International Journal of Multidisciplinary Research and Growth Evaluation.

Simulation of a Communication System Based on a Character Substitution Cipher

Tran Trung Kien
Faculty of Information Technology, University of Labour and Social Affairs, Hanoi City, Vietnam

* Corresponding Author: **Tran Trung Kien**

Article Info

ISSN (Online): 2582-7138
Impact Factor (RSIF): 8.04
Volume: 07
Issue: 02
Received: 08-01-2026
Accepted: 06-02-2026
Published: 03-03-2026
Page No: 42-50

Abstract
This paper presents the development and simulation of a communication system in a Local Area Network (LAN) environment based on the 8-bit ASCII character encoding standard combined with a monoalphabetic substitution cipher. The primary objective of the study is to provide a clear and systematic illustration of the fundamental principles of data encryption, transmission, and decryption in a computer network environment, while verifying the reliability of data exchange through the TCP/IP protocol.

The system is designed according to a client–server architecture and implemented in Python, utilizing the socket library to establish communication between two Windows-based computers within the same LAN. On the sender side (client), the input character string undergoes several processing stages: (1) conversion of each character into its 8-bit ASCII representation, (2) encryption using a predefined monoalphabetic substitution table, and (3) encapsulation of the encoded data into a byte stream for transmission via TCP. This procedure enables learners to clearly observe the relationship between textual data, binary representation, and the data encapsulation mechanism employed during network transmission.

On the receiver side (server), the system accepts the incoming byte stream through the established TCP connection and performs decryption using the corresponding inverse substitution table. The decrypted data are then converted from 8-bit ASCII codes back into the original character string, allowing comparison with the initial message to evaluate data integrity and transmission accuracy. The use of TCP ensures reliable communication with built-in error control and packet ordering, thereby highlighting the role of the transport layer in the network communication model.

Experimental results demonstrate that the system operates stably in a LAN environment, with transmitted data accurately recovered and no discrepancies observed under normal connection conditions. Although the proposed model employs a simple encryption technique, it provides significant pedagogical value for courses related to communication principles, data representation structures, computer networks, and fundamental cryptographic techniques. Furthermore, the system establishes a foundation for future extensions toward more advanced encryption schemes or the integration of error detection and correction mechanisms in subsequent studies.

Keywords: Substitution Cipher; 8-bit ASCII; TCP/IP; Communication System Simulation

1. Introduction

In the context of digital transformation and the rapid evolution of modern computer network systems, data have become a central asset for most organizations and individuals. Information exchange continuously occurs across network infrastructures of increasing scale, leading to growing demands for confidentiality, integrity, and reliability throughout the entire data transmission lifecycle. Therefore, equipping learners with a solid foundation in encryption principles and network communication mechanisms is not only of academic significance but also of clear practical value in the education and training of information technology professionals. In practical system deployment, modern cryptographic algorithms such as AES and RSA are commonly employed to ensure a high level of security. However, for educational purposes aimed at illustrating the fundamental principles of data transmission and cryptography, classical encryption techniques remain particularly valuable due to their structural simplicity, ease of analysis, and suitability for intuitive demonstration [3, 4].

The use of these methods enables learners to understand the essence of data transformation, the concepts of plaintext and ciphertext, encryption keys, and mapping rules before approaching more complex cryptographic systems.

Among classical cryptographic techniques, the substitution cipher is one of the oldest and most fundamental methods. According to the classification presented by Yu [3], classical ciphers can be broadly divided into two categories: substitution ciphers and transposition ciphers. While transposition ciphers rearrange the positions of characters in the plaintext, substitution ciphers map each plaintext character to another character according to a predefined rule. The simplicity of this one-to-one mapping allows students to clearly observe data transformation at the character level, while also providing a basis for analyzing the strengths and limitations of such cryptosystems.

Recent studies have continued to explore and extend substitution-based approaches in various directions. Kumar [1] proposed an ASCII value–based substitution technique to increase mapping complexity and expand the key space compared to traditional substitution tables. Lawal *et al*. [2] developed a 256-character substitution algorithm in hexadecimal form, enabling the utilization of the entire 8-bit character domain and enhancing security through an enlarged character set. In addition, Sharma *et al*. [7] presented an ASCII-based encryption approach, demonstrating the flexibility of this method in text processing and digital data transmission systems. Review studies [4] emphasize that although classical ciphers are no longer sufficiently secure for standalone use in modern security systems, they retain significant value for foundational research and education.

Parallel to encryption techniques, contemporary data transmission infrastructures primarily rely on the TCP/IP protocol suite. In this architecture, the Transmission Control Protocol (TCP) is a connection-oriented protocol that provides reliable data delivery through error control, packet sequencing, acknowledgment (ACK) mechanisms, and retransmission when errors occur. Consequently, TCP ensures that data arrive at their destination completely, in order, and without loss under normal network conditions. The implementation of client–server communication models using socket programming has been extensively studied and applied in both academic and practical contexts [6]. As noted by Hassan *et al*. [5], the Python programming language offers a rich ecosystem of libraries supporting TCP/IP networking, among which the socket library plays a fundamental role by enabling the development of low-level network communication applications with clear syntax and straightforward deployment.

From a pedagogical perspective, integrating a substitution cipher with the TCP/IP communication model in a LAN environment provides multiple benefits. First, students can observe the complete data processing cycle: from the original text, conversion to 8-bit ASCII codes, binary representation, substitution-based encryption, encapsulation into a byte stream, and transmission over a TCP connection. Second, the decryption process at the receiver side illustrates the inverse mapping mechanism and facilitates verification of message integrity. Third, deploying the system on two physical computers within the same LAN creates an experimental environment that closely resembles real-world operational conditions, rather than relying solely on theoretical simulation.

Furthermore, this approach helps learners clearly recognize functional layering in the communication model: the application layer performs encryption and decryption, the transport layer ensures reliable transmission, and the network layer handles packet routing. This separation of responsibilities constitutes a crucial foundation for understanding the architecture of modern communication systems as well as the operational principles of higher-level security protocols.

Based on the aforementioned theoretical and practical considerations, this paper develops a simulation model of a communication system in a LAN environment using the 8-bit ASCII standard combined with a substitution cipher, implemented in Python with the socket library. The model follows a client–server architecture and enables experimentation with the complete process of encryption, transmission, and decryption under realistic conditions. Through this system, the study not only clearly illustrates fundamental principles of data communication and classical cryptography but also verifies the stability and reliability of data exchange over TCP/IP in a LAN environment. The proposed framework provides an effective visual platform for teaching courses on communication principles, computer networks, and basic cryptographic techniques, while establishing a foundation for future research extensions.

2. System Model and Methodology

The objective of this study is not to develop a highly complex security system or to comprehensively simulate the physical characteristics of real-world communication channels (such as noise, attenuation, or man-in-the-middle attacks). Instead, the research focuses on constructing a visual, easy-to-implement, and reproducible simulation model suitable for laboratory environments. The core emphasis of the model is to clarify three fundamental aspects: (i) the representation of textual data using 8-bit ASCII encoding, (ii) the implementation of encryption and decryption using a monoalphabetic substitution cipher, and (iii) the transmission of data through the TCP/IP protocol within a Local Area Network (LAN).

The proposed model enables learners to observe the complete lifecycle of data: from its generation at the user side, conversion into binary form, execution of substitution-based encryption, encapsulation into a byte stream, transmission over a TCP channel, to decryption and restoration of the original plaintext at the receiver side. This approach integrates foundational concepts from classical cryptography and network communication into a unified experimental framework, thereby enhancing conceptual coherence and pedagogical effectiveness.

2.1. TCP/IP Transmission Model

2.1.1. Client–Server Architecture

The system is designed according to a client–server architecture based on the TCP/IP protocol stack. In this configuration:

1. Client functions as the transmitter, responsible for generating the source text data, converting characters into their 8-bit ASCII representations, and applying the monoalphabetic substitution cipher before sending the encoded data.
2. Server functions as the receiver, accepting data through a TCP connection, performing decryption using the corresponding inverse substitution table, and reconstructing the original plaintext to verify data

integrity.
3. The selection of TCP instead of UDP is motivated by the requirement for reliable data transmission. TCP provides the following mechanisms:
4. Establishment of a connection-oriented communication session.
5. Sequencing of data segments.
6. Acknowledgment (ACK) and retransmission in case of packet loss.
7. Guaranteed in-order and complete data delivery.

These features allow the system to focus on illustrating the encryption–decryption process at the application layer without being affected by transport-layer transmission errors under stable LAN conditions.
Implementation using Python's socket library enables direct representation of communication mechanisms between two independent processes running on separate computers within the same local network. Core operations such as socket creation, bind, listen, accept, connect, send, and recv are explicitly presented in the source code, helping learners understand the fundamental nature of client–server communication rather than relying on highly abstracted frameworks. Furthermore, Python's concise syntax and flexible string handling capabilities facilitate seamless integration of ASCII processing and byte-level encoding within a single program.

2.1.2. Transmission Procedure

The data transmission process in the proposed system is organized sequentially according to functional layers within the communication model, as described below.

Step 1. Server Initialization

The server creates a TCP socket, binds it to a local IP address and port number, and switches to the listening state (listen). Upon receiving a connection request from a client, the server accepts the request via the accept() function, thereby creating a dedicated socket for that specific communication session.

Step 2. Establishment of the TCP Connection

The client initializes its socket and sends a connection request to the server's IP address and port. The connection is established through TCP's three-way handshake mechanism:
The client sends a SYN packet.
The server responds with a SYN-ACK packet.
The client sends an ACK packet to confirm.
Upon completion of this procedure, a reliable full-duplex communication channel is established between the two parties.

Step 3. Conversion of Text to 8-bit ASCII Representation

The input text string at the client side is processed character by character. Each character is mapped to its corresponding integer value according to the extended ASCII table (range 0–255). This integer is then represented as an 8-bit binary value (1 byte).
This stage enables learners to clearly observe the transformation sequence:
Text → Character → Decimal value → 8-bit binary representation → Byte.

Step 4. Encryption Using a Monoalphabetic Substitution Table

The generated ASCII bytes are processed through a monoalphabetic substitution table. This table represents a one-to-one mapping (bijection) from the set {0, …, 255} onto itself according to a predefined permutation.
Let: x denote the original byte, f(x) denote the encrypted value.
The encryption process is defined as: $y = f(x)$
The one-to-one property guarantees the existence of an inverse mapping $f^{-1}(y)$, which is applied during decryption at the server side.

Step 5. Data Encapsulation and Transmission

The sequence of encrypted bytes is encapsulated into a Python bytes object and transmitted over the TCP connection using the send() or sendall() method. At the transport layer, TCP segments the data appropriately, assigns sequence numbers, and delivers the segments to the server.

Step 6. Reception and Decryption at the Server

The server receives the incoming byte stream using the recv() method. The received data are processed in reverse order:
Apply the inverse substitution table f−1(y)f^{-1}(y)f−1(y) to recover the original ASCII bytes.
Convert the 8-bit ASCII values back into their corresponding characters.
Concatenate the characters to reconstruct the complete text string.
Finally, the reconstructed text is compared with the original message to evaluate data integrity after transmission.

2.1.3. Pedagogical Significance of the Procedure

The above procedure is not merely a sequence of technical operations; rather, it constitutes a multilayered integrated model that effectively illustrates several fundamental concepts:
The functional separation between data processing at the application layer and data transmission at the transport layer.
The relationship between characters, ASCII encoding, and binary representation.
The operational principles of substitution ciphers and the mechanism of inverse mapping for decryption.
The role of TCP in ensuring reliable data delivery.
Through this structured workflow, learners can systematically and visually observe the entire data processing lifecycle—from data generation at the source, through encryption and network transmission, to final reconstruction at the receiver. This comprehensive perspective provides a solid foundation for extending the model toward more advanced scenarios, such as integrating error detection and correction mechanisms or incorporating modern cryptographic algorithms in future research.

2.2. Substitution Cipher Based on 8-bit ASCII

2.2.1. Code Structure

The encryption method employed in this study is a monoalphabetic substitution cipher defined over the entire 8-bit ASCII value domain. According to the classical cryptographic taxonomy, substitution ciphers operate by mapping each plaintext character to another character according to a fixed and predetermined rule throughout the transmission process [3, 4].
Unlike traditional examples that are typically restricted to the 26-letter alphabet, this study extends the mapping domain to the complete set of 256 values of extended ASCII. This

extension enables the processing not only of English alphabetic characters but also of digits, special symbols, and control characters within the 8-bit range, thereby creating an experimental environment that more closely reflects actual data representation in computer systems.

Let each plaintext character be represented by an ASCII value:

$x \in \{0,1, 2, \ldots, 255\}$

The encryption process is defined by the mapping: $y = f(x)$ where:

x is the ASCII value of the original character,

y is the ASCII value after encryption,

$f(\cdot)$ is a bijective (one-to-one and onto) mapping over the 256-element set.

The bijective property ensures that for each x there exists a unique y, and conversely, for each y there exists a unique x. Consequently, an inverse function $f^{-1}(\cdot)$ exists for the decryption process: $x = f^{-1}(y)$

From an implementation perspective, the substitution table can be represented in Python as either an array or a dictionary consisting of 256 elements, where the index or key corresponds to the original ASCII value and the associated value corresponds to the encrypted output. The inverse table is constructed by swapping key–value pairs from the encryption table.

This approach is consistent with ASCII value–based encryption methods presented in [1] and [7], and it can be extended to the full 256-character space as discussed in the work of Lawal *et al*. [2]. By utilizing the entire 8-bit domain, the model closely aligns with the actual representation of data in memory and its transmission over networks in the form of bytes.

2.2.2. Encoding and Decoding Procedure

The encryption and decryption processes are directly integrated into the TCP/IP data transmission workflow and executed at the application layer in two distinct phases.

A. Sender Side (Encoding Phase)

The encoding procedure at the client is performed sequentially as follows:

Conversion of characters to 8-bit ASCII

The input text string is processed character by character.

Each character is converted into its corresponding integer value according to the ASCII table.

This value is represented as a single byte (8 bits).

Substitution table mapping

Each ASCII value xxx is looked up in the substitution table to produce a new value y=f(x).

This operation generates a sequence of encrypted ASCII values.

Encapsulation into a byte sequence

The encrypted values y are concatenated into a Python bytes object.

This byte sequence constitutes the ciphertext transmitted over the TCP connection.

From a data processing perspective, this phase clearly illustrates the transformation sequence:

Text → ASCII (decimal) → 8-bit byte → Substitution mapping → Encrypted byte → TCP transmission

B. Receiver Side (Decoding Phase)

The decoding process at the server is performed in reverse order:

Reception of the encrypted byte stream:

The server uses the recv () function to receive data from the client.

The received data consist of a byte sequence representing the encrypted values y.

Inverse substitution mapping:

Each encrypted byte y is looked up in the inverse substitution table to recover the original value: $x = f^{-1}(y)$

Restoration of ASCII characters:

The recovered ASCII value x is converted back into its corresponding character.

The characters are concatenated to reconstruct the complete plaintext string.

If no transmission errors occur-owing to TCP's built-in reliability mechanisms-the decrypted text will exactly match the original source message. This direct equivalence provides a clear and practical means of verifying data integrity within the experimental environment.

2.2.3. Security Considerations

From a security perspective, a monoalphabetic substitution cipher does not provide strong protection against modern cryptanalytic attacks, particularly frequency analysis. Because each plaintext character is consistently mapped to the same ciphertext character, the statistical structure of natural language is partially preserved, thereby facilitating inference of the encryption key.

However, the objective of the proposed model is not to construct a robust cryptosystem for real-world deployment, but rather to illustrate the fundamental operational principles of classical encryption and decryption processes [3, 4]. In an educational context, the use of a monoalphabetic substitution cipher offers several advantages:

Conceptual simplicity and ease of implementation.

Clear demonstration of the bijective mapping property and the role of the encryption key.

Direct illustration of the relationship between characters and their binary representations.

Extending the mapping domain to the full 256-character ASCII space increases the permutation key space to 256!, which is significantly larger than that of systems restricted to the 26-letter alphabet. Although this expansion does not render the cipher resistant to advanced analytical techniques, it enriches the practical learning experience compared to traditional Caesar cipher examples [1, 2], and provides a meaningful basis for comparing classical ciphers with modern cryptographic algorithms in subsequent studies.

Therefore, the ASCII 8-bit–based substitution model serves not only as an illustrative tool but also as an integrative framework that connects data representation, mathematical mapping theory, and practical network communication within a unified experimental system.

2.3. Proposed Simulation Framework

2.3.1. System Architecture

The proposed simulation framework consists of three principal functional modules:

Client (Transmitter):
Input of textual data.
Conversion of characters into 8-bit ASCII representation.
Execution of monoalphabetic substitution encryption.
Transmission of encrypted data via TCP/IP.

Channel (TCP/IP in LAN):
Data transmission between two computers within a Local Area Network.
Assurance of reliable delivery through TCP mechanisms, including error control and acknowledgment.

Server (Receiver):
Reception of data through the socket interface.
Decryption using the inverse substitution table.
Conversion of ASCII values back into textual form.
Verification of data integrity by comparing reconstructed and original messages.
This modular architecture facilitates flexible extensibility. For instance, the monoalphabetic substitution component can be replaced with stronger cryptographic algorithms, or additional application-layer error detection mechanisms can be incorporated without altering the overall structural design.

2.3.2. Data Flow
The data flow within the proposed system can be summarized as follows:
Text input → 8-bit ASCII conversion → Substitution mapping → Byte encapsulation → TCP/IP transmission → Data reception → Inverse substitution mapping → ASCII restoration → Reconstruction of the original text.
This processing pipeline enables students to visually and systematically observe the complete transformation of data within a practical communication system.

2.3.3. Network Environment Model
The communication channel in this study is implemented within a Local Area Network using the TCP protocol. Owing to TCP's connection-oriented nature and packet acknowledgment mechanisms, data are guaranteed to reach the destination correctly and in proper order [6]. The use of Python's socket library simplifies the realization of the client–server model and aligns with existing analyses of TCP/IP library support in Python environments [5].
The LAN environment is deliberately selected to minimize the influence of complex physical-layer factors, thereby allowing the study to focus primarily on application-layer data processing and encryption mechanisms.

3. Experimental Setup and Results
This section presents the simulation environment, configuration parameters, and experimental results of the TCP/IP-based data transmission system employing 8-bit ASCII encoding combined with a substitution cipher. The primary objective is to develop a visual and instructional model that supports teaching fundamental encryption–decryption principles and data transmission processes within a Local Area Network.
It should be emphasized that the system is designed for pedagogical illustration and does not aim to evaluate real-world security strength or optimize network performance.

3.1. Simulation Environment
3.1.1. Software Configuration
The system is implemented using a client–server architecture over the TCP/IP protocol stack.
Client operations:
Convert the input character string into 8-bit ASCII representation.
Apply a monoalphabetic substitution cipher.
Encapsulate the encrypted data into a byte stream.
Transmit the byte stream to the server via a TCP socket.
Server operations:
Receive data from the TCP socket.
Perform decryption using the inverse substitution table.
Convert the 8-bit ASCII values back into the original character string.
The application is developed in Python using the socket library. It can be deployed either on a single machine (loopback testing) or on two separate computers connected within the same LAN or local Wi-Fi network.

3.1.2. Transmission Model
The system employs the TCP protocol, which ensures:
Reliable data transmission.
No packet loss under normal network conditions.
No bit-level errors at the application layer.
Consequently, the model focuses on illustrating:
The encryption and decryption process.
The structure of binary data representation.
The encapsulation and transmission of byte streams.
Channel noise and physical-layer errors are not simulated, as TCP inherently incorporates error detection, acknowledgment, and retransmission mechanisms.

3.1.3. Simulation Parameters
The simulation parameters are configured as follows:
Encoding scheme: 8-bit ASCII (0-255).
Encryption type: Monoalphabetic substitution cipher (fixed substitution table).
Test data size: 10 kB-100 kB.
Number of transmission trials: 30 repetitions.
Network environment: Internal LAN (100 Mbps).
Operating system: Windows.

Higher data loads are used solely to evaluate system stability rather than to conduct in-depth network performance benchmarking.

3.1.4. Graphical User Interface of the Simulation System
To support pedagogical objectives and enhance visual demonstration, the system is implemented with graphical user interfaces for both the transmitter and receiver sides.

A. Transmitter Interface (Client)
Figure 1 illustrates the simulation interface at the transmitter side. The interface provides the following functionalities:
Input of a plaintext message as a character string.
Display of ASCII values in both decimal and 8-bit binary formats.
Visualization of data after substitution-based encryption.
Observation of the generated byte stream prior to transmission.
Transmission of data via the TCP/IP connection.
The entire data transformation process is presented in a detailed, character-by-character manner, enabling learners to follow each step of the algorithm and clearly understand the sequence of operations involved in encryption and

transmission.

B. Receiver Interface (Server)

Figure 2 presents the simulation interface at the receiver side. The interface performs the following functions:

Receive the encrypted data stream from the client.

Display the received data in byte format.

Perform decryption using the inverse substitution table.

Convert binary ASCII values back into the corresponding text string.

Compare the transmitted and reconstructed data.

All intermediate results are displayed step by step, enabling learners to directly verify message integrity and observe the complete decryption and reconstruction process in a transparent and systematic manner.

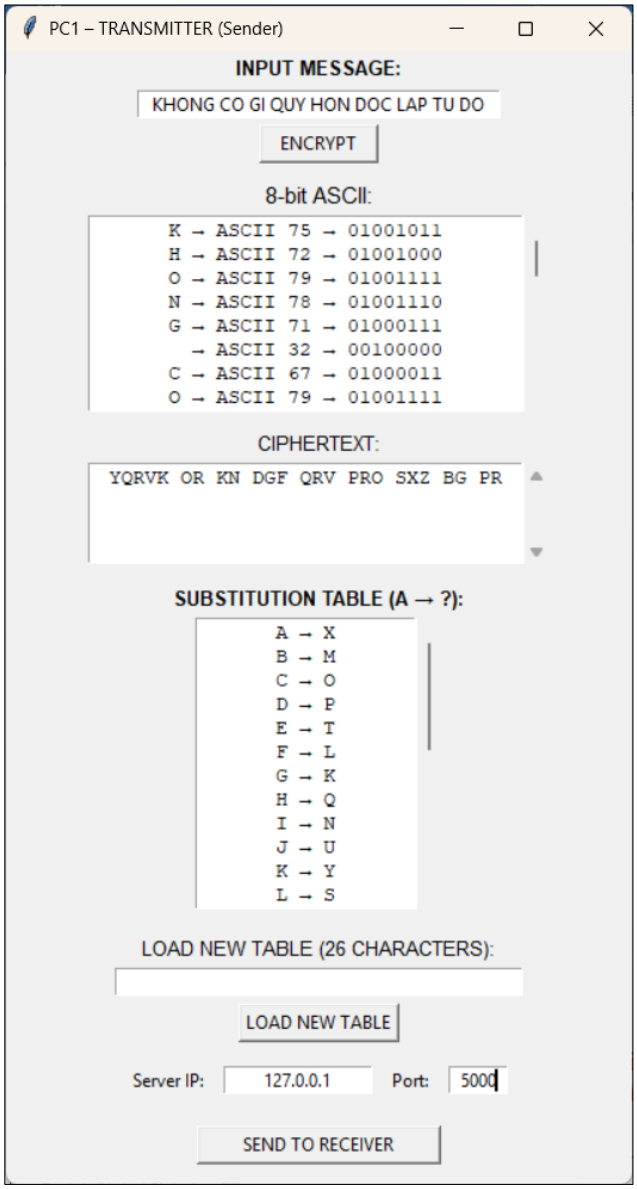

Fig 1: Graphical user interface of the transmitter implementing ASCII encoding and substitution cipher

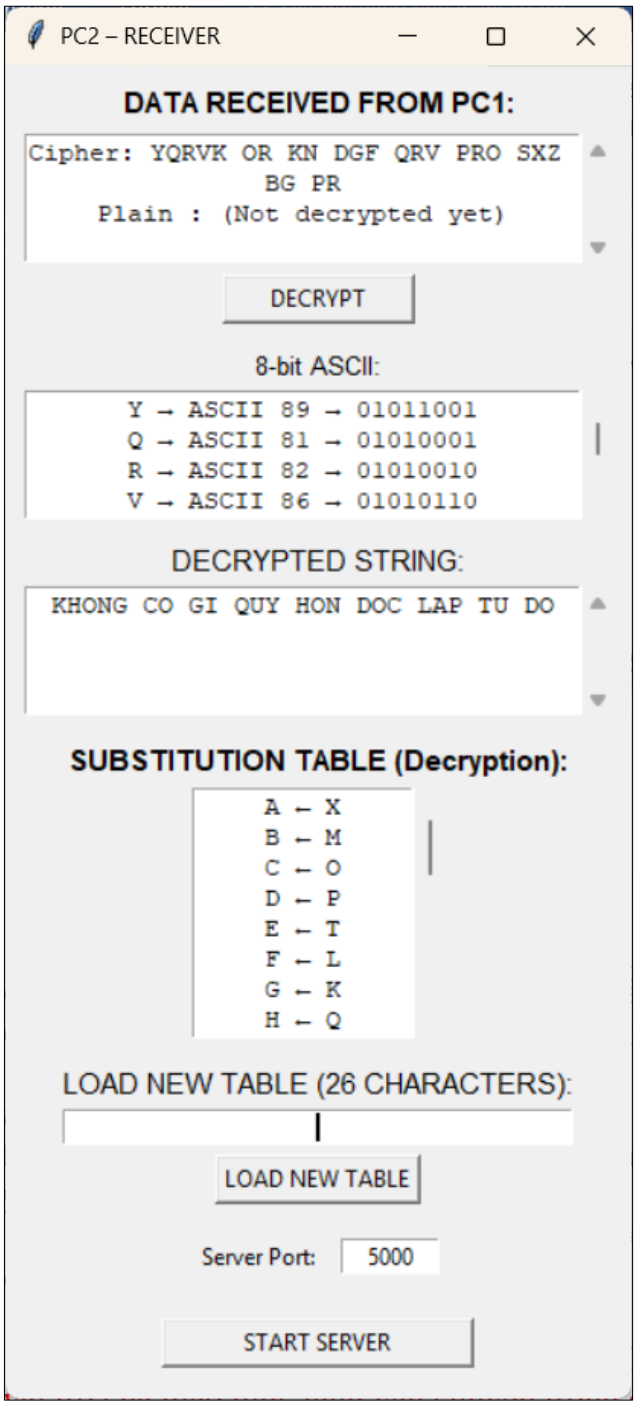

Fig 2: Graphical user interface of the receiver implementing substitution decoding and data recovery

C. Pedagogical Role of the Simulation Interface

The integration of a graphical user interface transforms the system into a visual instructional tool that effectively supports teaching and learning activities. Through the interface, students are able to:

Directly observe the transformation process from plaintext → ASCII representation → substitution-based encryption.

Understand the structure of 8-bit binary data.
Clearly perceive the mechanism of basic symmetric encryption.
Verify data integrity during transmission over TCP/IP.
Comprehend the role of the transport-layer protocol in ensuring reliable communication.
Consequently, the system functions as a virtual laboratory environment for courses on Communication Principles and Computer Networks, enabling learners to bridge theoretical concepts with practical experimentation in a structured and interactive manner.

3.2. Performance Metrics

The following evaluation metrics are employed to assess the behavior of the proposed system:

Data Integrity Rate

The ratio of data correctly recovered at the receiver side to the original transmitted data. This metric reflects the accuracy of the end-to-end transmission and decryption process.

Transmission Time

The total elapsed time from the moment the client initiates data transmission to the completion of reception and decryption at the server.

Encoding/Decoding Processing Time

The additional computational time introduced by the substitution-based encryption and decryption operations at the application layer.

Encoding Overhead

The potential increase in data size resulting from the encryption and encapsulation processes. This metric evaluates whether the applied substitution mechanism alters the original payload size during transmission.

3.3. Results and Discussion

3.3.1. Simulation Results

The system was tested using multiple uppercase character strings containing whitespace within an internal LAN environment. The updated illustration (PC1 – TRANSMITTER and PC2 – RECEIVER) demonstrates the transmission–encryption–decryption process for the message: "KHONG CO GI QUY HON DOC LAP TU DO"

A. Transmitter Side (PC1 – TRANSMITTER)

The input text string is entered into the INPUT MESSAGE field. After pressing the ENCRYPT button, the system performs the following operations:

1. Converts each character into its 8-bit ASCII representation.
2. Simultaneously displays: The decimal ASCII value. The corresponding 8-bit binary representation.
3. Examples displayed on the interface include:
   K → ASCII 75 → 01001011
   H → ASCII 72 → 01001000
   - → ASCII 79 → 01001111

   N → ASCII 78 → 01001110
   G → ASCII 71 → 01000111
   …

Subsequently, the monoalphabetic substitution algorithm (according to the mapping table shown in the SUBSTITUTION TABLE (A → ?) section) is applied to each character.
The resulting ciphertext is: YQRVK OR KN DGF QRV PRO SXZ BG PR
This ciphertext is encapsulated as a byte stream and transmitted via TCP to: Server IP: 127.0.0.1. Port: 5000.

B. Receiver Side (PC2 – RECEIVER)

1. After the server is initiated by pressing START SERVER, the encrypted message is displayed: YQRVK OR KN DGF QRV PRO SXZ BG PR
2. Prior to decryption, the data remain in ciphertext form.
3. When the DECRYPT button is pressed, the system performs:
4. Conversion of each received byte into its ASCII value.
5. Application of the inverse substitution table (SUBSTITUTION TABLE – Decryption).
6. Restoration of the original ASCII values.
7. Conversion back to the corresponding characters.
8. Examples displayed include:
9. Y → ASCII 89 → 01011001
10. Q → ASCII 81 → 01010001
11. R → ASCII 82 → 01010010
12. V → ASCII 86 → 01010110
13. The decrypted result is: KHONG CO GI QUY HON DOC LAP TU DO
14. The message is fully restored and matches the original plaintext exactly.

Across all experimental trials conducted within the internal LAN environment, no data discrepancies were observed. This outcome is consistent with the characteristics of TCP, which ensures reliable data transmission at the transport layer.

3.3.2. Comparison Between Plain Transmission and Substitution-Based Transmission

A. Plain ASCII Transmission (No Encryption)

1. When the client sends the ASCII string directly without applying encryption, the data displayed at the server matches exactly the data transmitted from the sender.
2. The Transmission Control Protocol (TCP) ensures:
3. No packet loss (reliable delivery)
4. Correct ordering of segments
5. No bit-level errors visible at the application layer
6. As a result, data integrity at the application level is effectively 100% under normal LAN conditions.

However, the message content remains fully readable in plaintext, meaning there is no confidentiality protection. Any entity capable of observing the traffic can directly interpret the transmitted information.

B. Transmission Using Substitution Cipher

1. When the substitution cipher is applied, the transmitted content appears in encrypted form (e.g., YQRVK OR KN DGF QRV PRO SXZ BG PR).
2. The message is no longer directly readable.
3. The string length remains unchanged.
4. The 8-bit ASCII byte structure is preserved.
5. No transmission errors are introduced.

After decryption at the receiver side, the original message is restored with 100% accuracy.
Unlike error-correcting code (ECC) models, this system does

not reduce the Bit Error Rate (BER), since TCP already ensures BER ≈ 0 at the application layer. Instead, the model demonstrates:

1. Basic confidentiality through content obfuscation
2. The complete pipeline of encryption → transmission → decryption
3. Preservation of data integrity under symmetric transformation

Thus, the comparison highlights that encryption affects semantic readability, not transmission reliability.

3.3.3. Cost–Benefit Analysis

A. Data Overhead

1 character → 1 character after encryption
No additional check bits
No packet size modification
Size overhead ≈ 0%
This is directly verified through the GUI display of transmitted byte counts on both client and server sides.
Therefore, the substitution process introduces no structural expansion of the transmitted data.

B. Processing Delay

The substitution algorithm has computational complexity: O(n), where n is the number of characters.
Characteristics:
Encryption time increases linearly with message length
Decryption time increases linearly
In a local LAN environment, processing delay is negligible compared to transmission time
Hence, the impact on total transmission time is statistically insignificant in practical classroom experiments.

C. Security Level

Limitations:
Vulnerable to frequency analysis
Fixed character mapping can be inferred with sufficient ciphertext samples
Not suitable for real-world secure systems

However, pedagogical advantages include:
Clear illustration of classical cryptographic principles
Demonstration of one-to-one mapping concepts
Foundational comparison basis for modern cryptographic systems such as: Advanced Encryption Standard (AES). RSA. Transport Layer Security (TLS).
In an educational context, the computational cost is effectively negligible, while the instructional value is significantly high.
Overall, the substitution-based transmission model provides a low-cost, high-clarity experimental framework for demonstrating core networking and cryptographic concepts without introducing unnecessary implementation complexity.

3.3.4. Scope of Application

In real-world TCP/IP systems, error detection and retransmission mechanisms are already integrated within the Transmission Control Protocol (TCP). Content confidentiality, however, is typically ensured through modern cryptographic frameworks such as: Transport Layer Security (TLS) / SSL. Advanced Encryption Standard (AES)-based encryption.
Therefore, the proposed model is not intended to replace modern security protocols. Instead, it serves as an instructional framework that illustrates the complete transformation pipeline: Plaintext → 8-bit ASCII → Ciphertext → Byte Stream → TCP Transmission → Decryption → Plaintext
This structured visualization enables students to:
Directly observe the difference between plaintext and ciphertext.
Understand how encrypted data is still transmitted as standard byte streams.
Implement a client–server architecture integrated with a basic encryption module.
Furthermore, the system is extensible and can be expanded to include: Vigenère Cipher. Data Encryption Standard (DES) / AES. RSA. Integration of SSL/TLS security layers.
Experimental results confirm:
Correctness of the substitution algorithm and its inverse mapping.
Data integrity during TCP/IP transmission.
System stability within a local area network environment.

3.3.5. Quantitative Performance Evaluation

A. Quantitative Metrics

1. The following metrics were observed in a controlled LAN environment:
   Transmission Time (T_t): Time from message dispatch to complete reception.
   Processing Time (T_p): Total time for encryption and decryption.
   Throughput (Th): Amount of successfully transmitted data per unit time.
2. Because the substitution cipher does not alter data size:
   Transmission Time depends primarily on TCP performance and message length.
   Processing Time increases linearly with the number of characters (O(n) complexity).
   Throughput remains stable in LAN conditions.

B. Analysis

Transmission Time increases approximately linearly with data size.
Processing Time accounts for only a small fraction of total delay.
No data corruption was recorded.
No packet loss occurred due to TCP's ACK and retransmission mechanisms.

C. Overall Assessment

Experimental results demonstrate that:
The substitution cipher introduces no size overhead.
Computational cost is minimal.
The system operates stably in a LAN environment.
Data recovery accuracy reaches 100%.
From a pedagogical perspective, the model functions as a virtual laboratory for courses such as: Principles of Data Communications. Computer Networks. Foundations of Cryptography
It helps learners clearly understand the relationship between: Data Representation ↔ Encryption ↔ TCP/IP Transmission ↔ Information Recovery
Thus, the system effectively bridges theoretical knowledge and practical implementation while maintaining simplicity, transparency, and extensibility.

4. Conclusion

This study successfully designed and simulated a TCP/IP-based data transmission system using 8-bit ASCII encoding combined with a substitution cipher under a client–server architecture. The primary contribution of the work does not lie in proposing a novel cryptographic algorithm, but in developing a visual, easy-to-implement, and extensible instructional model that effectively supports the teaching of data representation, classical cryptography, and computer networking principles. The system enables full observation of the transformation pipeline: Plaintext → 8-bit ASCII → Encryption → TCP/IP Transmission → Decryption → Message Recovery

Experimental results in a LAN environment demonstrate that the system operates stably and achieves 100% data recovery accuracy, ensured by the reliability mechanisms of the Transmission Control Protocol (TCP). The encryption and decryption processing time increases linearly with data size, yet remains small relative to total transmission time.

Moreover, the substitution process introduces no size overhead, thereby maintaining stable throughput.

Although the substitution cipher provides only basic confidentiality and is not suitable for real-world secure communication, the model offers significant pedagogical value by clearly illustrating the distinction between content security and transmission reliability.

Future work may extend the system by integrating stronger cryptographic algorithms such as Vigenère, Advanced Encryption Standard, or RSA, as well as simulating noisy channels and error-control mechanisms. Such enhancements would further increase both the research depth and educational impact of the model in the fields of cryptography and computer networks.

References

1. Kumar M. A secure substitution technique for text encryption and decryption based on ASCII value. Int J Innov Sci Res Technol. 2023;8(9). doi:10.5281/zenodo.8379740
2. Lawal I, Yusuf M, Usman AS, *et al*. A lightweight 256 hexadecimal character substitution Cipher algorithm. J Electr Syst Inf Technol. 2025;12:37. doi:10.1186/s43067-025-00217-w
3. Yu Z. Historical research on classification of Classical Cryptography. Theor Nat Sci. 2023;11:166-172. doi:10.54254/2753-8818/11/20230403
4. Thakkar B. A comprehensive study of substitution and transposition techniques in cryptographic systems. Int J Comput Sci Trends Technol. 2025;13(2).
5. Hassan GM, Hussien NM, Mohialden YM. Python TCP/IP libraries: a review. Int J Papier Advance Sci Rev. 2023;4(2):10-15. doi:10.47667/ijpasr.v4i2.202
6. Ahire P, Joshi A, Hajare S. Client-server communication using socket programming. Int J Innov Technol Comput Sci. 2023;15(4):50-62. doi:10.5815/ijitcs.2023.04.05
7. Sharma A, Bhatnagar A, Tak N, Sharma A, Avasthi J. An approach of substitution method based on ASCII codes in encryption technique. arXiv. 2013. Available from: https://arxiv.org/abs/1302.4510. doi:10.48550/arXiv.1302.4510

How to Cite This Article

Tran TK. Simulation of a communication system based on a character substitution cipher. International Journal of Multidisciplinary Research and Growth Evaluation. 2026;7(2):42–50.