds23seecs}@seecs.edu.pk

***Abstract*—The safeguarding of sensitive data in the modern digital environment has been considered increasingly vital. Although traditional encryption methods are successful, they usually pose challenges in usability for the average user. This paper introduces the Enhanced Rail Fence Cipher, a novel encryption method that utilizes emojis to secure alphanumeric information. In this scheme, a specific emoji is assigned to each character, and the encrypted text is arranged within a randomized grid structure, making ERFC successfully merge security with user-friendly features. This paper gives ERFC's design, benefits, and applications.**

CONTENTS

I. INTRODUCTION

Digital communication plays a pivotal role in our daily lives, connecting individuals, businesses, and governments across the globe. Ensuring the security of the information transmitted through these channels is paramount to prevent unauthorized access and maintain confidentiality. Encryption has long served as the cornerstone of securing digital communication, with classical methods such as the *Caesar* and *Vigenère* ciphers forming the foundation of cryptography. These traditional techniques introduced fundamental concepts of substitution and transposition, shaping the early development of secure communication systems.

However, as digital communication has evolved, so too have the threats and challenges associated with maintaining security. Modern encryption algorithms such as AES and RSA provide robust security but often involve significant complexity, which can make them less accessible to non-expert users. This complexity highlights the growing need for encryption techniques that strike a balance between security and usability, catering to the diverse needs of contemporary users.

This paper introduces the Enhanced Rail Fence Cipher (ERFC), an innovative encryption method that leverages emojis to secure alphanumeric data. Emojis, a ubiquitous feature of modern digital communication, provide a unique opportunity to enhance user engagement while simultaneously ensuring data security. By combining the simplicity of the Rail Fence Cipher with advanced features such as substitution mapping and randomized grid structures, ERFC offers a user-friendly yet secure encryption solution suitable for a wide range of applications.

II. BACKGROUND

The Rail Fence Cipher is one of the simplest forms of transposition cipher, requiring the arrangement of plaintext characters in a zigzag pattern across multiple "rails" or rows. The ciphertext is then derived by sequentially reading the characters from each row. While this method is easy to implement, it is highly vulnerable to cryptanalysis techniques such as frequency analysis, which can exploit predictable patterns in the ciphertext.

Modern encryption algorithms, including Advanced Encryption Standard (AES) and Rivest-Shamir-Adleman (RSA), offer significantly higher levels of security. AES

relies on complex substitution and permutation operations, while RSA employs mathematical principles such as prime factorization for secure key generation. These algorithms are widely used in applications requiring robust security, such as online banking, secure messaging, and government communications.

Despite their effectiveness, these advanced algorithms often present usability challenges for everyday users. The complexity of their implementation and the computational resources required can limit their accessibility, especially in scenarios where simplicity and user-friendliness are prioritized. This has created a demand for encryption methods that are both secure and easy to adopt.

Emojis, a widely recognized and utilized element of digital communication, present an opportunity to bridge the gap between security and usability. Their universal appeal and versatility make them an ideal candidate for integration into encryption techniques, enabling both enhanced security and greater user engagement. The Enhanced Rail Fence Cipher (ERFC) builds upon this concept, incorporating emojis into its encryption process to create a method that is both intuitive and effective.

III. Motivation

The development of the Enhanced Rail Fence Cipher (ERFC) was driven by several key motivations, reflecting the need for encryption techniques that are both secure and accessible:

- **Improved Security**: Classical encryption methods, such as the Rail Fence Cipher, offer simplicity but lack robustness against modern cryptographic attacks. ERFC enhances the security of the Rail Fence Cipher by integrating substitution mapping and arranging encrypted text in a randomized grid layout. These features significantly improve its resistance to cryptanalysis techniques like frequency analysis and pattern recognition.
- **User Engagement and Accessibility**: Many encryption systems prioritize security at the expense of user-friendliness, making them difficult for non-expert users to adopt. ERFC leverages emojis—an integral feature of contemporary digital communication—to create an encryption process that is engaging, intuitive, and relatable. This approach ensures that users of all ages and technical backgrounds can interact with the encryption system effectively.
- **Alignment with Modern Communication Practices**: The widespread use of emojis in digital communication underscores their relevance in today's online ecosystem. ERFC aligns encryption techniques with these modern practices, making them more compatible with contemporary communication platforms and enhancing their overall utility.
- **Flexibility and Customization**: ERFC offers a high degree of adaptability, allowing users to customize the encryption process through key-based mapping and adjustable grid sizes. This flexibility ensures that the cipher can cater to diverse use cases, ranging from personal communication to educational tools and secure data exchange.
- **Balancing Security and Usability**: While advanced encryption algorithms such as AES and RSA provide robust security, they often require significant computational resources and expertise. ERFC strikes a balance by delivering moderate security alongside ease of use, making it suitable for applications where simplicity and performance are prioritized.

These motivations emphasize the need for encryption techniques that blend security and user-friendliness, addressing the challenges posed by traditional methods while aligning with modern communication trends. ERFC represents a significant step forward in achieving this balance, offering a solution that is both innovative and practical.

IV. ERFC Architecture

ERFC is an improvement of the basic Rail Fence Cipher with **substitution mapping**, matrix-based structuring, salt generation, and integrity hashing. The design has four major components:

A. *Substitution Mapping*

Goal: Create a one-to-one mapping between alphabetic characters ($A-Z$, $a-z$, $0-9$) and 62 unique emojis.

Mechanism:

1. **Character Set**: The cipher uses 62 characters: 26 uppercase letters, 26 lowercase letters, and 10 digits.
2. **Emoji List**: A pre-defined list of 62 unique emojis is used for substitution.
3. **Shuffling Based on Key**:
   - The secret key `K` is hashed using **SHA-256** to generate a consistent seed.
   - This seed is used to shuffle the emoji list, ensuring the mapping is unique for each key.
4. **Mapping**:
   - Each character is assigned an emoji from the shuffled list.
   - A reverse mapping is maintained for decryption.

Security Implications:

- **Dependency on the Secret Key**: The substitution mapping relies on the secret key, preventing unauthorized access.
- **Uniqueness**: Each character maps to a unique emoji, eliminating decryption ambiguities.

Mathematical Representation:

Let:

- C be the set of alphanumeric characters: $C = \{A, B, \ldots, Z, a, b, \ldots, z, 0, 1, \ldots, 9\}$.
- E be the set of emojis: $E = \{e_1, e_2, \ldots, e_{62}\}$.

The substitution mapping M is defined as:

$$M : C \to E$$

$$M(c_i) = e_j \quad \text{for each } c_i \in C \text{ and } e_j \in E$$

The reverse mapping M^{-1} is:

$$M^{-1} : E \to C$$

$$M^{-1}(e_j) = c_i \quad \text{for each } e_j \in E \text{ and } c_i \in C$$

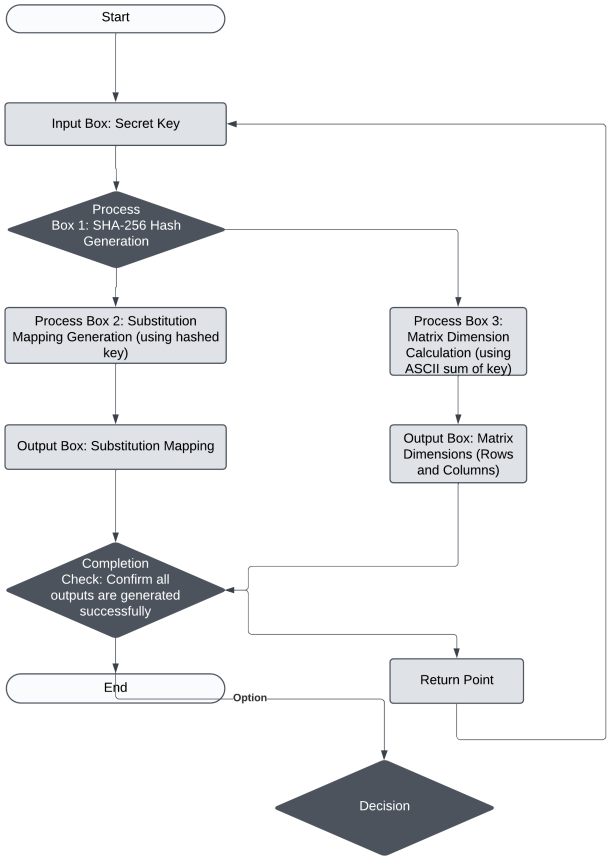

Fig. 1: **Substitution Mapping Diagram**

B. *Matrix Construction*

Objective: Organize the substituted emojis within a grid to obscure the plaintext's structure and enhance security.

Mechanism:

1. **Matrix Dimensions Calculation**:
   - **Expansion Factor** (α): Determines the grid size relative to the plaintext length. An expansion factor of 5 is used.
   - **Rows** (R) and **Columns** (C): Calculated based on the key's ASCII sum to introduce variability.
2. **Random Placement**:
   - Substituted emojis are placed randomly within the grid.
   - Remaining cells are filled with random filler emojis to mask the actual message.
3. **Serialization**:
   - The grid is read row by row to create the cipher body.

Security Implications:

- **Obfuscation**: Random placement within a large grid makes it difficult to identify the actual message.
- **Key Dependence**: Placement randomness is controlled by the key, ensuring unpredictability.

Mathematical Representation:

Let:

- L be the length of the plaintext.
- α be the expansion factor (e.g., 5).
- R be the number of rows.
- C be the number of columns.

Calculate:

$$R = (\text{ASCII sum of } K) \bmod 10 + 3$$

$$C = \left\lceil \frac{\alpha \times L}{R} \right\rceil$$

Ensure:

$$R \times C \;\geq\; \alpha \times L.$$

C. *Salt and Length Encryption*

Objective: Include the encrypted plaintext length in the salt of the ciphertext to avoid information leakage and facilitate decryption.

Steps:

- **Plaintext Length Encoding**:
  - The plaintext length L is expressed as a 4-digit string (e.g., "0001" for length 1).
- **Encryption of Length**:
  - The 4-digit string is XORed with the first 4 bytes of the key's **SHA-256** hash.
  - The result is hex-encoded into an 8-character string.
- **Salt Composition**:
  - The encrypted length is combined with 12 random alphanumeric characters to form a 20-character salt.

Security Implications:

- **Non-Disclosure**: Encrypting the plaintext length conceals the message's size from attackers.
- **Integrity**: Including the encrypted length helps verify the structure during decryption.

Mathematical Representation:

Given:

- SL = 4-digit string of plaintext length.
- $H(K)$ = SHA-256 hash of key K.
- $H(K)[:4]$ = first 4 bytes of the hash.

Encrypt:

$$EL = SL \oplus H(K)[:4]$$

Hex-encode:

$$HL = \text{Hex}(EL)$$

Generate random salt S_r of 12 characters. Final salt:

$$S = HL \parallel S_r.$$

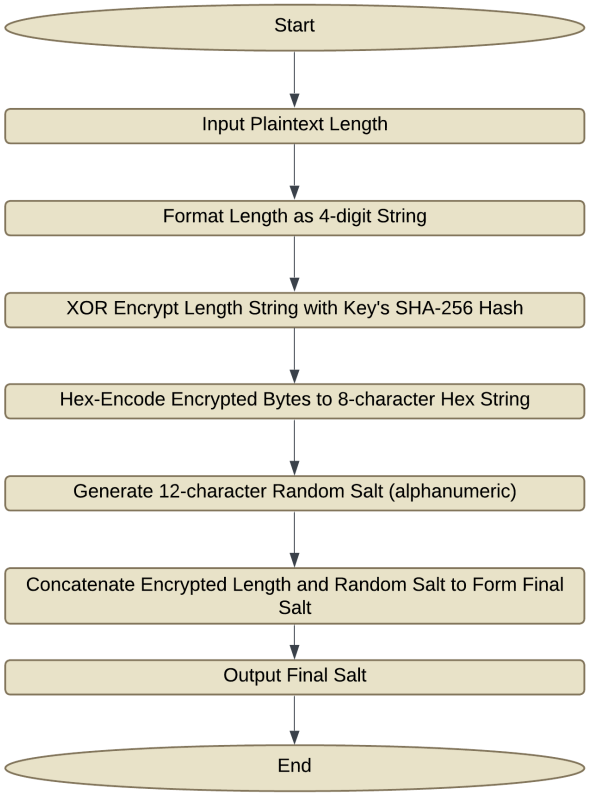

Fig. 2: **Salt Generation and Length Encryption Diagram**

D. Hashing for Integrity

Objective: Ensure the ciphertext's integrity and detect any tampering during transmission.

Mechanism:

1. **Hash Components**:
   - **Key** (K): Secret key used for encryption/decryption.
   - **Salt** (S): Contains the encrypted length and random characters.
   - **Plaintext** (P): The original message.
2. **Hash Generation**:
   - Concatenate the key, salt, and plaintext:
   $$C = K \parallel S \parallel P.$$
   - Compute the SHA-256 hash of the concatenated string:
   $$H = \text{SHA-256}(C).$$
3. **Hash Inclusion**:
   - The hash H is transmitted alongside the ciphertext for integrity verification during decryption.

Security Implications:

- **Integrity Verification**: Detects any changes to the ciphertext or salt.
- **Authentication**: Confirms the origin of message by verifying possession of the correct key.
- **Confidentiality**: Ensures that the encrypted data remains inaccessible to unauthorized parties.

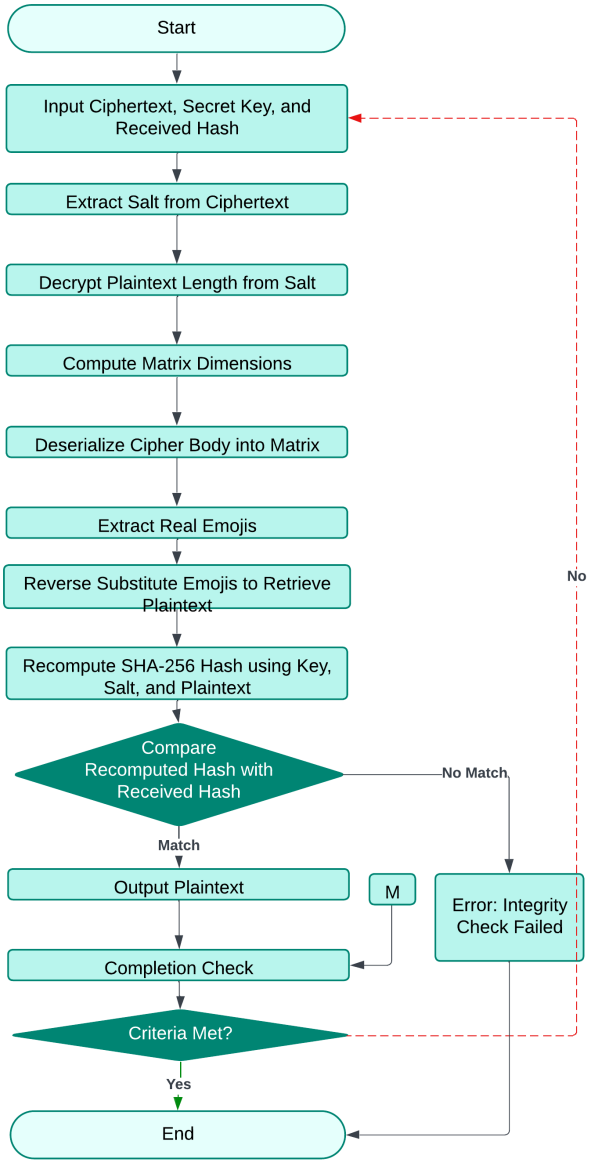

Fig. 3: **Hashing for Integrity Diagram**

V. Operational Workflow

ERFC maintains a clear encryption and decryption procedure to ensure security and integrity.

A. Encryption Process

1. **Plaintext Validation**:
   - Verify that the plaintext is alphanumeric. If not, reject the input.
2. **Substitution Mapping Generation**:
   - Generate a mapping between characters and emojis based on the secret key.
3. **Text Substitution**:

- Replace each character of the plaintext with its corresponding emoji to form the substituted text.

4. **Matrix Dimension Calculation**:
   - Determine the number of rows and columns in the grid using the ASCII sum of the key and the expansion factor.
5. **Construction of Matrix**:
   - Distribute the replaced emojis in the grid at random positions.
   - Fill the remaining cells with filler emojis at random positions.
6. **Serialization of Matrix**:
   - Read the grid row by row to construct the cipher body.
7. **Salt Generation using Cipher Length**:
   - Encrypt plaintext length and generate salt by concatenating the encrypted length with random characters.
8. **Hash Generation**:
   - Compute the SHA-256 hash of the key, salt, and plaintext for integrity verification.
9. **Final Composition of Ciphertext**:
   - Combine the salt and cipher body to obtain the final ciphertext.
   - Output the ciphertext and its corresponding hash.

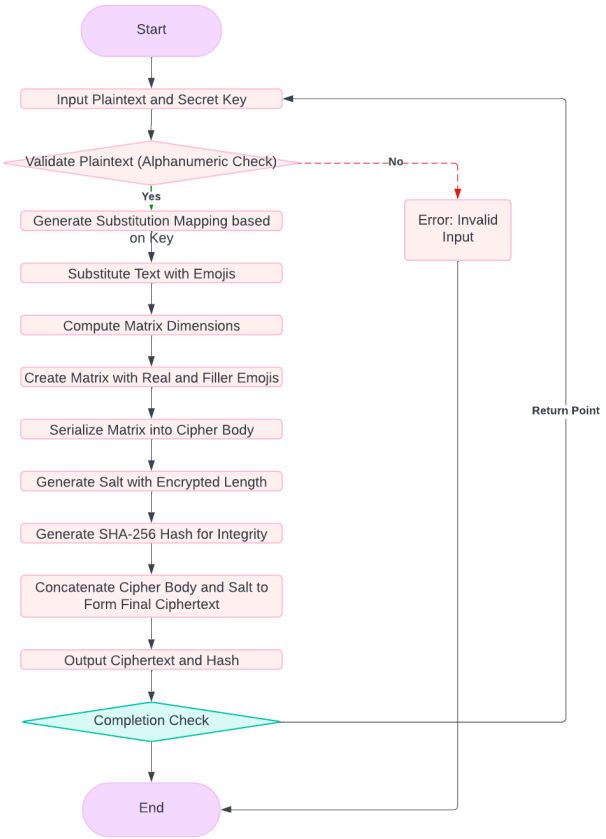

Fig. 4: **Encryption Process Flowchart**

B. Decryption Process

1. **Extract Salt**:
   - Extract the last 20 characters of the ciphertext as the salt.
   - Decrypt the first 8 characters of the salt to obtain the plaintext length.
2. **Grid Dimension Recalculation**:
   - Recalculate the number of rows and columns of the grid using the decrypted plaintext length and the expansion factor.
3. **Cipher Body Extraction**:
   - Remove the salt from the ciphertext to retrieve the cipher body.
   - Reconstruct the grid by reading the cipher body row by row.
4. **Plaintext Emoji Extraction**:
   - Determine the positions of the actual emojis based on the ASCII sum of the key.
   - Extract the specified number of emojis corresponding to the plaintext length.
5. **Reverse Substitution**:
   - Convert the extracted emojis back to their original alphanumeric characters using the reverse mapping.
6. **Integrity Verification**:
   - Rehash the message and compare it to the received hash to confirm no tampering has occurred.
7. **Output Plaintext**:
   - If integrity checks pass, output the decrypted message.

VI. Security Analysis

ERFC's security is evaluated based on its resistance to common attacks, effectiveness of its *confusion and diffusion* mechanisms, and the strength of its *avalanche effect*.

A. Confusion and Diffusion

- **Confusion**: ERFC employs a substitution mapping based on the secret key, which masks the mapping of plaintext characters to emojis. This approach creates difficulties in frequency analysis, as every character maps to a different, random emoji.
- **Diffusion**: By spreading the emojis throughout the entire grid and filling spaces with filler emojis, ERFC ensures that any change in the plaintext affects several areas of the ciphertext. The scattered distribution makes it challenging to establish patterns.

Example: Changing just one character in the plaintext results in different emojis and their positions in the grid, making it hard to discern patterns.

B. Avalanche Effect

Cryptography's *Avalanche Effect* refers to the phenomenon where a slight change in input (usually in plaintext

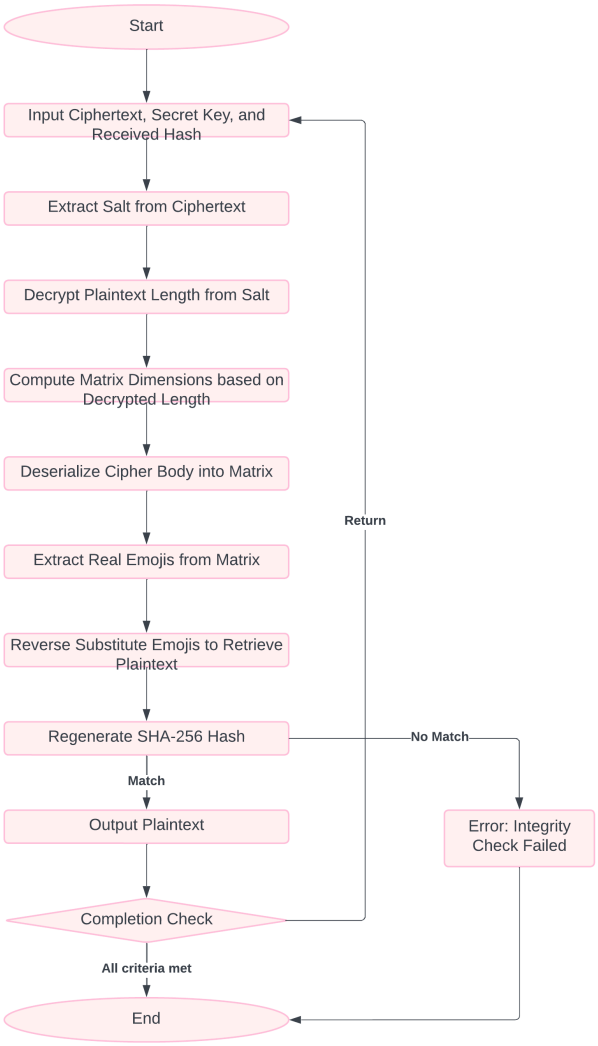

Fig. 5: **Decryption Process Flowchart**

and/or key) generates considerable variations in the output, i.e., the ciphertext. ERFC exhibits the avalanche property through the following factors:

- **Key Dependency**: A minor alteration to the key results in a change in the substitution mapping, which, in turn, causes a completely different set of emojis to represent the same input message.
- **Random Placement**: Even when the same key is used, modifying the plaintext size changes the grid size and the location of the emojis. This results in a different structure for the ciphertext.

Example: Two keys differing by just one character will result in a completely different set of emoji substitutions and ciphertext, demonstrating the avalanche effect.

C. *Resistance to General Attacks*

- **Frequency Analysis**: With the random nature of emoji substitution and the arbitrary placement of the grid, frequency analysis is not useful if the mapping of substitution is unknown.
- **Known-Plaintext Attacks**: Even if an attacker is aware of parts of the plaintext, the substitution mapping cannot be deduced without the key.
- **Brute Force Attacks**: The strength of ERFC depends on the complexity of the key. A strong key significantly makes brute-force attacks more difficult.
- **Integrity Attacks**: The use of a **SHA-256** hash ensures that any form of tampering with the ciphertext or salt can be detected during decryption.

Note: Although ERFC is more secure than the classical Rail Fence Cipher, it does not come close to modern algorithms such as **AES** or **RSA**. It is best used in applications where moderate security combined with user-friendly characteristics is desired.

VII. Performance Evaluation

Evaluating ERFC's performance involves analyzing its computational efficiency, scalability, and security compared to traditional ciphers.

A. *Efficiency Metrics*

- **Time Complexity**:
  - *Encryption*: $O(n)$, linear in plaintext length.
  - *Decryption*: $O(n)$, linear in ciphertext length.
- **Space Complexity**:
  - *Ciphertext Size*: Approximately $5\times$ the plaintext length, due to grid expansion.

Metric	ERFC	Rail Fence Cipher	AES-256
Time Complexity	*Linear*	*Linear*	*Linear*
Space Complexity	$\sim 5\times$ Plaintext	$\sim 1\times$ Plaintext	Fixed (block size)
Encryption Speed	**Fast**	Very Fast	Moderate
Decryption Speed	**Fast**	Very Fast	Moderate

TABLE I: **Efficiency Metrics Comparison**

B. *Comparison with Traditional Ciphers*

Feature	ERFC	Rail Fence Cipher	AES-256
Security Level	**Moderate**	Low	High
Ciphertext Size	$\sim 5\times$ Plaintext	$\sim 1\times$ Plaintext	Fixed (block size)
Speed	**Fast**	Very Fast	Moderate
Complexity	**Moderate**	Simple	Complex
Ease of Use	**User-Friendly**	Very User-Friendly	Requires Libraries
Resistance to Attacks	**Good for basic attacks**	Poor	Excellent
Key Dependency	**High**	Low	High
Implementation Effort	**Moderate**	Very Low	High
Cipher Type	*Substitution & Transposition*	*Transposition*	*Substitution & Permutation*

TABLE II: **Comparison of ERFC with Traditional Ciphers**

VIII. Applications and Use Cases

ERFC's combination of security and user-friendly features makes it suitable for various applications:

- **Personal Communication**: Encrypting messages on social media platforms using emojis for added privacy and engagement.

- **Educational Tools**: Teaching basic cryptography concepts through an engaging, emoji-based cipher.
- **Gaming**: Incorporating ERFC in games to encrypt in-game messages, adding a puzzle-solving layer for players.
- **Social Media Security**: Protecting sensitive information shared via emojis in posts or direct messages.
- **Artistic Projects**: Integrating cryptography with digital art using emojis to embed secret messages into artistic creations.

Limitations:

- ***Scalability***: Ciphertext expansion may be inefficient for large datasets.
- ***Standardization***: Limited adoption in professional security contexts due to a lack of standardization.

IX. Future Work

ERFC is a promising approach to secure emoji-based encryption, and several directions remain to be explored:

- **Enhanced Security Features**: Integrate modern cryptographic primitives to strengthen resistance against advanced attacks.
- **Scalability Improvement**: Optimize grid size and reduce ciphertext expansion for large-scale data transfers.
- **User Experience**: Develop user-friendly tools (mobile apps, web plugins) that seamlessly integrate ERFC into popular communication platforms.
- **Formal Security Analysis**: Conduct a rigorous cryptographic security proof to identify potential vulnerabilities and strengthen ERFC.
- **Standardization and Protocol Integration**: Work toward standardizing ERFC and integrating it with existing cryptographic protocols (e.g., TLS, SSH) for broader adoption.
- **Cross-Platform Compatibility**: Ensure ERFC's implementation is compatible across various operating systems and devices.

X. Conclusion

The Enhanced Rail Fence Cipher (ERFC) introduces a novel encryption method by embedding emojis within a transposition-based structure. ERFC offers a balance between security and user engagement by providing a *key-dependent substitution mapping*, randomized grid placement, encrypted length embedded in salt, and SHA-256 integrity hashing.

Although ERFC does not match the strength of AES or RSA, it is ideal for contexts where **moderate security** and **user-friendliness** are paramount. Future enhancements will target scalability, formal security proofs, and broader integration into mainstream communication tools.

References

[1] K. Nahar and P. Chakraborty, "Improved Approach of Rail Fence for Enhancing Security," *International Journal of Innovative Technology and Exploring Engineering*, vol. 9, no. 9, pp. 583–585, 2020. https://www.ijitee.org/wp-content/uploads/papers/v9i9/I7637079920.pdf

[2] A. A. Kumar, S. Kiran, R. P. K. Reddy, and S. R. Devara, "A New Variant of Rail Fence Cipher using Hybrid Block-Swap Method," *International Research Journal of Engineering and Technology*, vol. 8, no. 7, pp. 1735–1739, 2021. https://www.irjet.net/archives/V8/i7/IRJET-V8I7299.pdf

[3] A. Banerjee, M. Hasan, and H. Kafle, "Secure Cryptosystem Using Randomized Rail Fence Cipher for Mobile Devices," in *Proc. Intelligent Computing*, AISC vol. 998, Springer, 2019, pp. 737–750. https://www.researchgate.net/publication/334301599_Secure_Cryptosystem_Using_Randomized_Rail_Fence_Cipher_for_Mobile_Devices

[4] F. O. Aranuwa, F. Olubodun, and D. Akinwumi, "Hybridized Model for Data Security Based on Security Hash Analysis (SHA-512) and Salting Techniques," *International Journal of Network Security & Its Applications*, vol. 14, no. 2, pp. 31–39, 2022. https://www.researchgate.net/publication/359951983_Hybridized_Model_for_Data_Security_Based_on_Security_Hash_Analysis_SHA_512_and_Salting_Techniques

[5] H. J. Ali, T. M. Jawad, and H. Zuhair, "Data Security Using Random Dynamic Salting and AES Based on Master-Slave Keys for Iraqi Dam Management System," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 23, no. 2, pp. 1018–1029, 2021. https://www.researchgate.net/publication/353753738_Data_security_using_random_dynamic_salting_and_AES_based_on_master-slave_keys_for_Iraqi_dam_management_system

[6] R. P. Naik, N. T. Courtois, and Ł. Grajek, "Optimizing SHA256 in Bitcoin Mining," in *Cryptology and Network Security*, Lecture Notes in Computer Science, vol. 8813, Springer, 2014, pp. 131–145. https://link.springer.com/content/pdf/10.1007/978-3-662-44893-9_12.pdf

[7] R. Roshdy, M. Fouad, and M. Aboul-Dahab, "Design and Implementation of a New Security Hash Algorithm Based on MD5 and SHA-256," *International Journal of Engineering Sciences & Emerging Technologies*, vol. 6, no. 1, pp. 29–36, 2013. https://www.researchgate.net/publication/311019269_DESIGN_AND_IMPLEMENTATION_A_NEW_SECURITY_HASH_ALGORITHM_BASED_ON_MD5_AND_SHA-256

[8] A. M. Abdullah, "Advanced Encryption Standard (AES) Algorithm to Encrypt and Decrypt Data," *Cryptography and Network Security*, vol. 16, pp. 1–11, 2017. https://www.researchgate.net/publication/317615794_Advanced_Encryption_Standard_AES_Algorithm_to_Encrypt_and_Decrypt_Data

[9] A. M. Abdullah, "A Review of Encryption and Decryption of Text Using the AES Algorithm," *ResearchGate Preprint*, 2022. https://www.researchgate.net/publication/379871849_A_Review_of_Encryption_and_Decryption_of_Text_Using_the_AES_Algorithm

[10] M. Abdullah and A. Abdullah, "Comparative Study of RSA Asymmetric Algorithm and AES," *Jurnal Edukasi dan Teknologi Pembelajaran*, vol. 1, no. 1, 2017. https://journal.unnes.ac.id/sju/edukom/article/view/57389