Journal of Research in Artificial Intelligence for Systems and Applications (RAISA)
2025, Vol. 01, No. 02, E-ISSN : 3090-3157

Original Research

Implementation of Layered Encryption Using Vigenère and Rail Fence Cipher on Live Chat Systems for Customer Data Security

M. Fathony Ramdhan

Departement of Information Technology, University of Jember

Abstract

The security of digital communication has become increasingly crucial with the growing use of instant messaging services. This study aims to design and implement a live chat system that applies layered encryption using two classical cryptographic algorithms, namely Vigenère Cipher and Rail Fence Cipher, to enhance message confidentiality. Messages sent by users are first encrypted using the ASCII-based Vigenère Cipher with mod 256, and then encrypted again using the Rail Fence Cipher with a zigzag pattern. The testing process considered several parameters such as message length (10–100 characters), variations in key length, number of rails (2–5), and the number of messages sent per minute. The results indicate that the combination of these two algorithms can maintain message security without affecting real-time communication performance. This approach demonstrates that layered encryption using classical algorithms is still relevant for enhancing the security of digital communications in small to medium-scale applications.

Keywords

Live Chat, Vigenère Cipher, Rail Fence Cipher, Layered Encryption, Data Security

1. Introduction

The rapid development of digital technology has significantly increased the need for real-time communication systems, especially to support interactions between customers and service providers. One of the most widely used forms of communication is live chat, which allows users and service providers to communicate directly and instantly. However, data exchange in live chat systems is highly vulnerable to security threats such as eavesdropping, hacking, and exploitation by unauthorized parties. This raises serious concerns regarding the confidentiality and integrity of customer data transmitted through such media. To address these issues, cryptography serves as an effective solution to protect data by converting messages into an unreadable format for anyone without the proper decryption key. Both classical and modern cryptography offer various encryption methods, each with its own advantages and limitations. Nevertheless, using a single encryption method is often not sufficient to withstand various types of increasingly complex and sophisticated cyberattacks. Therefore, a stronger approach is required by implementing layered encryption to significantly enhance data security.

This study proposes the implementation of a combination of two classical encryption methods, namely Vigenère Cipher as a substitution algorithm and Rail Fence Cipher as a transposition algorithm. This combination is expected to increase the level of complexity in the decryption process for unauthorized parties, thereby providing additional protection for data transmitted through the live chat system. In this context, the research focuses on designing and building a live chat

*Corresponding author: **M. Fathony Ramdhan**

Email addresses:
mfathonyramdhan@gmail.com (M. Fathony Ramdhan)

Received: 19 Juli 2025; **Accepted:** 25 Juli 2025; **Published:** 30 Juli 2025

application that applies layered encryption using these two algorithms, ensuring secure communication in real- time.

2. Research Method

2. 1 Data Collection

The researcher conducted an in-depth literature review by examining various scholarly sources, such as cryptography textbooks, research journals, scientific articles, previous theses, and reputable online sources discussing classical cryptography, particularly the Vigenère Cipher and Rail Fence Cipher algorithms. This review aimed to gain a comprehensive understanding of the operational principles of each algorithm, including the encryption and decryption processes, character manipulation mechanisms in text, and the mathematical aspects underlying both methods.

Additionally, the researcher analyzed the strengths and weaknesses of each algorithm from the perspective of cryptographic theory and their application in digital communication systems. The study specifically highlighted the potential weaknesses of using a single algorithm, such as vulnerability to frequency analysis or structural patterns within the ciphertext, which are often exploited by cryptanalysts to reveal the original message.

Through this review, the researcher identified the opportunity to combine the Vigenère Cipher as a substitution algorithm and the Rail Fence Cipher as a transposition algorithm to form a more robust layered encryption method. This combination is expected to complement each other and enhance the protection of data transmitted in the live chat system while increasing the complexity of cryptanalysis for unauthorized parties. The results of this literature review serve as an essential foundation for defining system requirements, designing the encryption-decryption mechanism, and establishing evaluation indicators during the system testing phase.

2. 2 Requirement Analysis

System requirements were identified, including both functional and non-functional requirements. The analysis results cover key features such as sending and receiving encrypted messages in real time, as well as the expected security specifications and system performance.

2. 3 System Design

The live chat system is designed using a web-based client-server architecture. This stage includes designing the database structure, user interface, client-side encryption- decryption flow, and server-side API endpoints. The design also involves the use of PHP and JavaScript programming languages as well as a MySQL database.

2. 4 Implementation

This stage covers the development of the system based on the previously created design. The implementation process includes creating database tables, coding the Vigenère and Rail Fence algorithms on the client side, and integrating client-server communication using AJAX. The system is tested locally using XAMPP.

2. 5 Testing

The system is tested to ensure that all functions work properly. The testing includes black-box testing, functional testing for the encryption and decryption processes, as well as performance testing by measuring encryption-decryption time and the system's response to message traffic.

2. 6 Research Tools

The tools and software used in this research are presented in the following table:

Table 3. List of Tools and Software Used in the Research

No	Type of Tool	Name / Specification	Usage
1	Programming Languages	PHP 8.2, HTML, CSS, JavaScript	Used to develop the backend and frontend of the system.
2	Database	MySQL	Stores user data and encrypted messages.
3	Code Editor	Visual Studio Code	Writing and managing program code.
4	Local Server	XAMPP	Running a local server for application testing.
5	Web Browser	Google Chrome	Running and testing the live chat application interface.
6	Stopwatch / Timer	PHP microtime() function	Measuring the duration of the encryption and decryption process.
7	Browser Developer Tools	Chrome Inspect Tools	Monitoring network traffic and debugging.

2. 7 Data Analysis Method

The data obtained in this research will be analyzed using a descriptive quantitative approach. The analysis focuses on several key aspects, namely:

1. The time required in the encryption and decryption process to measure the efficiency of the algorithms.
2. A comparison of the ciphertext forms produced by using a single algorithm and the combined algorithms to observe differences in encryption complexity.
3. An evaluation of the effectiveness of the double encryption method based on the success rate of decryption and its resistance to basic cryptanalysis

The results of this analysis are expected to provide an overview of the extent to which the combination of the Vigenère Cipher and Rail Fence Cipher algorithms can enhance data security in the live chat system.

3. Results and Analysis

3. 1 Implementation of the Live Chat System

This live chat system is designed using a client-server architecture, where the encryption process is carried out entirely on the server side after the message is received from the client. With this approach, messages sent by users are received by the server in plain text, then immediately encrypted using the predefined algorithm before being stored or forwarded to the recipient. On the recipient's side, the message is still in encrypted form and will be decrypted again by the server before being displayed to the user.

The combination of substitution and transposition increases the complexity of the resulting ciphertext and provides additional protection against cryptanalysis attacks. This layered encryption approach, implemented on the server side, ensures the integrity and confidentiality of messages during storage and transmission, while also facilitating centralized security control.

3. 2 Process layered Ecryption

The encryption stage in this system uses a layered encryption approach that combines two classical cryptographic algorithms, namely the Vigenère Cipher and the Rail Fence Cipher. The encryption process is performed server-side, where messages received from users are immediately encrypted by the server before being stored in the database. This approach aims to enhance the security of customer data during both transmission and storage.

1. Encryption Using the Vigenère Cipher

The first step in the encryption process is carried out using the Vigenère Cipher algorithm, a polyalphabetic substitution algorithm that uses a secret key to transform each character in the message into an encrypted character. Unlike the classical approach, which is limited to the letters A–Z, this implementation applies ASCII values to enable encryption of all characters, including lowercase letters, numbers, symbols, and spaces.

The following is the implementation of the encryption function in PHP

```
function vigenereEncryptMod256($plaintext, $key)
{
 $encrypted = '';
 $keyLength = strlen($key);
 for ($i = 0; $i < strlen($plaintext); $i++) {
 $p = ord($plaintext[$i]);
 $k = ord($key[$i % $keyLength]);
 $c = ($p + $k) % 256;
 $encrypted .= chr($c);
 }
 return $encrypted;}
```

The implementation of the Vigenère encryption, packaged in the function vigenereEncryptMod256() using the PHP programming language, is used to encrypt text messages with a modified Vigenère Cipher algorithm capable of handling all 8-bit ASCII characters. This function works by adding the ASCII value of each character in the message (plaintext) with the corresponding character from the key, and then processing the result using the modulo 256 operation to ensure that it remains within the valid ASCII character range.

2. Encryption Using the Rail fence Cipher

After the message is encrypted using the Vigenère Cipher algorithm, the next step is the application of the Rail Fence Cipher, which functions as a transposition algorithm. The Rail Fence Cipher rearranges the positions of characters in the ciphertext produced by the Vigenère Cipher based on a zig-zag pattern across a number of "rails" or rows. In this system implementation, the use of 3 rails in the Rail Fence Cipher is chosen as it provides a balanced configuration between encryption pattern complexity and implementation efficiency. With 3 rails, the zig-zag pattern is sufficient to significantly scramble the order of characters while remaining manageable for both encryption and decryption processes, whether performed manually or programmatically.

```
function railFenceEncrypt($text, $rails = 3)
{ if ($rails <= 1) return $text;
$fence = array_fill(0, $rails, '');
$rail = 0;
$direction = 1;
for ($i = 0; $i < strlen($text); $i++) {
$fence[$rail] .= $text[$i];
$rail += $direction;
if ($rail === 0 || $rail === $rails - 1) {
$direction *= -1; } }
return implode('', $fence); }
```

The railFenceEncrypt() function implements the Rail Fence Cipher algorithm with three rows (rails). This cipher works by rearranging the characters of the input ciphertext (the result of the previous encryption) into a zigzag pattern across the rows, and then concatenating the characters from each row to produce the final transposed ciphertext. This technique strengthens encryption security by adding an additional layer of character rearrangement.

The Rail Fence Cipher encryption process is carried out as follows:

1. The message is placed in a zigzag pattern across several rows (the number of rows equals the number of rails).
2. Once the entire message is placed, the characters from each row are read sequentially from top to bottom to form the final ciphertext.
3. For example, when using 3 rails, the arrangement of characters will form an up-and-down staircase-like pattern, and the final result will be the combination of characters from each rail arranged vertically

3. 3 Description Message

After the encryption process is carried out in layers using the ASCII-based Vigenère Cipher followed by the Rail Fence Cipher, the original message (plaintext) can be retrieved through a decryption process executed sequentially but in the reverse order of encryption. The decryption begins by reversing the character arrangement scrambled by the Rail Fence Cipher, and the result is then used as the input for decryption with the ASCII-based Vigenère Cipher algorithm. The ultimate goal of this process is to successfully recover the original message that had previously been secured.

```
function railFenceDecrypt($cipher, $rails = 3)
{ $len = strlen($cipher);
$rail = array_fill(0, $rails, array_fill(0, $len, "\n"));
$dir_down = null;
$row = 0;
$col = 0;
for ($i = 0; $i < $len; $i++) { if ($row == 0) {
$dir_down = true;
} elseif ($row == $rails - 1) {
$dir_down = false; }
il[$row][$col++] = '*';
$row += $dir_down ? 1 : -1; }
$index = 0;
for ($i = 0; $i < $rails; $i++) { for ($j = 0; $j < $len; $j++)
{ if ($rail[$i][$j] == '*' && $index < $len) {
$rail[$i][$j] = $cipher[$index++]; } } }
$result = '';
$row = 0;
$col = 0;
for ($i = 0; $i < $len; $i++) { if ($row == 0) {
$dir_down = true;
} elseif ($row == $rails - 1) {
$dir_down = false; }
if ($rail[$row][$col] != '*') {
$result .= $rail[$row][$col++];
$row += $dir_down ? 1 : -1; }
return $result; }
```

The implementation of the railFenceDecrypt() function in PHP is used to reverse the character transposition process of the Rail Fence Cipher with three rails. This function reconstructs the zigzag positions of the characters based on the ciphertext length and then places the characters back into their original positions. The result is the initial encrypted text, which is then ready to be further decrypted using the Vigenère algorithm to obtain the original message. This process represents the initial stage in layered decryption.

3. 4 System Testing

Testing was carried out to ensure that the developed live chat system functions properly, particularly in the encryption and decryption processes using the Vigenère and Rail Fence Cipher algorithms. The testing also covered overall system functionality and performance aspects.

1. Types of Testing

Several types of testing conducted in this study include:

a) Black-box Testing
This testing was performed by examining all system features without looking into the program code. The main focus was to ensure that the system responds correctly to inputs and produces the expected outputs.

b) Performance Testing (Semi-Automated)
This testing was carried out to measure the speed of the encryption and decryption processes using the microtime() function in PHP. The purpose of this test is to determine execution time with precision and to ensure that the system remains responsive when used in real-time conversations.

2. Testing Tools

System testing was carried out entirely using a browser, with the help of built-in features and additional scripts embedded in the code. Several tools and techniques used include:

a. Browser & Live Chat System Interface
Used to monitor the communication process between the client and the server, including:

- Sending and receiving message activities.

- The content of messages sent and received.
- Validation to ensure that stored data is not in plaintext form.

b. microtime() Function in PHP
Used to measure the duration of the encryption and decryption processes on the server side with precision. The measurement results are recorded directly in the system log or displayed in the browser as part of performance testing.

3. Testing Parameters

Testing was carried out using several parameter variations as follows:

a. Message Length
Experiments were conducted with messages ranging from 10 to 100 characters to observe the effect of message length on processing time.

b. Vigenère Key Variations
Keys of different lengths were used (short: 3 characters, medium: 8 characters, long: 16 characters) to examine the impact of key complexity.

c. Number of Messages per Minute
A simulation of sending and receiving messages was conducted within a one-minute timeframe to test system stability and responsiveness.

3. 5 System Testing Result

Based on the tests conducted with the specified parameters, the results obtained are as follows:

1. Encryption and Decryption Accuracy

The system successfully encrypted and decrypted all messages accurately, both for short and long messages (up to 2000 characters), as well as with various key length combinations.

2. Data security in the database

Each original message was encrypted and stored as ciphertext, then decrypted back to be compared with the initial message. The test results showed that all ciphertexts were successfully decrypted accurately, whether the messages were plain text, questions, numbers, or symbols. This proves that the encryption-decryption process works correctly and consistently, ensuring that data in the database is not stored in a directly readable form (plaintext), thereby increasing the security level of the stored information.

Table 2. Black-box Testing of Encryption and Decryption

No	Test Scenario	Test Case	Expected Result
1	Sending a message from user	User sends message “halo” with a short key	Message is encrypted and stored in the database as ciphertext
2	Receiving a message by CS	CS receives and decrypts the message	Message appears as “halo” after decryption
3	Encryption with long key	User sends a message with a key length of 16 characters	Message can still be encrypted and decrypted correctly
4	Encrypting a long message (100 characters)	User sends a message with a length of 100 characters	Encryption time remains < 5 ms and decryption result remains accurate
5	Database storage	Check the database contents after the message is sent	Content is stored as ciphertext, unreadable directly

3. Resistance to Cryptanalysis

This test aims to assess the system’s level of resistance to cryptanalysis attacks, which are attempts to break encrypted messages (ciphertext) without going through the legitimate decryption process. The test was conducted by simulating attacks based on various levels of attacker knowledge and capability, represented in five different scenarios.

4. System Performance

Processing time testing was carried out to determine how fast the system performs encryption and decryption based on message length and key complexity.

Table 3. Encryption and Decryption Time Testing

No	Message Length	Encryption Key	Key Length	Encryption Time	Decryption Time	Total Time	Status
1	characters 5	K3y#S3cur3@2024!P4ss	19 chars	0.016 ms	0.016 ms	0.032 ms	SUCC ESS
2	characters 20	Str0ng#S3cur1ty!K3y$2024	24 chars	0.025 ms	0.049 ms	0.074 ms	SUCC ESS

3	characters 50	Sup3r#S3cur3@K3y!2024P4ss	27 chars	0.052 ms	0.116 ms	0.0168 ms	SUCC ESS
4	characters 100	V3ry#L0ng@S3cur3!K3y$2024#P4ss	33 chars	0.088 ms	0.151 ms	0.239ms	SUCC ESS
5	characters 2500	L0ng#M3ss4g3@K3y!2024S3cur3	28 chars	1.739 ms	3.033 ms	4.772 ms	SUCC ESS

Table 3 shows the results of encryption and decryption time testing using a combination of Vigenère Cipher Mod 256 and Rail Fence Cipher. The testing was carried out with the microtime() function in PHP against messages of varying lengths and keys.

The results show that the process remains very fast— even for a message of 2,500 characters, it only takes about 4.7 milliseconds. This proves that the algorithms used are efficient and suitable for real-time systems such as live chat applications

4. Conclusion

Based on the analysis, design, and implementation of the live chat application system with double encryption using the Vigenère Cipher and Rail Fence Cipher algorithms, several conclusions can be drawn:

1. The system successfully implemented layered encryption and decryption, where messages sent by users were first encrypted using the Vigenère algorithm and then re-encrypted with Rail Fence before being stored and delivered to the recipient.
2. The double encryption feature enhances communication security, as messages cannot be read without knowledge of both methods and the parameters used (Vigenère key and Rail Fence rail count). This was proven by the test results, which showed that all data was stored in ciphertext form and could not be understood without the decryption process.
3. The system demonstrated stable and efficient performance, with relatively fast encryption and decryption times (< 5 ms) even when handling messages of varying lengths and higher parameter complexity levels.
4. By combining lightweight yet effective classical algorithms, this application provides a practical example that even simple cryptographic algorithms remain relevant for securing small to medium-scale communication, especially when combined strategically.

References

[1] Aryanti, A., & Mekongga, I. (2018). Implementation of Rivest Shamir Adleman (RSA) and Vigenère Cipher in web-based information system. E3S Web of Conferences, 10007.https://doi.org/10.1051/e3sconf/20183110007

[2] Hartanto, P., Nguyen, P. T., Muchtar, T., Hayadi, B. H., & Rahim, R. (2020). Application of Rail Fence Cipher of encrypting and decrypting text messages. Journal of Advanced Research in Dynamical and Control Systems, 12.https://doi.org/10.5373/JARDCS/V12I2/S20201094

[3] Hidayatulloh, M., & Insannudin, E. (n.d.). Enkripsi dan dekripsi menggunakan Vigenère Cipher ASCII Java. Teknik Informatika, UIN Sunan Gunung Djati Bandung.

[4] Stallings, W. (2017). Cryptography and network security: Principles and practice (7th ed.). Pearson.

[5] Forouzan, B. A. (2007). Cryptography and network security. McGraw-Hill.

[6] Munir, R. (2006). Pengantar kriptografi. Informatika Bandung.

[7] Kurniawan, A. (2015). Membangun aplikasi chat sederhana dengan PHP dan WebSocket. Andi Offset.

[8] Faris, A. M., Rijal, M., & Sa'ad, M. (2023). Keamanan data dengan super enkripsi: Gabungan Vigenère dan Rail Fence Cipher. Jurnal Media Komputer dan Sains, 11(2), 52–60.

[9] Zailani, R., Yuliansyah, H., & Kurniawan, A. (2023). Implementasi algoritma Rail Fence Cipher pada aplikasi chat Android untuk pengamanan pesan. Jurnal Sistem dan Informatika, 5(1), 61–69.