International Journal of Multidisciplinary Research and Growth Evaluation.

Developing a Practical Exercise on Secure Encryption and Decryption Using Character Transposition Algorithm

Dang Phan Thu Huong [1], Vuong Thuy Linh [2*]
[1] Faculty of Information Technology, University of Labour and Social Affairs, No. 43, Tran Duy Hung Street, TrungHoa Ward, CauGiay District, Hanoi City, Vietnam
[2] Master, Lecturer, Faculty of Information Technology, University of Labour and Social Affairs, Vietnam

* Corresponding Author: **Vuong Thuy Linh**

Article Info
ISSN (online): 2582-7138
Volume: 06
Issue: 02
March-April 2025
Received: 21-02-2025
Accepted: 18-03-2025
Page No: 1286-1289

Abstract
Information security plays a crucial role in the digital age, especially in communication and data exchange. To enhance students' understanding of encryption and decryption algorithms, this paper presents the development of a practical exercise applying the Rail Fence transposition cipher, which rearranges characters based on a virtual fence pattern. The practical exercise is implemented in MATLAB, allowing students to program, experiment, and visually verify the results. The exercise includes key steps such as algorithm design, data input, encryption-decryption processing, and result comparison to evaluate the algorithm's performance. In addition to mastering the principles of encryption, students are encouraged to analyze the security level of the algorithm to better understand its limitations and possible improvements in real-world applications. Experimental results demonstrate that this approach improves learning effectiveness, increases interaction between students and instructors, and provides a solid foundation for approaching more advanced encryption algorithms in the field of information security.

Keywords: Encryption, Decryption, Transposition, Information Security

1. Introduction

In today's digital era, information security plays a critical role across various domains such as communications, banking, e-commerce, and cybersecurity. Encryption and decryption techniques are essential in protecting data against unauthorized access and malicious attacks. Secure encryption not only ensures data integrity but also maintains the authenticity of information during transmission and storage [1].

There are two widely used types of encryption: symmetric and asymmetric encryption. Symmetric encryption uses the same key for both encryption and decryption, while asymmetric encryption employs a pair of public and private keys [2]. Commonly implemented encryption algorithms include AES, DES, RSA, and ECC, which are widely applied in real-world systems [3]. In particular, elliptic curve cryptography (ECC) is gaining increased attention due to its ability to offer high levels of security with smaller key sizes compared to RSA [4, 3, 5].

In the context of education and training in information security, developing practical exercises is vital for helping students grasp the fundamental principles of data encryption and decryption. These exercises provide hands-on insights into the mechanisms of classical encryption techniques such as the Caesar cipher and transposition ciphers, as well as modern algorithms like ECC [6, 7]. MATLAB is a powerful tool that supports simulation and implementation of encryption algorithms, enabling learners to verify and evaluate the performance of various encryption methods effectively [8].

This paper presents the development of practical exercises on secure encryption and decryption, with a focus on transposition cipher techniques. The content includes foundational theory and practical experiments to assess the effectiveness of each method. Ultimately, this work aims to enhance learners' understanding of encryption techniques and raise awareness about data security in today's increasingly digitalized environment [9, 10].

2. Problem Statement

This study aims to develop a MATLAB program for secure text encryption and decryption using the columnar transposition cipher with the secret key [3, 1, 5, 2, 6, 4]. The program should feature a user-friendly interface for input and output, displaying the original text, the encrypted text, and the decrypted text.

3. Objectives

This practical exercise is designed to help students approach and apply the columnar transposition algorithm in the process of encrypting and decrypting information. Upon completing the exercise, students are expected to achieve the following objectives:

Understand the principle of transposition encryption: Grasp how the columnar transposition algorithm works, including the process of arranging data into columns and reconstructing the text based on a secret key.

Implement encryption and decryption algorithms: Write MATLAB code to perform encryption and decryption using the columnar transposition method, ensuring both accuracy and efficiency.

Develop a simple user interface: Design an interface that allows text input, executes encryption/decryption processes, and displays results in an intuitive manner.

Analyze and evaluate the results: Compare the original text, the encrypted text, and the decrypted text to verify the correctness of the algorithm.

Raise awareness of data security: Understand the practical application of transposition ciphers, especially in protecting information within communication and data storage systems.

Through this practical exercise, students will not only reinforce their theoretical knowledge of encryption but also develop skills in programming, data processing, and evaluating the security level of an information protection method.

4. Tools and Software

To conduct the practical exercise on transposition encryption and decryption, students are required to use the following tools and software:

MATLAB: A powerful programming and simulation software that supports matrix processing, string manipulation, and graphical user interface (GUI) development.

MATLAB Editor: An integrated development environment (IDE) for writing, editing, and executing the source code of encryption and decryption algorithms.

MATLAB string-processing libraries: Includes functions such as reshape, mod, char, and double for text data manipulation.

Personal computer: Equipped with MATLAB and sufficient hardware specifications to efficiently execute encryption and decryption programs.

Reference materials on columnar transposition cipher: Textbooks, scientific papers, or other reference materials on transposition encryption to support the learning process.

Students should prepare a working environment with MATLAB and familiarize themselves with basic software operations before proceeding with the practical exercise.

5. Procedure

- **Step 1:** Initialize the working environment

Open MATLAB and create a new working directory to store the source code and input data.

Initialize the MATLAB Editor to write and run the program code.

- **Step 2:** Input the text to be encrypted

Design a user interface (GUI) or use input commands to receive the original text from the user.

Convert the text into a matrix format suitable for the columnar transposition algorithm.

- **Step 3:** Encrypt the text using the columnar transposition algorithm

Divide the text into columns based on the secret key [3, 1, 5, 2, 6, 4].

Rearrange the columns in the specified order to create the encrypted text.

Output the encrypted text on the screen and store it in the variable BanMa.

- **Step 4:** Decrypt the text

Input the encrypted text generated in the previous step.

Use the secret key to reorder the columns back to their original positions.

Combine the characters to reconstruct the original text.

- **Step 5:** Display and verify the results

Display the original text, the encrypted text, and the decrypted text for comparison.

Verify the correctness of the algorithm by ensuring that the decrypted text matches the original text.

- **Step 6:** Conclusion and evaluation

Students discuss the security level of the columnar transposition cipher.

Evaluate the strengths and weaknesses of the method and suggest possible improvements if applicable.

6. Code Implementation

```
function HUONG_LINH
    % Create user interface
    fig = figure('Name', 'COLUMN
PERMUTATION CIPHER', 'Position', [500 300
600 400]);
    % Input original text
    uicontrol('Style', 'text', 'Position',
[20 350 100 20], 'String', 'ORIGINAL
TEXT:', ...
              'FontWeight', 'bold',
'FontSize', 12, 'ForegroundColor', 'k');
    inputBox = uicontrol('Style', 'edit',
'Position', [130 350 400 30], 'String',
'', 'FontWeight', 'bold');
    % Display encrypted text
    uicontrol('Style', 'text', 'Position',
[20 270 100 20], 'String', 'ENCRYPTED:',
...
              'FontWeight', 'bold',
'FontSize', 12, 'ForegroundColor', 'k');
    encodedBox = uicontrol('Style',
'edit', 'Position', [130 270 400 30],
'String', '', ...
                           'Enable',
'inactive', 'FontWeight', 'bold');
    % Display decrypted text
    uicontrol('Style', 'text', 'Position',
[20 190 100 20], 'String', 'DECRYPTED:',
...
```

```
            'FontWeight', 'bold',
'FontSize', 12, 'ForegroundColor', 'k');
    decodedBox = uicontrol('Style',
'edit', 'Position', [130 190 400 30],
'String', '', ...
                           'Enable',
'inactive', 'FontWeight', 'bold');
    % Encrypt and decrypt buttons
    uicontrol('Style', 'pushbutton',
'Position', [150 120 100 40], 'String',
'ENCRYPT', ...
            'FontWeight', 'bold',
'FontSize', 12, 'ForegroundColor', 'k',
'Callback', @encrypt_callback);
    uicontrol('Style', 'pushbutton',
'Position', [350 120 100 40], 'String',
'DECRYPT', ...
            'FontWeight', 'bold',
'FontSize', 12, 'ForegroundColor', 'k',
'Callback', @decrypt_callback);
    % Clear all button
    uicontrol('Style', 'pushbutton',
'Position', [250 60 100 40], 'String',
'CLEAR ALL', ...
            'FontWeight', 'bold',
'FontSize', 12, 'ForegroundColor', 'k',
'Callback', @clear_callback);
    % Permutation key
    key = [3,1,5,2,6,4];
    % Encryption function
    function encrypt_callback(~, ~)
        plainText = upper(get(inputBox,
'String')); % Convert text to uppercase
        cipherText =
column_permutation_encrypt(plainText,
key);
        set(encodedBox, 'String',
cipherText);
    end
    % Decryption function
    function decrypt_callback(~, ~)
        cipherText = get(encodedBox,
'String');
        decryptedText =
column_permutation_decrypt(cipherText,
key);
        set(decodedBox, 'String',
decryptedText);
    end
    % Clear all data function
    function clear_callback(~, ~)
        set(inputBox, 'String', '');
        set(encodedBox, 'String', '');
        set(decodedBox, 'String', '');
    end
end
% Column permutation encryption function
function cipherText =
column_permutation_encrypt(plainText, key)
    keyLength = length(key);
    % Add end-of-string marker
    plainText = [plainText, '~'];
    % Calculate the padding length
    padLength = mod(-length(plainText),
keyLength);
    plainText = [plainText, repmat('X', 1,
padLength)];
    % Convert to matrix (row-wise)
    numRows = length(plainText) /
keyLength;
    matrix = reshape(plainText, keyLength,
numRows)';
    % Apply column permutation
    permutedMatrix = matrix(:, key);
    cipherText = reshape(permutedMatrix',
1, []);
end
% Column permutation decryption function
function decryptedText =
column_permutation_decrypt(cipherText,
key)
    keyLength = length(key);
    numRows = length(cipherText) /
keyLength;
    % Create decryption matrix
    matrix = reshape(cipherText,
keyLength, numRows)';
    % Apply inverse permutation
    [~, invKey] = sort(key);
    originalMatrix = matrix(:, invKey);
    decryptedText =
reshape(originalMatrix', 1, []);
    % Remove trailing padding characters
    decryptedText =
regexprep(decryptedText, '[X]+$', ''); %
Remove trailing X's
    decryptedText = erase(decryptedText,
'~'); % Remove the end marker
end
```

7. Results and Evaluation

7.1. Experimental Results

The columnar transposition cipher program has been implemented and tested experimentally on the MATLAB user interface.

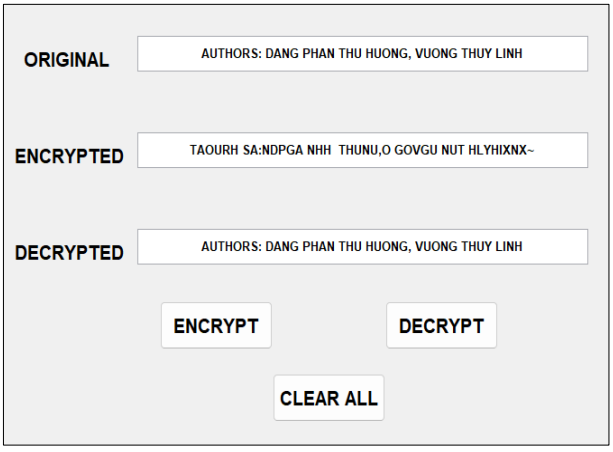

Fig 1

The main functions of the program include:

Inputting the original text and displaying the results of the encryption and decryption.

Accurate encryption and decryption according to the columnar transposition algorithm.

Providing a user-friendly interface that allows for easy interaction.

Example of an experimental result:

Original Text: "AUTHORS: DANG PHAN THU HUONG, VUONG THUY LINH"

Encrypted Text: "TAOURH SA:NDPGA NHIH THUNU,O GOVGU NUT HLYIHXNX~"

Decrypted Text: "AUTHORS: DANG PHAN THU HUONG, VUONG THUY LINH"

7.2. Performance Evaluation

The practical exercise is evaluated based on the following criteria:

Security: The columnar transposition algorithm provides a certain level of security by scrambling the order of characters in the text. However, due to the lack of integration with more complex encryption techniques, it may be vulnerable to frequency analysis attacks.

Accuracy: The encryption and decryption processes are implemented correctly according to the algorithm, ensuring that the decrypted output matches the original text. Experimental testing shows no data loss or character mismatch.

Automation: The system is capable of automatically performing encryption and decryption when the user inputs text and clicks a button. The program includes a function to clear all data, making it easy to conduct multiple trials.

Practical Applicability: The algorithm can be applied in simple systems that require internal information protection. However, for real-world applications, it should be combined with stronger algorithms such as AES or RSA to enhance security.

Performance: The program executes quickly even with long strings of text. High performance is achieved since the algorithm only performs simple permutations on a matrix.

8. Conclusion

In this paper, we studied and implemented the columnar transposition cipher algorithm using MATLAB and developed a practical encryption and decryption exercise for students. The program is designed with an intuitive user interface, allowing students to easily input text, perform encryption and decryption, and gain a clear understanding of how the algorithm functions in practice.

This exercise plays an important role in teaching and learning information security by introducing students to a basic, yet meaningful, encryption method in the field of data protection. Through programming and testing the algorithm, students can strengthen their algorithmic thinking, analyze information security approaches, and gain a deeper understanding of the principles behind encryption systems.

Experimental results show that the program operates correctly, ensuring that the decrypted text fully matches the original input. Additionally, the system includes a function to clear all data, allowing students to easily experiment with various datasets. The exercise also helps students recognize the strengths and limitations of the columnar transposition method, leading to discussions on possible improvements or integration with stronger algorithms such as AES or RSA to enhance security.

Furthermore, the program can be extended by allowing students to customize the transposition key, increasing its flexibility and practical applicability. Optimizing the algorithm's performance for processing larger texts is also a valuable direction for future development.

In summary, this practical exercise not only helps students understand the columnar transposition cipher but also provides a hands-on environment to develop programming skills, security-oriented thinking, and explore information protection methods in the digital age. It represents a crucial step in better preparing students for advanced research and real-world applications in the field of information security.

9. References

1. Dhanda SS, Singh B, Jindal P. Elliptic Curve Cryptography: A Software Implementation. In: Intelligent Computing and Communication Systems. Singapore: Springer; 2021. p. 143–8.
2. Majumder S, Ray S, Sadhukhan D, Khan MK, Dasgupta M. ECC-CoAP: Elliptic Curve Cryptography Based Constraint Application Protocol for Internet of Things. Wireless Personal Communications. 2020;116:1867–96.
3. Sedlacek V, Chi-Domínguez JJ, Tibouchi M. A Unified Approach to Elliptic Curve Special-Point-Based Attacks. In: Advances in Cryptology – ASIACRYPT 2021. Cham: Springer; 2021. p. 3–32.
4. Sharma PL, Gupta S, Monga H, Nayyar A, Gupta K, Sharma AK. TEXCEL: Text Encryption with Elliptic Curve Cryptography for Enhanced Security. Multimedia Tools and Applications. 2024.
5. Bernstein DJ, Lange T. Safe Curves for Elliptic-Curve Cryptography. In: Topics in Cryptology – CT-RSA 2024. Cham: Springer; 2024. p. 140–61.
6. Sharma PL, Gupta S, Monga H, Nayyar A, Gupta K, Sharma AK. A Novel Technique of Image Encryption through Projective Geometry and Elliptic Curve Cryptography. Multimedia Tools and Applications. 2024.
7. Majumder S, Ray S, Sadhukhan D, Khan MK, Dasgupta M. ECC-CoAP: Elliptic Curve Cryptography Based Constraint Application Protocol for Internet of Things. Wireless Personal Communications. 2020;116:1867–96.
8. Dhanda SS, Singh B, Jindal P. Elliptic Curve Cryptography: A Software Implementation. In: Intelligent Computing and Communication Systems. Singapore: Springer; 2021. p. 143–8.
9. Sedlacek V, Chi-Domínguez JJ, Tibouchi M. A Unified Approach to Elliptic Curve Special-Point-Based Attacks. In: Advances in Cryptology – ASIACRYPT 2021. Cham: Springer; 2021. p. 3–32.
10. Sharma PL, Gupta S, Monga H, Nayyar A, Gupta K, Sharma AK. TEXCEL: Text Encryption with Elliptic Curve Cryptography for Enhanced Security. Multimedia Tools and Applications. 2024.