A project report on

Enhancing the Security of Data Transmission in Networks by Integrating Error Correction Mechanisms with Advanced Cryptographic Protocols

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology in Computer Science Engineering Core

by

Tanmay Manoj Wani (21BCE5126)

Khadadiya Chinkal Mukeshkumar (21BCE5548)

Patel Smit Mineshkumar (21BCE5597)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

November, 2024

Enhancing the Security of Data Transmission in Networks by Integrating Error Correction Mechanisms with Advanced Cryptographic Protocols

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology in Computer Science Engineering Core

by

Tanmay Manoj Wani (21BCE5126)

Khadadiya Chinkal Mukeshkumar (21BCE5548)

Patel Smit Mineshkumar (21BCE5597)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

November, 2024

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

DECLARATION

I hereby declare that the thesis entitled "**Enhancing the Security of Data Transmission in Networks by Integrating Error Correction Mechanisms with Advanced Cryptographic Protocols**" submitted by **Tanmay Manoj Wani (21BCE5126),** for the award of the degree of Bachelor of Technology in Computer Science and Engineering, Vellore Institute of Technology, Chennai is a record of bonafide work carried out by me under the supervision of **Dr. Punitha K**.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Chennai

Date: 10/11/2024 Signature of the Candidate

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

School of Computer Science and Engineering

CERTIFICATE

This is to certify that the report entitled **“Enhancing the Security of Data Transmission in Networks by Integrating Error Correction Mechanisms with Advanced Cryptographic Protocols”** is prepared and submitted by **Tanmay Manoj Wani (21BCE5126)** to Vellore Institute of Technology, Chennai, in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in Computer Science and Engineering Core** is a bonafide record carried out under my guidance. The project fulfills the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma and the same is certified.

Signature of the Guide:
Name: Dr. Punitha K
Date: 15 /11/2024

Signature of the Examiner	Signature of the Examiner
Name:	Name:
Date: /11/2024	Date: /11/2024

Approved by the Head of Department,
Computer Science and Engineering Core

Name: **Dr. Nithyanandam P**

Date: 15/11/2024

(Seal of SCHOOL)

ABSTRACT

Data communication via networks is now a constituent element in most sectors of human existence; it ranges from broadcasting streams of media to telecommunications operations of transferring sensitive data information. With increasing dependency in almost all sectors of humanity on data communication, this comes along with a very imperative requirement for security beyond comparison ever before. Problems ranging from data corruption and network unauthorized access present one of the most critical issues threatening networks, and by association, leakage of critical organization data may have extremely hazardous outcomes. Most of the data security and error correction methods are in the form of Hamming codes, Low-Density Parity-Check (LDPC) codes, Reed-Solomon codes, and symmetric-key encryption algorithms like DES and AES, offering solutions but with a set of limitations such as complexity, latency, key management, and compatibility.

This project covers the above shortcomings by discussing some advanced techniques to secure data communication, focusing on error correction and encryption. It is meant to improve the integrity and security of data being transmitted over networks. Based on established approaches, developing a python program for this project will evaluate and adapt modern algorithms to optimize safety in transmitting data.

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to **Dr. Punitha K**,
Associate Professor, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai, for her constant guidance, continual encouragement, understanding; more than all, she taught me patience in my endeavor. My association with her is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of **Data Mining and Business Analytics**.

It is with gratitude that I would like to extend my thanks to the visionary leader **Dr. G. Viswanathan** our Honorable Chancellor, **Mr. Sankar Viswanathan**, **Dr. Sekar Viswanathan**, **Dr. G V Selvam** Vice Presidents, **Dr. Sandhya Pentareddy**, Executive Director, **Ms. Kadhambari S. Viswanathan**, Assistant Vice-President, **Dr. V. S. Kanchana Bhaaskaran** Vice-Chancellor, **Dr. T. Thyagarajan** Pro-Vice Chancellor, VIT Chennai and **Dr. P. K. Manoharan**, Additional Registrar for providing an exceptional working environment and inspiring all of us during the tenure of the course.

Special mention to **Dr. Ganesan R, Dean, Dr. Parvathi R**, Associate Dean Academics, **Dr. Geetha S**, Associate Dean Research, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai for spending their valuable time and efforts in sharing their knowledge and for helping us in every aspect.

In jubilant state, I express ingeniously my whole-hearted thanks to **Dr. P Nithyanandam,** Head of the Department, B.Tech. **Computer Science and Engineering Core** and the Project Coordinators for their valuable support and encouragement to take up and complete the thesis.

My sincere thanks to all the faculties and staffs at Vellore Institute of Technology, Chennai who helped me acquire the requisite knowledge. I would like to thank my parents for their support. It is indeed a pleasure to thank my friends who encouraged me to take up and complete this task.

Place: Chennai

Date: 15/11/2024 **Tanmay Manoj Wani**

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

1. **RS** – Reed-Solomon
2. **AES-GCM** – Advanced Encryption Standard in Galois/Counter Mode
3. **AES** – Advanced Encryption Standard
4. **RSA** – Rivest–Shamir–Adleman
5. **RSA-OAEP** – RSA Optimal Asymmetric Encryption Padding
6. **BER** – Bit Error Rate
7. **FER** – Frame Error Rate
8. **DMA** – Direct Memory Access
9. **DRM** – Digital Rights Management
10. **DPI** – Deep Packet Inspection
11. **5G** – Fifth Generation (telecommunications)
12. **IoT** – Internet of Things
13. **QoS** – Quality of Service (common in communication systems, although not used explicitly, it is implied in performance discussions)

Chapter 1
INTRODUCTION

1.1 INTRODUCTION TO TOPIC

Data communication underlies the heartbeat of the digital ecosystem-from personal entertainment to critical infrastructure. As digital networks emerge, so too does the need to protect data from unauthorized access and maintain its integrity across diverse applications. Data communication functions in many dimensions of everyday life-streaming videos, handling sensitive banking information, transmitting confidential communications from small networks. These scenarios hint at the very essence of data security in this new, more interconnected world.

The data on the move requires maximum security because of emergent threats like cyberattacks but also because of natural errors in transmission. Data traveling may be intercepted, altered, or even corrupted during transit by a hacker over a public or private network. Such attacks threaten privacy, intellectual property, and even national security. These data risks, therefore, have been guarded with encryption and error correction. Encryption refers to the conversion of the data into unreadable format as a way of blocking unauthorized access, while error correction detects and corrects errors that are usually introduced during the transmission.

This project is designed to enhance security and reliability of communication using the cryptography techniques-error correction combination. The study aims to design a security model that is Python-based, and such a model would aim at identifying possible combinations of techniques to present the best possible robust solution for securing data integrity and confidentiality. Here, two approaches are undertaken, one for improving data reliability in error correction techniques by correcting transmission errors and, in encryption techniques, by protecting the data from unauthorized access. This will result in the model of a safe and reliable system that would be apt to use in any kind of telecommunication application or in general, for secure network communication.

The project will also study how different encryption and error-correcting algorithms relate to efficiency, compliance, and compatibility in respect to modern network requirements. The presented model is optimized to have security accompanied by both speed and usability with adaptability. This research is important because it seeks to outline the way to attain data security at as many sites as possible, especially for those requiring high-speed and high-volume data exchanges, by constructing multi-layered defense against errors and attacks.

1.2 INTRODUCTION TO DATA ENCRYPTION

Data encryption is considered the basis of information security, providing confidentiality by converting readable data, known as plaintext, into an unreadable format for unauthorized users. Even though encryption dates to ancient times, the digital application has progressed tremendously in time and became a rather sophisticated tool in digital communications, online transactions, and secure data storage. Such encryption provides protection to the data that moves over networks which may be vulnerable ensuring that only authorized parties can decode and gain access to the original information using a decryption key.

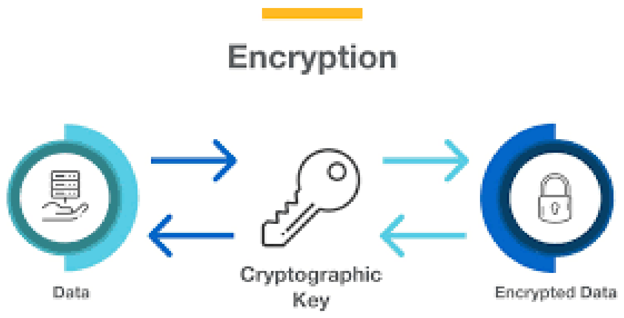


Fig 1.1 Data Encryption

The methods of encryption can be differentiated as symmetric and asymmetric algorithms. Such symmetric encryption, similar to its simple format in Advanced Encryption Standard (AES), uses one key for encryption and another for decryption. It so happens to be efficient for the bulk data encryption and is widely used on most secure communication channels. A kind of asymmetric encryption is RSA. It uses two keys for encryption and decryption purposes while securely sending it over an untrusted network. Both have their strengths and weaknesses, or rather their combination, called hybrid encryption, relies on the strengths of each.
Encryption algorithms are chosen based on factors such as speed of processing, security level, and especially for compatibility with applications or methods. One of these is AES, the most widely implemented scheme because it ensures that balance between security and performance. RSA is used in digital signatures and for secure key exchange. However, encryption does not guarantee data integrity; encryption gives confidentiality but cannot stop alteration of data. In fact, encryption is typically integrated with error correction algorithms toward building a more complete model of data security.

In this work, encryption is considered as the first layer to be developed for data transmission, eliminating unauthorized access and interfering with data transfer. This model will be adapted to optimize methods of encryption to harmonize them with error correction; thus, the approach devolves into multi facet data protection. The project will check the widely available encryption techniques and analyze level of security, and performance in real world applications. This project aims at the building of a safe data communication framework that is flexible based on requirements in networks for various types of safety of information using optimal encryption algorithms.

1.3 INTRODUCTION TO DATA ERROR CORRECTION

Data error correction is a technique that detects and corrects errors that may occur during data transmission. Communication channels are prone to noise, signal interference, among other forms of disruption that can distort data. Error correction ensures that data integrity is preserved even when these transmission issues arise. Applications such as medical data transfer, small network communication, and financial transactions require critical accuracy.

Error correcting codes ECCs introduce redundancy on transmitted data to allow the receiving end to detect and correct any errors without needing for retransmission. Among the types of ECCs, most widely used are Hamming codes, Reed-Solomon codes, and Low-Density Parity-Check (LDPC) codes. For example, single-bit error detection and correction capabilities make the Hamming codes suitable for low error rate environment. Another simple code class that can correct more than one error is Reed-Solomon codes. This is widely used in most digital media and CDs, hence is capable of correcting multiple errors and is suitable for noisy environments. LDPC codes offer very good performance in error correction in applications such as deep-space communication, and are known for being able to efficiently handle data transmissions with high error rates.

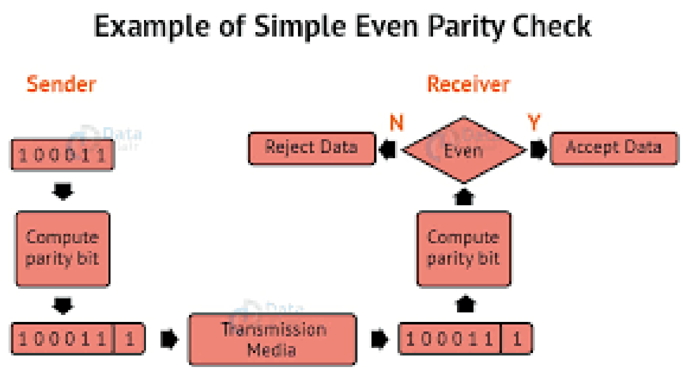


Fig 1.2 Parity Check Data Correction

In one-way communications, such as broadcasting, retransmission cannot be done; therefore, error correction plays an important role. ECCs are usually chosen regarding the transmission channel error rate, processing power, and degree of required data reliability. Error correction achieves data integrity but has no capability to stop unauthorized access; thus, the need for combining error correction with encryption in practice.

This project employs the error-correcting technique as a necessary component to protect data communication systems. Error-correcting techniques are carefully selected and incorporated into the model in order that no losses of the data or corruption of the data occur. Besides, high accuracy in terms of transmit/receive during the process is ensured. Simulations in network will be carried out and the various ECCs compared using their respective performances with a view to determining which one is apt, balancing between accuracy and efficiency. In our model, error correction will complement encryption as a means of ensuring the information transmitted between nodes is both of integrity and confidentiality and improves generally the dependability of the system.

1.4 PROJECT STATEMENT

This network-based data communication project integrates the encryption and error correction techniques with enhanced security and reliability, and the utmost focus lies in identifying an optimum technique combination; that will be developed into a security model of data integrity, confidentiality, and access. Today, modern encryption and error correction algorithms are only explored for the development of a Python-based model of high standards in data security.

This model addresses the problems of single-method approaches in that they always trade security for efficiency or cannot be adjusted according to complex network requirements. The project will comprise selection of encrypting and correcting error techniques in accordance with the best performance in specific data communication environments. This set of techniques will neutralize the threat from interception of corrupting data and unauthorized access, thus providing an efficient framework for secure data communication.

The model will be tested with key factors, to include but not limit to regulatory compliance, sensitivity of the data, and performance of the network. The project seeks a solution that will meet the security requirements in real-world scenarios and serve applications in such a sensitive industry as telecommunications and data exchange. The research contributing in the development of adaptable and resilient security solutions could resist the continually changing digital threats and keep high standards on the integrity of the data.

So, our problem statement is as follows:

"Design a hybrid model to enhance data transmission security and integrity by combining error correction and cryptographic protocols for smaller networks."

1.5 OBJECTIVE

- Objectives: This project aims at designing a secure data communication model that includes state-of-the-art encryption along with error correction techniques to ensure the integrity and secrecy of data while transmitting. The specific objectives of the project are as follows:

- Techniques Evaluation and Comparison: Analyze and Compare Encryption Techniques such as AES and RSA with Error Correction Techniques such as Reed-Solomon and LDPC in order to Determine Which Combinations of Such Techniques are Most Useful for Effective Data Security.

- Network-based Model Development: Design a Python framework that includes the best combination of encryption and error correction techniques. The model will then be a simulation of data transmission over a network with the applied methods to demonstrate their combined effectiveness.

- As such, performance of the model has to be assessed in light of key criteria such as the strength of security, efficiency, compatibility, compliance with relevant regulations, and adaptability to real-world applications. Thus, there is a good balance of criteria towards the goal of achieving practical security while keeping network communication safe.

- It is balanced both for security and for efficiency such that it is optimal about security, cost, and resource utilization. The model can therefore cater for diverse application needs with the aspect of fewer weaknesses. As data security and reliability are critical, this solution shall support critical applications.

Fig 1.3 Network Security

1.6 SCOPE OF THE PROJECT

The scope of this project is based on the selection, implementation, and assessment of encryption and error correction techniques for security data communication in a network communication-based security model for small-scale networks. The primary concern areas in this project are as follows:

Selection of Techniques: Identify and select the best possible encryption and error correction techniques that can be utilized for effective and secure data communication. The selection is based on the advantageous features and disadvantages of each technique to make it possible with a balance of efficiency, security, and adaptability.

Network Implementation: Develop a secure communication framework for small scale networks to mimic the process of data sending over a network. It must employ the techniques selected and be ready to be tested for performance and analyzed.

Performance Testing and Evaluation: Test and evaluate the model against dimensions such as regulatory compliance, security strength, latency, data integrity, and resource efficiency. Compare the model's flexibility for different network requirements, especially in high-security scenarios, for example, financial or medical data transfer.

Application Relevance: The outcome of this project will be quite relevant to any kind of industry involving telecommunications, banking, or healthcare where data communication security is a concern. This project aims to propose valuable insights into the design of robust data security solutions by providing a versatile and adaptable security model.

Fig 1.4 Secure Data Transmission

Chapter 2
BACKGROUND

2.1 INTRODUCTION

Data communications in the digital age today undertake data transfer over networks that stretch through tremendous distances and from a small local network to the larger internet. With the sudden growth in data volumes and an increased reliance on digital transactions, an ongoing demand is left for methods of communication that assure data integrity and confidentiality. This chapter covers the main objectives of data communication security, which clearly reflects and indicates how error correction and encryption complement each other for an effectively secure and reliable transmission environment.

Data communication security is an interdisciplinary approach that combines aspects of cryptography, network security, information theory, and computer science. The major goals include protection against unauthorized access to sensitive information and accuracy and availability of data during transmission. While encryption preserves data from unauthorized access, error correction ensures integrity within the data itself, correcting whatever errors are introduced by noise or other disturbances that occur during transmission. Both these approaches are crucial for applications such as secure messaging, online banking, health care exchange, and cloud computing.

Security and reliability in traditional network set-ups are approached as separate concerns where encryption solves the problem of privacy and error correction addresses integrity. However, with increasingly complex channels, this project identifies importance in hybrid approaches that incorporate these techniques. This research aims at testing combined effects of encryption and error correction by designing a model using Python to be useful for the accomplishment of security and resilience in data communication.

This chapter begins with an overview of the challenges and issues inherent in securing data communication. Challenges include ever-increasing levels of sophistication of cyberattacks, constraints of computational power and bandwidth, and regulatory compliance. We also describe limitations and vulnerabilities in existing security measures, especially in areas in which combined encryption and error correction techniques can provide stronger protection. Lastly, existing approaches and methods used in data communication security will be discussed. Key traditional areas such as traditional cryptographic methods and error-correcting codes are factored in, as well as the latest advancements in hybrid systems using both areas to offer a comprehensive security solution.

2.2 CHALLENGES AND ISSUES

Data communication faces a myriad of challenges and issues mainly because of the need to protect and ensure that information sent through various, often unsecured avenues, is kept secret. The ever-evolving cyber threat landscape is probably the greatest challenge of all. Cyber attackers come up with more sophisticated techniques - man-in-the-middle attacks, eavesdropping, and packet injection among others - to intercept or tamper with data as it is being transmitted. These are not countered fully by mere encryption methods, especially when attackers use advanced decryption methods. This calls for a strong combination of encryption for data confidentiality with error correction for data integrity.

Another key consideration is the problem of noise and interference in transmission channels. Specifically, in wireless communication, data packets tend to get lost or corrupted because of physical boundaries, electromagnetic interference, and environmental conditions. These errors cause incorrect information, and retuning of data can occur which, subsequently, decreases the efficiency of communication. Error correction techniques are designed to solve such an issue, although in practice, the actual implementation of error correction along with encryption is always computationally demanding. The problem of the design of a secure communications system is how to trade off computational intensity, speed of processing, and security strength.

Of course, regulatory compliance adds another layer of complexity to the use of data security models. Different industries and countries enforce specific data protection standards; examples include the General Data Protection Regulation (GDPR) in the EU, or the Health Insurance Portability and Accountability Act (HIPAA) in the United States. Such regulations call for a data security model that not only seeks to ensure confidentiality and integrity but also auditability, control over data, and accountability. Perhaps the most challenging aspect of adapting encryption and error correction techniques to a variety of regulatory frameworks will have to do with strict rules regarding data handling and storage.

Another problem is performance constraints, especially in resource-constrained environments. Complex encryptions and error checking also strain system resources, which may increase processing times, latency, and power consumption. For IoT networks, mobile devices, and communication applications requiring real-time communications, high-speed data processing is critical. Therefore, a balanced approach with robust security but without impacting performance drastically is highly required.

Scalability is the last challenge. With increased data volumes and network infrastructures, traditional data protection methods may not be efficient because they make heavy computational demands on the systems. Here, scalable solutions that can efficiently be applied to large networks without compromising security are required. These challenges sum up how an optimized approach by integrating encryption and error correction is needed to balance security, performance, and regulations on a modern data communication model.

2.3 EXISTING APPROACHES AND METHODS

Various approaches and methods have been developed concerning data communication security. These include encryption, mainly focusing on confidentiality and error correction, which presents methods to ensure the integrity of the data. Traditional encryption methods like AES and RSA are still in use because it has proven that sensitive data is indeed protected from unauthorized access. AES is a symmetric key encryption scheme, offering high speed and strength. It is therefore optimal for securing bulk data. RSA is an asymmetric technique where the key used to encrypt is normally used separately for decryption, hence supporting transfers over channels without compromising security. Although these techniques have tremendous effectivity toward secured transfer, they are not enabled to account for corrupted data during transmission. This is an area where error correction techniques are favorable.

ECCs comprise Hamming codes, Reed-Solomon codes, and LDPC codes, among others, that provide solutions capable of guaranteeing data integrity in noisily enclosed communication mediums. One such example is Reed-Solomon codes, used to correct multiple errors while transmitting data through storage devices and digital television. LDPC codes have good error-correcting capabilities in high-error-rate environments, such as deep-space communication, because they can process a large amount of data with very few errors. These codes are very useful in areas where the cost of retransmission is either too high or impossible, like in small network and real-time communications.

Recently, hybrid security models have been developed combining encryption with error correction to solve the confidentiality and integrity problems in data communication. Hybrid encryption models, using both symmetric and asymmetric encryption, have proved to offer greater security in data transmission. For instance, hybrid systems employ RSA to ensure secure key exchange followed by AES to encrypt data, so doing a balancing of the concern of security with efficiency needs of computation. If combined with ECCs, these ensure data integrity and confidentiality throughout its transmission.

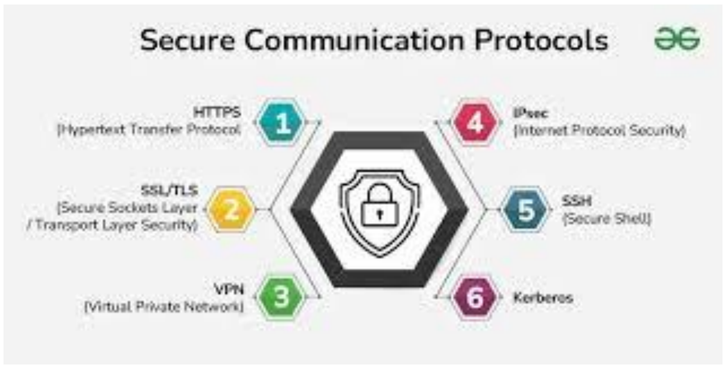

Secure Communication Protocols
HTTPS
(Hypertext Transfer Protocol)
1
SSL/TLS
(Secure Sockets Layer / Transport Layer Security)
2
VPN
(Virtual Private Network)
3
IPsec
(Internet Protocol Security)
4
SSH
(Secure Shell)
5
Kerberos
6

Fig 2.1 Existing Security Approaches

One of the promising application fields in data security is the deployment of error correction mechanisms in synergy with encryption, forward error correction, for that matter. Application of FEC techniques, including Turbo codes as well as convolutional codes, enables to introduce redundancy to allow error-corrective capability on the receiver's side while eliminating retransmissions. It is possible to combine such FEC techniques with encryption and, thus, achieve both secure protection against unauthorized access and resilience against errors for real-time applications, such as live video broadcasting.

Machine learning and AI are also entering data security in the form of anomaly detection for error correction and adaptive encryption methods. For example, AI-based models can dynamically update encryption strength or error correction depth depending upon the assessed risk of the channel through which the data is being transmitted. This way, dynamic optimum security can be ensured. Such approaches mark a trend towards intelligent adaptive security systems responding in real-time to both network conditions and levels of threats.

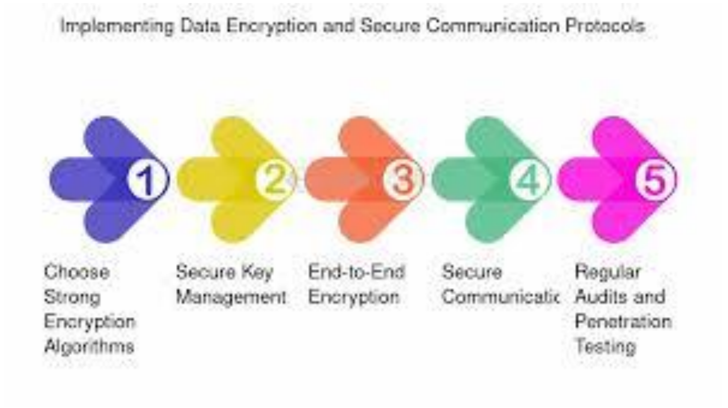

Implementing Data Encryption and Secure Communication Protocols
1
2
3
4
5
Choose Strong Encryption Algorithms
Secure Key Management
End-to-End Encryption
Secure Communicatic
Regular Audits and Penetration Testing

Fig 2.2 Data Encryption Flow

An integrated approach developed in this project relies on the virtues of these existing methods to create a model which interlaces encryption and error correction to provide a secure and efficient and complaint data communication model. Network simulations can be used as a simulation environment to simulate in detail and optimize the final model against modern data communication challenges.

Chapter 3
LITERATURE REVIEW

3.1 INTRODUCTION

These techniques are the backbone of error correction and encryption algorithms. The chapter proceeds to present these critical techniques ensuring that communications through a network remain in data integrity and security. Error correction or encryption algorithms alone have proven inadequate, especially against sophisticated attacks nowadays against isolated methods' weak points, since cyber threats evolve over time. Instead, there is a trend toward an increase of hybridizing these techniques as part of a multi-layered defense which greatly increases resilience against both accidental errors and intentional data breaches.

Error correction techniques are particularly helpful in ensuring data reliability over noisy communication links and thus often a prerequisite of telecommunication, small network communications, and wireless networks. Aided by detection as well as correction of errors while transmitting data, error correction techniques ensure increased reliability of data and are therefore very useful in businesses that heavily depend on the accuracy of their data to provide services. For their part, encryption algorithms play a crucial role in ensuring against unauthorized use of information through coding it in an unreadable form for unauthorized users while ensuring confidentiality and integrity of data.

The combination of these techniques will offer some unique advantages as well as disadvantages. For instance, ciphertext that is both encoded and error-protected will survive both transmission errors and possible interceptions. Applications of a very high value include secure military communications, financial transactions, and personal medical records. Finally, the following are several complexities caused by adding these techniques together: the increased demands on processing, optimized allocation of resources, and vulnerability if one technique introduces a weakness into the other.

This chapter shall review each technique's strengths and weaknesses and explore possible combinations that maximize both security and performance. Below is the popular error correction and encryption techniques currently in use, with discussions on the pros and cons of integrating them for optimal security. This analysis will further determine which of these combinations is most effective to handle the problems of communication of data, keeping in mind performance requirements and regulatory standards. Finally, this chapter provides insight into strategic applications of error correction and encryption, giving a basis to the selection of optimal techniques in use cases.

3.2 ERROR CORRECTION TECHNIQUES REVIEWED

The techniques of error correction prevent the channels from carrying noise and interference in such a manner that may result in deletion and alteration of the data. A few of these error correction methods have found widespread acceptance in the networks today, owing to unique capabilities, benefits, and trade-offs involved.

- Forward Error Correction (FEC): This method includes the techniques that add redundant information for the purpose of information transmission, enabling errors at the receiver end to be detected and corrected on-site, obviating the need for retransmission. Reed-Solomon codes are significant in FEC as they provide excellent resistance against bursty errors and, thus, are much more suitable for data transmission in a noisy channel, such as the small network or optical fiber communication. Another approach to the FEC is that of LDPC codes known as Low-Density Parity-Check. Such codes are widely applied in applications wherein utmost reliability is required without any error rate. 5G networks work on LDPC because of data integrity as a very high bandwidth and very low latency condition require it. The other type of FECs is convolution codes, that encode data into a stream that could be streamed in real-time, more appropriate for mobile communications or streaming.

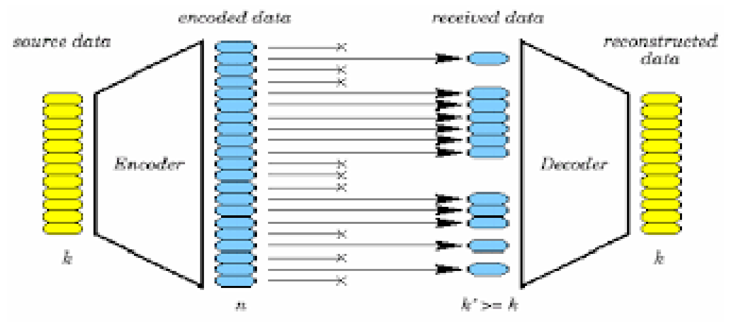


Fig 3.1 Forword Error Correction (FEC)

- Automatic Repeat Request (ARQ): ARQ relies on repeated transmission of detected error blocks during data transfer. Wherever the noise levels are low in the transmission media, it reduces the amount of redundant data needed for it. For applications that may be retransmitted with a higher success rate at reducing errors, ARQ is often used in combination with FEC. For example, in the TCP/IP network, ARQ is applied since reliable data transfer is important.

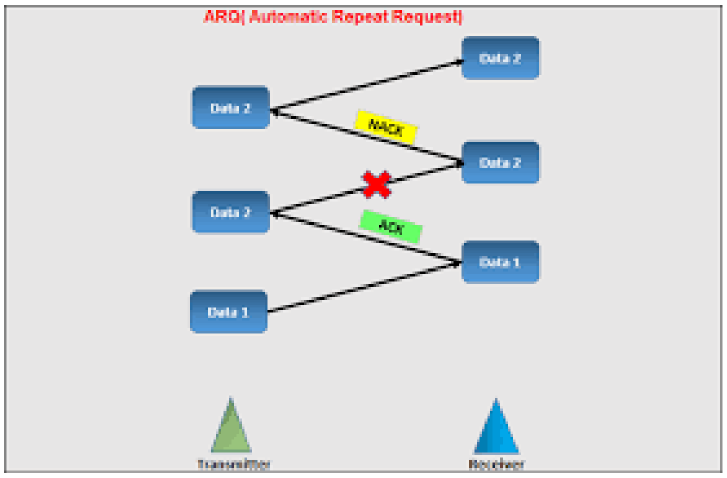


Fig 3.2 Automatic Repeat Request (ARQ)

- Checksum and Cyclic Redundancy Check (CRC) Checksums are simple, but they provide fundamental error detection because of the summing of the values of the transmitted bytes and validating them when they receive the data. CRC is more advanced, however, in that it can calculate a checksum of the given block of data, thus supporting a greater level of errors. CRC is mostly used by Ethernet protocols and storage devices where fast error detection is required, though no correction capabilities are needed.

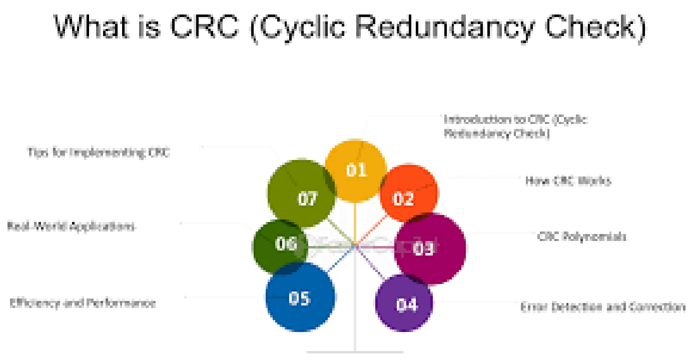

What is CRC (Cyclic Redundancy Check)
01
Introduction to CRC (Cyclic Redundancy Check)
02
How CRC Works
03
CRC Polynomials
04
Error Detection and Correction
05
Efficiency and Performance
06
Real-World Applications
07
Tips for Implementing CRC

Fig 3.3 Cyclic Redundancy Check (CRC)

Each type of error correction has pros and cons. On the one hand, with FEC, one can correct errors without retransmission but this takes a lot of computation and bandwidth. On the other hand, ARQ is resource efficient, though it could take too much time in environments of high error probabilities. CRC is lightweight and highly effective for error detection but cannot correct errors that are detected. This difference must be understood to use error correction codes effectively with encryption.

Feature	Forward Error Correction (FEC)	Automatic Repeat reQuest (ARQ)	Cyclic Redundancy Check (CRC)
Purpose	Detects and corrects errors at the receiver	Detects errors and requests retransmission if needed	Detects errors only; no correction mechanism
Error Handling	Corrects errors without retransmission	Requests retransmission upon detecting errors	Only detects errors, relying on external correction
Latency	Low latency, as no retransmission is needed	Higher latency due to retransmissions	Low latency; only detection is performed
Overhead	High, due to additional error-correcting bits	Moderate, due to retransmission overhead	Low, as it only uses a check sequence
Reliability in Noisy Channels	High, as it does not require a clean feedback channel	Moderate, as retransmissions depend on feedback	Low on its own; depends on retransmission or correction
Application Suitability	Suitable for real-time applications (e.g., streaming)	Suitable for applications that can handle delay (e.g., file transfer)	Suitable for applications needing lightweight error detection
Energy Consumption	Higher, due to complex error correction processing	Moderate, as retransmissions require additional power	Low, as only error detection processing is performed
Complexity	High, due to encoding and decoding requirements	Moderate, requiring retransmission protocols	Low, as it only needs to generate and check CRC values
Examples	Reed-Solomon, Turbo Codes, LDPC	TCP, ARQ protocols in data link layer	Ethernet frames, data storage devices

Table 1. FEC vs ARQ vs CRC

Let us understand the table:

1. Purpose and Error Handling: Every method has a different purpose with an associated way of handling error. FEC adds redundancy to data in such a way that the receiver can correct the errors without retransmission. This is very valuable for certain applications where retransmission may cost or be impossible, like real-time streaming or satellite communication. ARQ, on the other hand, only focuses on error detection and has a built-in retransmission request from the sender, and that in a reiterated cycle until data arrives without errors; therefore, ARQ is more appropriately used whenever there is a requirement for accuracy over time. CRC, on the other hand, is a purely error detection mechanism with no correction mechanism; the only difference is that it flags errors by checking the integrity of data but an independent method, such as ARQ, is required to correct the detected errors.

2. Latency: Latency is the delay that is introduced in data transmission. This is a low latency type because it corrects errors on the fly rather than asking for retransmission. Therefore, it makes suitable applications to the system such as voice or video streaming

where delay can easily interfere with the user experience. High latency occurs in ARQ due to retransmission. Every time an error is encountered, ARQ asks the source to retransmit the data, which causes subsequent delay. Thus, ARQ is more suitable for applications that can tolerate some delays, such as downloadable files. CRC, although lightweighted, introduces extremely minimal delay simply because it detects errors without having any mechanism of correcting them.

3. Overhead: Overhead are quite varied between FEC, ARQ and CRC. FEC is overhead-intensive because error-correcting bits are also transmitted along with data. Larger the size of these error correcting bits larger is the data being communicated and that utilizes more bandwidth. However, such overheads are justified wherever such transmissions must be error corrected and where retransmission is either not possible or not highly desirable. ARQ has a medium overhead since it does not add redundant information to the data in advance but may have the cost of retransmission, especially in noisy channels where repeated transmissions may get repeated. CRC has the lowest overhead since it just adds a checksum (usually several bits) to make an error detection that is suitable for lightweight error detection that does not affect the size of data much.

4. Reliability in Noisy Channels: FEC offers near 100 percent reliability in noisy environments because it corrects errors without any dependency on the sender's feedback. It is ideal for all those environments where channel noise is prevalent, like in satellite and mobile communication, because round-trip communication for retransmission is not possible. The round-trip dependency of ARQ hinges on clean feed back channels. With repeated requests, this can become a problem in a noisy environment. After all, CRC alone is not reliable in noisy channels. It can detect errors but cannot correct them. So, it waits for an upper layer such as ARQ to manage the error which makes it less effective as a standalone solution in the face of high noise.

5. Application Suitability FEC is appropriate for most real-time applications because delay and accuracy of the data are at a premium. It is applied in most streaming services, VoIP, and satellite communication applications. ARQ best suits applications where data accuracy is at a premium and can tolerate a particular delay. Such applications include data file transfers and email transmissions. CRC is used for fast error detection in applications where a check requires not necessarily immediate correction but to depend on other higher layers to act against the errors. The light weight of CRC finds application, for example, in hardware like checking data integrity in frames on an Ethernet or on a memory device.

6. Energy Consumption: Generally, due to the error correction algorithms requiring lots of computational resources for encoding and decoding, FEC generally consumes more power than error-detecting schemes. This might be the case of battery-based devices. ARQ has moderate energy consumption due to the retransmission mechanism; it will

add power consumption, which could be significant in poor communication channels where frequent transmissions are anticipated. CRC consumes the smallest amount of energy as it only detects errors while calculating the simple checksum.

7. Complexity: The FEC will be the most complex of the three, as it consists of rather complex encoding and decoding to introduce redundancy into the message and interpret that redundancy. The complexity tradeoff for robust error correction capabilities comes at a price-for example, in terms of needing additional processing power and memory. The complexity level of the ARQ is average because it relies on a protocol to use retransmission and must manage both ACKs and retransmission requests. CRC is the simplest one since only generation and check of a checksum is required. This form is so simple that it makes it highly efficient and easy to implement, which is why it is widely used in hardware and low-level protocols.

8. Examples: Each method is exemplified in different applications. FEC is used in applications that demand high reliability such as Reed-Solomon codes in CDs, DVDs and in a communication satellite and Turbo codes and LDPC in mobile networks. ARQ protocols are used in TCP for reliable data transfer over the Internet and are also used in wireless networks. CRC is used within Ethernet frames and in data storage devices to detect transmission errors in the message, making the systems data-integrity friendly.

In a nutshell, the three technologies-FEC, ARQ, and CRC-have different benefits that make them suitable for various types of applications based on the character of error-handling requirements, latency tolerance, overhead and complexity. FEC is apt for real-time communication systems where retransmission cannot be tolerated, ARQ for applications requiring accuracy at the cost of some delay, and CRC for simple lightweight error detection with simplicity and minimal overhead being of major importance. An appropriate error handling mechanism would depend on the need for reliability, efficiency, and the environment of operation for an application.

3.3 ENCRYPTION TECHNIQUES REVIEWED

Encryption is such an essential step in passing data through networks that have a tendency towards interceptor usage to ensure confidentiality and integrity. Below are some of the widely applied encryption algorithms, all developed to meet certain security, performance, or resource constraints.

- Advanced Encryption Standard (AES): AES is the gold standard in symmetric encryption. As secure as one can get with keys that are 128, 192, or 256 bits in length, it is certainly versatile enough to help adjust security with computational efficiency, which makes it a good candidate for applications as diverse as secure internet transactions and encrypted storage. AES tends to require large amounts of processing power and can put a lid on applications that need deployment on low-resources devices or in certain applications demanding real-time generation of encryption and decryption.

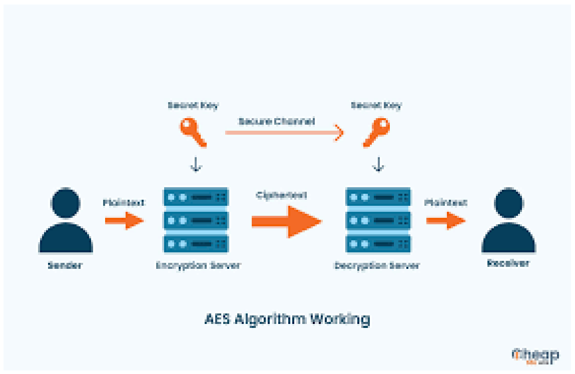


Fig 3.4 AES

- Rivest-Shamir-Adleman (RSA): Asymmetric encryption such as RSA has different security advantages, in particular concerning key management, using a public-private key pair. RSA is very applicable in applications in which the secure exchange of keys should take place over an untrusted network. The bad news here is that RSA is slower than symmetric algorithms, which makes it impractical to apply on large amounts of data. Still, the very strength of RSA against most kinds of cryptographic attacks makes email encryption and digital signatures widespread applications.

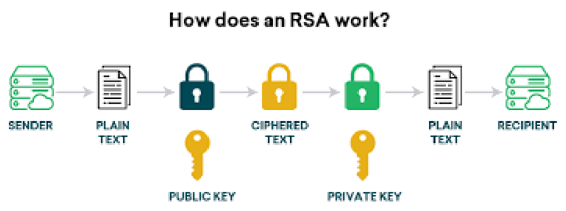


Fig 3.5 RSA

- Elliptic Curve Cryptography (ECC): ECC is a relatively new form of asymmetric encryption, providing a high security level with smaller key sizes, thus improving performance, and reducing memory needs. ECC is much more efficient for applications running in environments of low processing power, such as in mobile devices or IoT networks. ECC also supports digital signatures, encryptions, and secure key exchange, making it very versatile for modern network requirements.

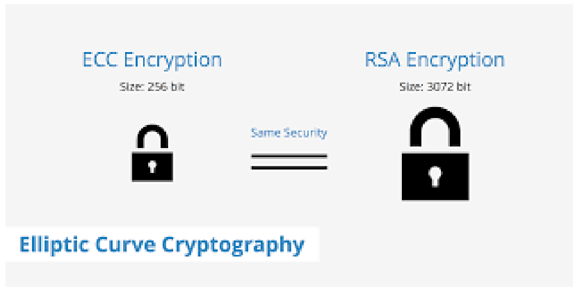


Fig 3.6 ECC vs RSA

- ChaCha20 and Blowfish: ChaCha20 is a recent symmetric encryption algorithm that strives for a balance between speed and security. Those best fits mobile usage applications since one of the reasons of developing such applications is the friendliness required at the processor level. Blowfish is another small input-size symmetric encryption algorithm, that was simple by design and strong at low memory cost. Although with a 64-bit block size it is no good for use in modern applications that require throughput, Blowfish is still a popular choice for specific applications, like secure storage of files.

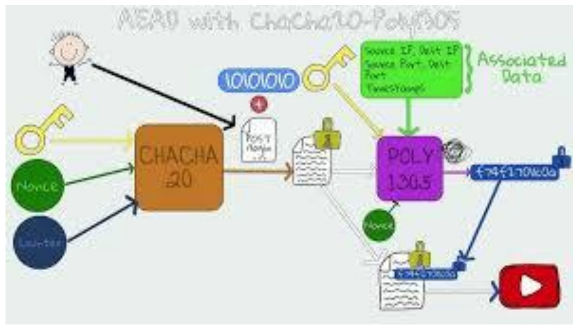


Fig 3.7 ChaCha-20

Strengths and weaknesses exist with each encryption technique: while one might be flexible, such as AES, the other is computationally efficient, such as ECC. While AES and ECC support very complex applications like VPN and mobile devices, any algorithm that computes a lot, such as RSA, has just the capacity to be effective in certain applications that can take advantage of the equivalent computational cost. Comparing these techniques is necessary to decide on their appropriateness for simultaneous error-correcting codes and encryption applications.

Feature	AES (Advanced Encryption Standard)	RSA (Rivest–Shamir–Adleman)	ECC (Elliptic Curve Cryptography)
Type	Symmetric encryption	Asymmetric encryption	Asymmetric encryption
Key Length	128, 192, or 256 bits	1024, 2048, 4096 bits	256 bits (offers equivalent security to larger RSA keys)
Security	Strong, with a small key size	Strong, but requires a larger key size	Strong, with shorter key length and less computational demand
Performance	Very fast, suitable for large data volumes	Slower, especially with larger keys	Faster than RSA due to smaller key size
Encryption/Decryption Speed	Very fast for both encryption and decryption	Slower encryption/decryption	Faster than RSA for both operations
Energy Consumption	Low, suitable for resource-constrained devices	High, especially for decryption	Lower than RSA, suitable for mobile devices
Use Case	Data-at-rest encryption, VPNs, secure file storage	Digital signatures, secure key exchange	Mobile applications, secure messaging
Strength Against Quantum Attacks	Vulnerable to quantum attacks on symmetric keys (though larger keys help)	Very vulnerable to quantum attacks (factorization)	Somewhat resistant, but research is ongoing
Complexity	Moderate, relatively easy to implement	High, requires prime factorization	Moderate, uses elliptic curves and modular arithmetic

Table 2. AES vs RSA vs ECC vs ChaCha-20

Let us understand the above given table:

1. Type of Encryption AES and ChaCha20 are symmetric encryption algorithms, in which a secret key is used both in encrypting and decrypting the data, thus they are highly effective to encrypt large volumes of data. RSA and ECC are asymmetric encryption algorithms where the encryption and decryption steps make use of a public and private key pair

respectively. An important distinction is between symmetric and asymmetric encryption, as symmetric encryption tends to be faster and more efficient while asymmetric encryption is used more often with secure key exchange, digital signatures, and the specification of a secure connection.

2. Key Length: Typically, Keys used for AES are either 128, 192, or 256 bits in length, providing robust security even with the smaller keys. On the other hand, a comparable level of security can be achieved with much larger key sizes for a RSA key i.e., 1024, 2048, or even 4096 bits. That comes at a higher computational overhead cost. ECC provides equivalent security using a very much shorter key length; for instance, a 256-bit ECC key is roughly comparable to a 3072-bit RSA key. Therefore, ECC could prove to be more efficient as well as a good candidate for being used in mobile applications. ChaCha20 uses a 256-bit key and is optimized for high-performance encryption. It has strong security.

3. Security: All four algorithms are considered secure against classical attacks. However, asymmetric algorithms RSA and ECC are vulnerable to future quantum attacks because they depend on the hardness of factorization and discrete logarithmic problems respectively. Though AES and ChaCha20 are susceptible to quantum attacks, introducing longer keys would slightly reduce the threats. Both AES and ChaCha20 do offer high security, as well as high resistance; but the nature of ChaCha20 being optimized against timing attacks means it finds best fit applications such as side channel attacks.

4. Performance: AES and ChaCha20 perform faster than RSA and ECC generally since the former are symmetric encryption algorithms that do not require complicated key-pair generation operations. AES is more efficient, especially with hardware acceleration, such as AES-NI, so the former is more suitable for encrypting a larger volume of data. ChaCha20 is very fast, especially in software implementation. However, it is optimized for mobile or other environments without access to hardware acceleration. ECC is faster than RSA because an ECC can achieve the same level of security by using shorter key lengths and based upon simpler computation, where RSA gets slower and slower with larger key sizes, especially because RSA basically depends on prime factorization that is computationally intensive.

5. Encryption/Decryption Speed: AES has high speeds on operations regarding encryption and decryption, mainly due to hardware support, making it broadly used for data-at-rest encryption and real-time secure communication. ChaCha20 is also known for its high-speed encryption and decryption, especially in software-only environments. RSA is relatively slower, especially while decrypting it, which might require higher processing power in case the key size is larger. ECC is faster compared to RSA as encryption and decryption methods as their key length is short and the mathematical calculations involved with them are simpler, and they are increasingly used in applications requiring low latency asymmetric encryption.

6. Energy Consumption: AES and ChaCha20 are low in energy consumption. Thus, they can be applied to resource-constrained devices such as mobile phones, IoT devices, and embedded systems. AES was optimized for hardware performance. ChaCha20 is optimized

for software performance and really depends on the kind of environment it is being used in. RSA is energy-hungry, considering large key sizes and complex calculations. decryption mainly consumes more energy. ECC is considered more energy-efficient than RSA because it can attain the same security level using much smaller key sizes and requires less computation, making it practical for use in mobile devices and for applications that are limited by power constraints.

7. Application: AES is used for encrypting data at rest, VPNs, secure file storage, and large data- processing applications due to its efficiency and security. RSA is often used for SSL/TLS certificates, secure email (PGP), and secure key exchange in web applications. ECC has its applications where the length of its keys is smaller, and the security is better; that is, where there's limited resources but a high demand for security in mobile applications, secure messaging, and digital signatures. ChaCha20 finds its place in applications such as secure messaging and VPNs due to its performance and resilience against timing attacks, making it suitable for encrypting HTTPS and the Signal Protocol, among others.

8. Resiliency Against Quantum Attacks: Quantum computing poses a significant threat to traditional asymmetric encryption methods. RSA, for instance, is very prone to quantum attack since quantum algorithms, such as Shor's algorithm, can break its security by factorizing it very efficiently. Similarly, ECC is vulnerable to quantum attacks, but due to their relatively small key sizes, quantum-resistant versions of them are currently under development. AES and ChaCha20 are symmetric encryption algorithms; therefore, because their respective operations are symmetric, they are more resistant to quantum attacks than RSA and ECC. However, larger key sizes such as AES-256 are advisable to increase resistance against probable quantum attacks further.

9. Complexity: AES and ChaCha20 are of average complexity, with rather simple algorithms. AES is based on rounds of substitutions, permutations, and depends on complexity for encryption; however, ChaCha20 depends on XOR operations, rotations, and add-rotate XOR (ARX) operations and is therefore very simple and highly efficiently adapted for computing in software. RSA relies for the complexity of its methods on the factorization of primes, which means large amounts of processing for bigger keys used in encryption and decryption operations. While ECC is slightly more involved than RSA in that it relies on elliptic curve mathematics and the performance of modular arithmetic, ECC is easier to use for the same level of security as RSA.

10. Applications AES has very broad applications running Wi-Fi security (WPA2), encrypts disks (BitLocker, File Vault), and supports TLS connections of secure web pages. RSA is commonly used to issue server SSL/TLS certificates and, via PGP, to encrypt email. ECC is used for digital signatures, secure mobile transactions, and secure messaging in applications like WhatsApp. Because ChaCha20 has a strong performance characteristic, it is used in the Signal Protocol, HTTPS, and VPNs due to fast encryption, resistance to

side-channel attacks, and suitability for high-speed safe communications.

To summarize, AES, RSA, ECC, and ChaCha20 serve different purposes and suit different environments based on their strengths and weaknesses. AES and ChaCha20 are the most appropriate for use when one require high speed symmetric encryption where performance is key, while RSA and ECC suit better in cases of key exchange and digital signatures as well as those cases requiring asymmetrical encryption. The process of selecting an algorithm would entail an ability to address specific needs calling for either speed, security, complexity, and the environment that enables smooth operation. AES, ChaCha20 are used for the encryption purpose due to its speed and fewness latency, but the RSA and ECC have preferable use cases for secure key exchange and signature generation because they embody a public-key characteristic.

3.4 PROS AND CONS OF COMBINATION OF TECHNIQUES

Layer security using both error correction and encryption offer a good security system for the data, but each method of combining them has certain advantages and challenges in terms of how the techniques are arrayed.

- Encryption Preceding Error Correction: Data is encrypted before it undergoes error correction to make sure that the data is not revealed to unauthorized people before error correction. The two together prove to be useful because they give a secured transmission that does not reveal data to potential attackers. However, Encrypt first may mask some of the error patterns and tends to reduce the effectiveness of error correction and even complicates error detection. In practice, noise added in encrypted form may tend to make it hard for the receiver to discern between the error or the legitimate data. Therefore, this combination is best suited for high-security channels with minimal noise.

- Error correction preceding encryption: Providing error correction to the raw data before encryption further facilitates a good error detection capability since the data before encryption is inherently easier to examine. The hybrid combination supports good integrity even at a high rate of error, since error correction is applied to the unencrypted version of the data. One of the main issues with this method is not as significant if the encryption key is compromised and there is no error correction provided to the potential attacker but shows up as corrected data. This method is suitable where error resilience is of utmost importance, and the encryption key is relatively secured.

- Simultaneous Encryption and Error Correction: Simultaneous encryption of the data while simultaneously correcting the errors in it will protect against both unauthorized access as well as real-time data transmission errors. It is promising, though high in processing power; its implementation may be tough due to increased computational complexity, but at times it results in better performance in higher security applications, making it an attractive option for systems with powerful processing capabilities.

- Hybrid Models: Hybrid models allow flexible combinations of AES with Reed-Solomon codes or ChaCha20 with LDPC, offering a trade-off between security and performance. These models can be instantiated to suit application requirements but introduce significant cost overheads often requiring special hardware and introducing latency. For critical applications that must require both security and error resilience simultaneously, this is the best hybrid approach.

This understanding leads to the identification of the best available solution for each specific application, ensuring that data will be both secure and robust. The trade-offs inherent in every combination need to be seriously evaluated according to the specified needs of an application for efficiency, security, and processing capabilities.

Chapter 4
PROPOSED METHODOLOGY

4.1 INTRODUCTION

This chapter introduces a structured approach taken for secure data communication within the small network networks using encryption and error correction techniques. For modern communications through small networks, data security and reliability must be ensured due to the vulnerability possessed by interference, noises, and even interceptions. So, given the vulnerability of the small networks in transmission, we are using a combination of encryption and error correction to have a robust communication channel with security over it. We strive, by encrypting the data first and then use with error correction, for two obvious goals that are data confidentiality to deny unauthorized access and data integrity against errors of transmission.

The chapter has portrayed a five-stage process: selecting techniques of error correction, selecting encryption techniques, integration of both into an all-inclusive model, simulation, and testing of the system, and then its implementation with continuous monitoring for maintenance of security standards over time. At every step, issues and challenges related to small network communications are included, such as an environment with noise unique and restriction on processing, and security threats also constantly evolve. The methodology is based on decreasing the vulnerability through evading several available weak links by choosing appropriate error correction techniques that must be integrated with encryption methods, thus avoiding any potential weak point for the attacker.

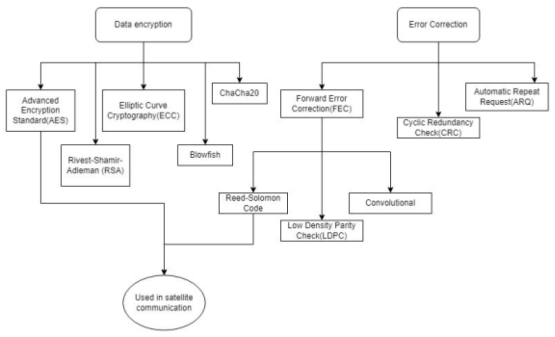

Data encryption
Error Correction
Advanced Encryption Standard(AES)
Rivest-Shamir-Adleman (RSA)
Elliptic Curve Cryptography(ECC)
Blowfish
ChaCha20
Forward Error Correction(FEC)
Cyclic Redundancy Check(CRC)
Automatic Repeat Request(ARQ)
Reed-Solomon Code
Low Density Parity Check(LDPC)
Convolutional
Used in satellite communication

Fig 4.1 Flowchart Showing Various Methods

With this layering, security is ensured through the reduction of mistaken errors and minimizing security holes. In addition, we ensure a balance between security and efficiency by choosing methods which are computationally feasible within the operating constraints of the small network. Encryption techniques will ensure that data cannot be intercepted during transmission, but error correction mechanisms will determine whether data received is accurate and restore such data if necessary. Finally, the overview of the introduction is concluded by summarizing how each approach is tailored for the high-risk, high-noise environment in small network networks and highlights the need for a combination approach to meet reliability and confidentiality requirements.

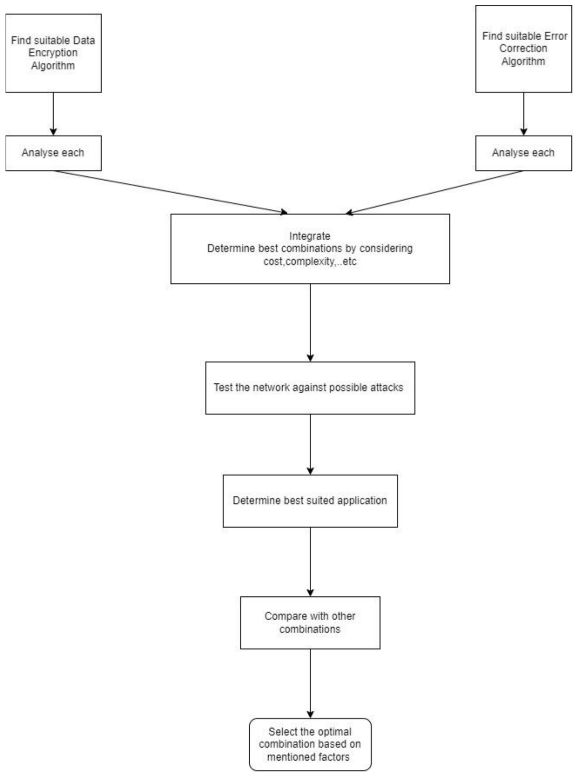

Find suitable Data Encryption Algorithm
Find suitable Error Correction Algorithm
Analyse each
Analyse each
Integrate
Determine best combinations by considering cost,complexity,..etc
Test the network against possible attacks
Determine best suited application
Compare with other combinations
Select the optimal combination based on mentioned factors

Fig 4.2 Flowchart for choosing technique

4.2 SELECTION OF DATA ENCRYPTION TECHNIQUE

The selection of an appropriate encryption technique on a small network is crucial so that unauthorized access is not gained to the sensitive information being sent via small network communications. Small network communications include sensitive data information such as military or corporate data and personal ones that require good encryption. A proper encryption method that balances security with efficiency is necessary in order not to degrade small network resources to an extent that protection may be compromised. To this end, we consider some encryption methods, including AES and ECC, in that every one of them offers different advantages based on the requirements of the small network.

AES is a symmetric encryption standard by many counts, characterized by high security levels and efficiency in processing large volumes of data. It supports different key sizes up to 128, 192, or 256 bits and may be set according to the requirements of security thus it can be used in applications with different levels of sensitivity. Since AES is a symmetric algorithm, it does not require much computational resource and, therefore, it is best-suited for high-throughput environments like small network networks. However, it requires secure key exchange and management since both the sender and the receiver must share the same key.

ECC, on the other hand, offers high security with less key size, thereby reducing their computationally intensive load as well as the memory usage. This was especially very important for some resource-constrained environments that were really valuable for efficient processing; an example is in the small network hardware. Smaller key sizes offered by ECC are similarly secure but with very low computational requirements compared to RSA. Therefore, it would be more suitable for devices with limited processing capacity.

AES, RSA, and SHA
1
AES (Advanced Encryption Standard)
2
RSA (Rivest-Shamir-Adleman)
3
SHA (Secure Hash Algorithm)
4
Comparing AES, RSA, and SHA
5
Best Option

Fig 4.3 AES and RSA

Its choice also depends on the type of sensitivity of data along with processing power as well as vulnerabilities. For instance, AES would be preferred if there is the need to transfer huge data in a short time, while ECC would be optimum for light-weighted resource-constrained applications. The method of encryption ensures that the secret data reads out unreadable to unauthorized parties yet its decryption on reception occurs with no restrictions nor friction. Our method is based on the capabilities of AES and ECC, required by the operational necessities of the small network, which do not trade off with performance.

4.3 SELECTION DATA ERROR CORRECTION TECHNIQUE

The employment of error correction in the small network communications supports the elimination of errors that are transmitted in space because of the internal space noise and interference. We enhance the reliability as well as accuracy of the message at the destination through the incorporation of error correction techniques. In our methodology, we are choosing techniques of error corrections compatible with the selected encryption algorithm to prevent vulnerabilities that may arise from non-compatible security measures. Data integrity is ensured by effective error correction, which assumes special importance in applications where accuracy and precision are paramount: real-time navigation, imaging, or relaying critical messages, for instance.

Out of such appropriate error correction techniques, Reed-Solomon codes, and Low-Density Parity-Check (LDPC) codes are prominent due to their ability to recover well under harsh conditions as prevalent in a small network. Reed-Solomon codes are excellent for correcting burst errors, which happen very frequently within the small network channels due to the interference created from electromagnetic radiation and physical obstructions. Such codes rely on providing extra redundancy data to allow detection and correction of multiple error patterns and thus are amiable for multimedia as well as other data-sensitive applications.

LDPC is one of the most efficient error-correcting codes for high-bandwidth applications and low error rates. It achieves this by creating sparse matrices, thus giving a way to detect and correct errors with minimal bandwidth usage. LDPC has good error correction characteristics in addition to efficient bandwidth usage; therefore, it is ideal for hard-real-time applications, which demand low latency and reliability. Thus, based on requirements of transmission, Reed-Solomon or LDPC should be selected whether the burst-error resilience is a priority or the high throughput.

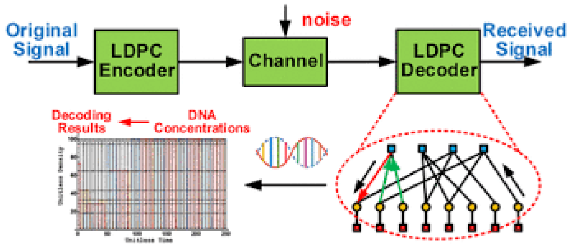


Fig 4.4 LDPC Functioning

A key that selects and employs the appropriate error correction method, which goes along well with the chosen encryption algorithm, is essential. For instance, we would eliminate even the possibility of such known vulnerabilities by employing an error correction method that does not rely on the same key as the encryption algorithm. Accordingly, with a strategic approach in employing Reed-Solomon and LDPC, our system can be resilient to errors while at the same time being efficient, resisting both accidental data loss and intentional interference, which could compromise the integrity of data in small network networks.

4.4 JUSTIFICATION

The use of encryption and error correction together in our methodology assures the securing and reliability of communication in small network networks. It consequently addresses both sources of risk in small network communication: interference from the environment, which may corrupt data during transmission, and interception by unauthorized parties. By encrypting data before applying error correction, our approach means that we obtain layering of security where encrypted data are transmitted with built-in resilience to transmission errors. This framework helps us attend to both confidentiality and integrity of the data.

First, encryption provides protection against access of the data, so even if it is intercepted, the information becomes unintelligible without an appropriate decryption key. Second, error correction provides another form of reliability in detection and correction of errors which might have occurred during transmission. The technique enjoys the merits of two methods: encryption keeps it safe from interception and error correction ensures authenticity of data. Another merit in the approach is the monitoring periodically in the system and updating security with the necessary due to possible vulnerabilities. This will address the shifting threats and helps maintain system integrity.

This methodology is justified because it balances security with efficiency. That is, the traditional small network-based networks either entrench a principle of security by concerning themselves with either security or error correction, and in that sense, they leave a gap in either data integrity or confidentiality. Combining the attributes ensures that it is a very robust system that can stand disruption at both the levels of randomness and intent. Continuous monitoring indeed supports this as it allows real time error detection and updates to encryption and error correction protocols.

Moreover, using this hybrid method ensures compliance with regulatory requirements for secure small network communication, and the data transferred is safe and sound. With this approach, the results obtained focus on both reliability and security objectives, thus generating a general protection against some of the common challenges in small network communication, such as signal interference, environmental noise, or even interception. It imparts long-term resiliency thereby maximizing its usability for sensitive applications where the integrity and confidentiality of data are considerations of utmost importance.

This two-layered security model is of great benefit in small network communication where data corruption results because of noise, interference, or signal degradation due to environmental factors. The encryption layer serves as the first line of defense, while the error correction layer ensures that errors introduced in the transfer could be detected and rectified. This offers higher resilience to attacks and accidental loss of data.

This approach monitors and updates periodically so that threats from the changing paradigm evolve. This system is relevant due to continuous assessment and adjustment of encryption protocols and error correction strategies because vulnerabilities are found due to the identification of emerging vulnerabilities, which ensures a high level of security and reliability. Moreover, the hybrid model allows regulatory compliance along with general protection of critical data in small network communications. It can be used for mission-critical applications, including military operations, global financial transactions and sensitive research data, etc.

Chapter 5
IMPLEMENTATION

5.1 INTRODUCTION

This chapter explores the actual implementation of our proposed data security and error correction framework achieved by combining Reed-Solomon (RS) error correction with various encryption methods-AES with RSA and AES-GCM. This chapter attempts to document the steps taken for algorithmic implementation in Python, capture the design choices, parameters measured, and the resultant benefits and limitations for each combination. This solution addresses the problems of reliability and security in the transmission of data, which particularly happens in noisy environments such as the channels used in small network communications.

We shall emphasize more on the error correction codes rather than the encryption algorithms. Our goal is to seek those combinations that maximize the error reduction and offer optimal security for the data being transferred. These include parameters such as BER, FER, overhead, and latency. These are critical factors determining if each approach is viable in real-world applications. Reed-Solomon has been used here mainly because it is suitable for error correction over noisy channels; AES is preferably utilized for data security in encryption or transmission of information. The measures of real-time performance analysis consist of BER, FER assessed probability of errors occurring in the whole frame, overhead as an indication of how the size of data increases due to the addition of redundancy for error correction, and latency points to delay due to encoding and decoding.

All the methods have been applied on various encoded and encrypted data sets. These tests have brought into light the performance of error correction as well as security for all the methods. From these parameters, we have selected the most appropriate combinations, which will be suitable for implementation in small network networks with minimum latency overhead and minimized values of BER and FER. Moreover, all the methods also include Python code and example runs for practical demonstration.

Discusses an implementation of a data security framework combining Reed-Solomon error correction with AES-GCM and AES with RSA encryption. Focusing on real-time performance metrics such as BER, FER, overhead, and latency, it evaluates the effectiveness of each combination for the secure, reliable transmission of data in noisy environments.

5.2 ALGORITHMS IMPLEMENTED

Error correction was deployed in the form of Reed-Solomon (RS) in our project, while multiple encryption algorithms applied to secure data: AES, RSA, and AES-GCM. We selected these algorithms because of their characteristics in aspects related to secure data communication. Our objective was to design a balanced system that might coincide with reliable data transmission while it protects it from unauthorized access. This part shall outline each algorithm, its functionality, and the role played by each of them in our implementation.

Reed-Solomon (RS) Error Correction:

Reed-Solomon is one of the most powerful error correcting schemes that are applied so frequently in communication systems and used to correct burst errors which are very common in small network transmission. The message is encoded with some extra redundant information provided in the form of parity symbols, so that multiple errors could be detected at the receiver side and corrected without requiring retransmission. Reed-Solomon is a block-based error-correcting code, processing fixed-size blocks of data. That makes it an excellent candidate for use when high deterministic performance as well as minimal latency in the recovery of data are required.

In this project, Reed-Solomon operates on encrypted data, thus guaranteeing that any errors that could arise during transmission will be corrected before decryption so that integrity and confidentiality of data are maintained. With the error-correcting capability of Reed-Solomon, it finds applicability for use in situations where degradation in data can be severe, such as small network communications. Since we add redundancy to the data prior to transmission, even if some parts of the information are corrupted while being transmitted, we will be able to retrieve the original information.

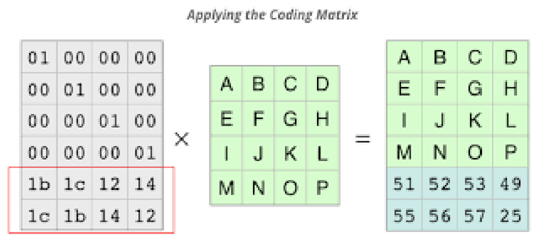


Fig 5.1 Reed-Solomon

Advanced Encryption Standard (AES):

Coming to the third one, which is AES, it is a symmetric encryption algorithm which gives fast and safe encrypting of the data. It works by converting plain text into a form known as the cipher-text, using an agreed-upon secret key so that only those with the right key can decrypt the data. AES is considered one of the most efficient algorithms. It also, because of its nature, is flexible and can apply different key sizes-128, 192, or 256 bits-to provide varying levels of security. AES is commonly employed in the project using CBC (Cipher Block Chaining) mode, which introduces randomness into the encryption process because each block is chained with the previous one, thus this makes it resist specific forms of cryptographic attacks.

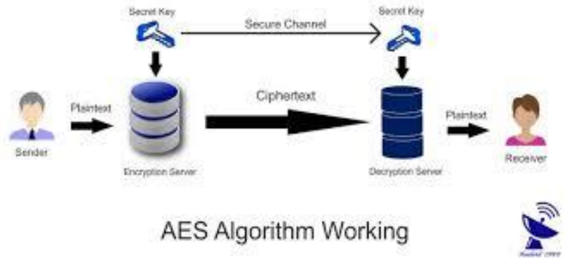


Fig 5.2 AES Working

In error correction before AES is performed on the data where content is encrypted, so that no unauthorized access to the same is made, as AES is computationally efficient; hence, it can be utilized in real-time data transmission without latency and actually can perform encryption in a very fast time, with no significant latencies due to its strength in securing large data sets with low computational overhead in different small network networks where fast and secure encryption needs to be employed.

<u>RSA Encryption:</u>

RSA is an asymmetric encryption algorithm in which the algorithm uses a pair of keys, public and private keys, to ensure secure data communication exchange. RSA is particularly valuable to our project for secure key management because it enables distribution of encryption keys in a safe manner, such as over a network but without sensitive information transfer over the network. The use of a public key to encrypt data thus assures that the keys themselves will not be an easier target for any would-be interceptors, since only the private key corresponding to the public key can decrypt the data.

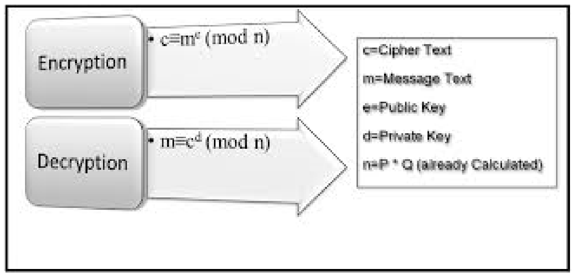


Fig 5.3 RSA Working

The RSA is used to distribute AES keys encrypted from the sender to receiver. The hybrid approach encryption, in which RSA is used for key exchange and AES for encrypting data, combines symmetric and asymmetric encryption's benefits, such as the efficiency of AES and secure key management of RSA. RSA is only used for secure key distribution, thereby increasing the protection offered to transmitted data without directly sending the sensitive keys across and preventing man-in-the-middle attacks.

AES-GCM Mode:

AES-GCM is an encryption mode, which combines AES encryption with integral data authentication. The mode would offer both confidentiality and integrity, meaning the data will be kept confidential and allow a receiver to verify whether or not data has been modified on route. GCM, Galois/Counter Mode, is the new one among all AES modes that makes use of counter-based encryption, combined with an authentication tag based upon the Galois Field, which gives encryption and authentication for one data transmission in one step, yet it is resistant to many attacks, hence more secure.

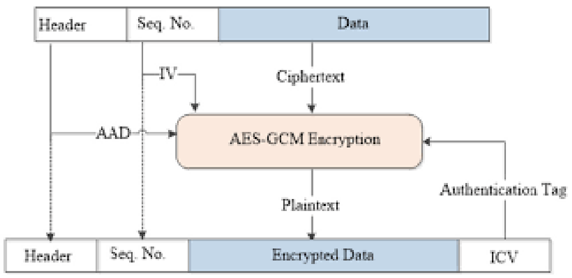


Fig 5.4 AES-GCM Working

In this project, AES-GCM is used for the encryption process with another layer of data authentication, such that any change in the data will be quite visible since tamper detection verifies the integrity of the data when it arrives. The verification tag enables the recipient to confirm that the information has not been altered in any manner; this is a necessity for small network communications, where integrity of data is as crucial as confidentiality of data. In terms of efficiency and the ability to secure the data against both tapping and interference, AES-GCM is the correct option to be used in high-security, real-time communication systems.

Implementation Strategy-Combined:

The use of Reed-Solomon for error correction and AES or AES-GCM for encryption forms a layered approach for secure communication. This is because by encrypting data with AES or AES-GCM before applying Reed-Solomon error correction, the encrypted form of data will be transmitted accurately and securely. Instead of directly transcribing sensitive information between sender and receiver, the usage of RSA for key management may add a further layer of security, which enables encrypted key exchange between sender and receiver.

This multi-layered implementation strategy addresses integrity as well as data security concerns. The Reed-Solomon code ensures correct reception of the data by detecting and correcting transmission errors, while AES and AES-GCM prevent unauthorized access and information tampering. Using the hybrid provision for RSA in key exchange and AES for data encryption for an application with significant error resilience, the provision of security along with efficiency with such hybrid provision is suited to communication with small networks whose data is prone to interference and interception. With this approach, we achieve a secure yet error-resistant framework for the communication for application at stake values.

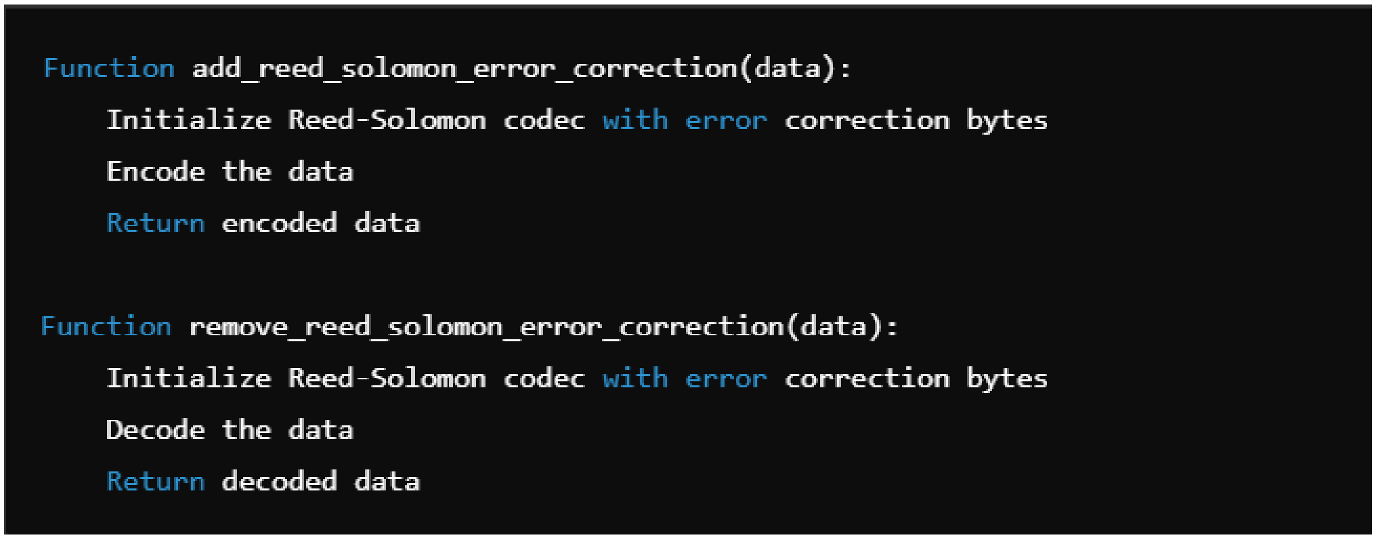
5.3 RS+AES+RSA IMPLEMENTATION

Let us take a look at the implementation performed by my team for this project. Implementation of algorithm using Reed-Solomon + Advanced Encryption Scheme and RSA.

Code Breakdown and Pseudocode

1. **Error Correction with Reed-Solomon**

- **Code:** add_reed_solomon_error_correction(data) and remove_reed_solomon_error_correction(data)
- **Pseudocode:**



```
Function add_reed_solomon_error_correction(data):
    Initialize Reed-Solomon codec with error correction bytes
    Encode the data
    Return encoded data

Function remove_reed_solomon_error_correction(data):
    Initialize Reed-Solomon codec with error correction bytes
    Decode the data
    Return decoded data
```

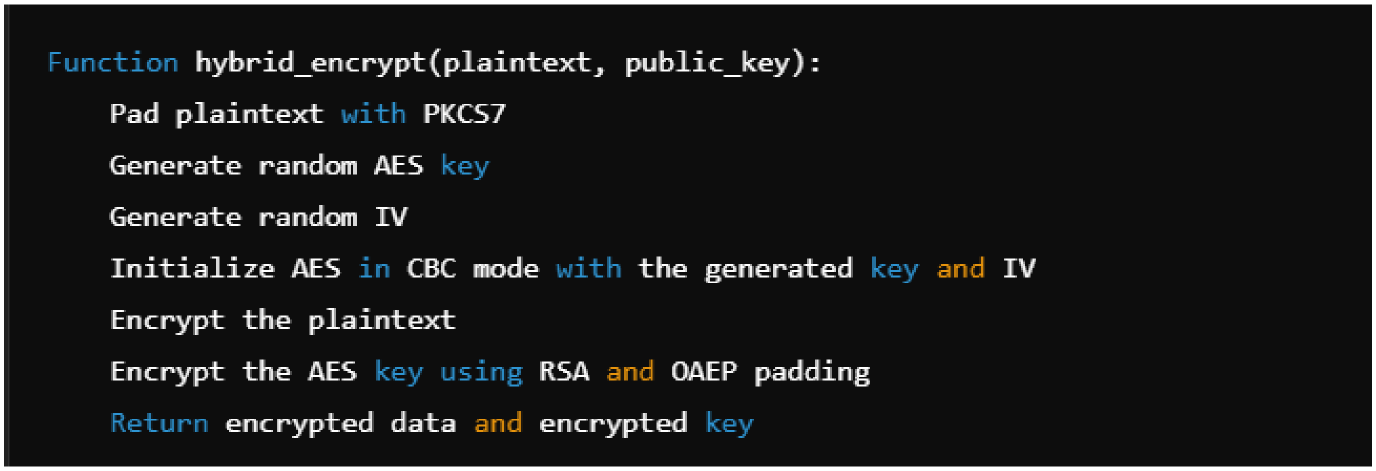
Fig 5.5 Pseudocode of Reed-Solomon Technique

- **Explanation:** These functions add and remove error correction to the data using Reed-Solomon codes, which allows the system to correct errors in transmission by adding redundancy. It increases resilience to errors by encoding the message with extra bytes, then decoding it to retrieve the original data.

2. **Hybrid Encryption Setup and AES Key Generation**

- **Code:** hybrid_encrypt(plaintext, public_key)

- **Pseudocode:**



```
Function hybrid_encrypt(plaintext, public_key):
    Pad plaintext with PKCS7
    Generate random AES key
    Generate random IV
    Initialize AES in CBC mode with the generated key and IV
    Encrypt the plaintext
    Encrypt the AES key using RSA and OAEP padding
    Return encrypted data and encrypted key
```

Fig 5.6 Pseudocode of Encryption Technique

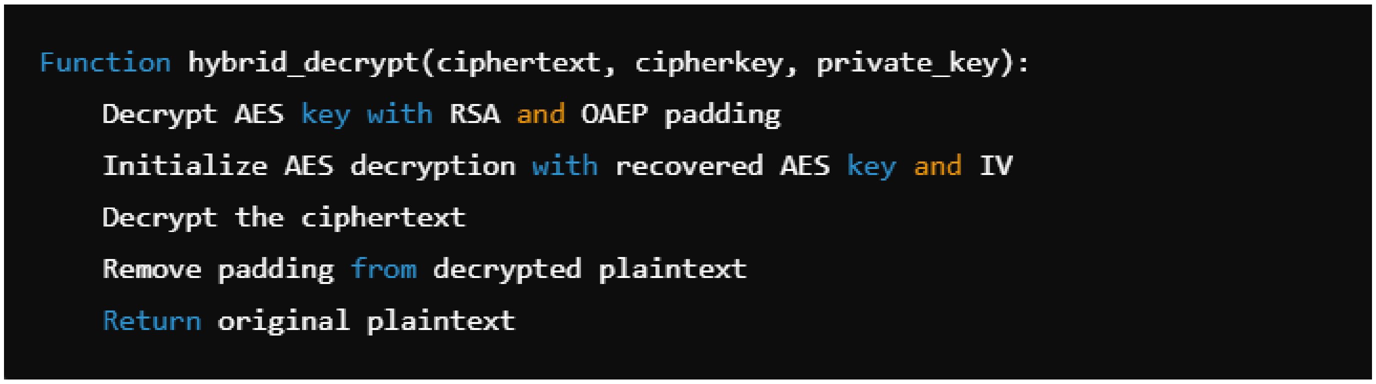
- **Explanation:** This function pads the plaintext and creates an AES encryption key and an initialization vector (IV) for CBC mode. It encrypts the data with AES, securing it with symmetric encryption, then encrypts the AES key with RSA using the recipient's public key, forming a secure hybrid encryption scheme.

  The encryption process begins with the padding of the plaintext using PKCS7 padding, ensuring the length of the data is a multiple of the AES block size. After padding, Reed-Solomon error correction encoding is applied to add redundancy and prepare the data for error correction during transmission. Next, a random 256-bit AES key is generated, and a 128-bit initialization vector (IV) is created for AES encryption in CBC mode. The padded and encoded plaintext is then encrypted using AES with the generated key and IV. To protect the AES key, it is encrypted using the RSA public key with OAEP padding, which ensures secure key exchange. The output of the encryption process is the encrypted ciphertext, the IV used in encryption, and the encrypted AES key.

3. **Hybrid Decryption process**

- **Code:** hybrid_decrypt(ciphertext, cipherkey, private_key)

- **Pseudocode:**



```
Function hybrid_decrypt(ciphertext, cipherkey, private_key):
    Decrypt AES key with RSA and OAEP padding
    Initialize AES decryption with recovered AES key and IV
    Decrypt the ciphertext
    Remove padding from decrypted plaintext
    Return original plaintext
```

Fig 5.7 Pseudocode of Decryption Technique

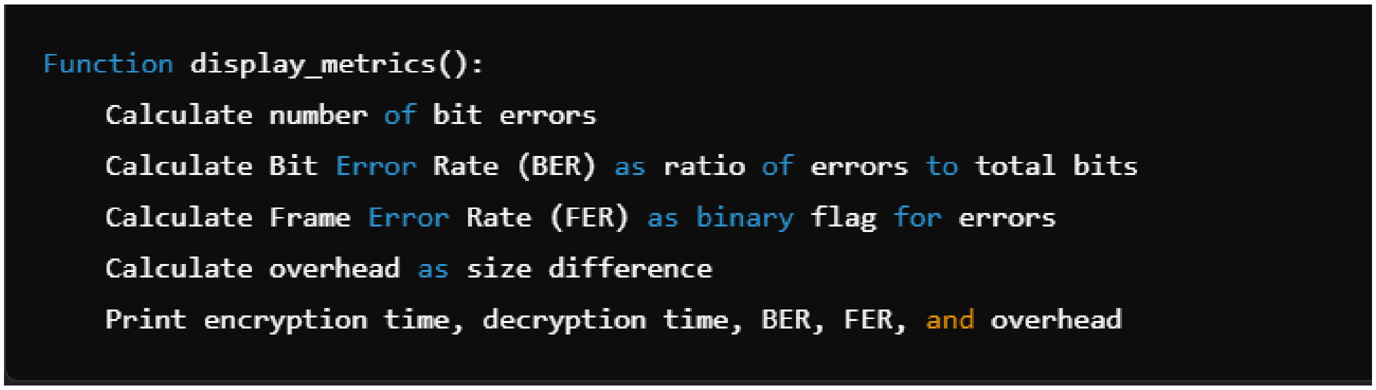
- **Explanation:** The decryption function first recovers the AES key using the RSA private key, enabling secure decryption of the AES-encrypted data. After decrypting, it removes the PKCS7 padding to retrieve the original plaintext. This allows data to be securely decrypted and made readable only by the intended recipient.

  For the decryption process, the AES key is first decrypted using the RSA private key with OAEP padding. With the recovered AES key and the IV, the ciphertext is decrypted using AES in CBC mode. The decrypted plaintext is then unpadded to remove the PKCS7 padding. Finally, the Reed-Solomon error correction decoding is applied to recover the original plaintext from the error-corrected data. The result is the recovered plaintext, identical to the original message before encryption. This process ensures both confidentiality and data integrity, while the error correction techniques help mitigate potential transmission errors.

4. **Performance Metrics Calculation**

- **Code:** display_metrics()

- **Pseudocode:**



```
Function display_metrics():
    Calculate number of bit errors
    Calculate Bit Error Rate (BER) as ratio of errors to total bits
    Calculate Frame Error Rate (FER) as binary flag for errors
    Calculate overhead as size difference
    Print encryption time, decryption time, BER, FER, and overhead
```

Fig 5.8 Pseudocode of Metrics calculation

- **Explanation:** This function measures performance by calculating Bit Error Rate (BER), Frame Error Rate (FER), and overhead. BER represents the fraction of bits that were incorrectly received, while FER shows if any error occurred in the frame. Overhead is computed as the increase in data size due to error correction, giving insights into the performance trade-offs in securing data.

Code Explanation

The code to be presented thus implements a secure data transmission system using a combination of Reed-Solomon error correction and hybrid AES-RSA encryption. The first step here in the system is encoding data with Reed-Solomon error correction, adding redundancy with the possibility of detecting and correcting errors that occurred due to transmission. This is important in high-error environments where Reed-Solomon codes actually correct multiples of errors per block without needing retransmission.

Encryption follows in this system using a hybrid approach. It pads the plaintext and encrypts with a randomly chosen AES key in CBC mode for fast secure data encryption. After that, it encrypts the AES key using RSA with the public key of the recipient. This RSA encryption of the AES key ensures that only the recipient decrypts the AES key, hence a secure yet efficient hybrid system that leverages the speed of symmetric encryption and the key exchange security of asymmetric encryption.

The receiver then decrypts the AES encryption key received using his private RSA. Then this AES encryption key can be used to decrypt the original ciphertext received. After decryption, the receiver will remove PKCS7 padding to arrive at plaintext. Thereafter, Reed-Solomon decoding is performed on the plaintext to correct errors introduced during transmission.

All such performance metrics are calculated in order to determine the reliability and efficiency of the system. The BER expresses how accurately the data bit is received. The FER represents the number of frames received erroneously, while overhead points towards the extra size of data arising due to the process of error correction.

5.4 RS+AES-GCM IMPLEMENTATION

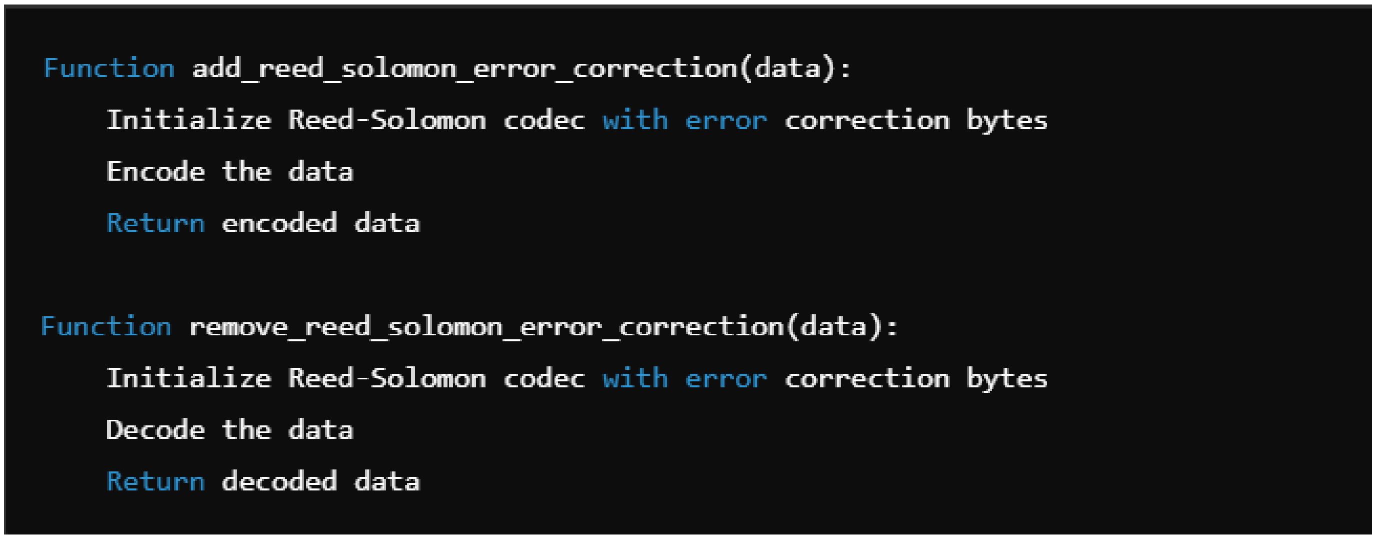
Let us look at the implementation performed by my team for this project.

Implementation of algorithm using Reed-Solomon + Advanced Encryption Scheme and GCM

Code Breakdown and Pseudocode

1. **Error Correction with Reed-Solomon**

- **Code:** add_reed_solomon_error_correction(data) and remove_reed_solomon_error_correction(data)
- **Pseudocode:**



```
Function add_reed_solomon_error_correction(data):
    Initialize Reed-Solomon codec with error correction bytes
    Encode the data
    Return encoded data

Function remove_reed_solomon_error_correction(data):
    Initialize Reed-Solomon codec with error correction bytes
    Decode the data
    Return decoded data
```

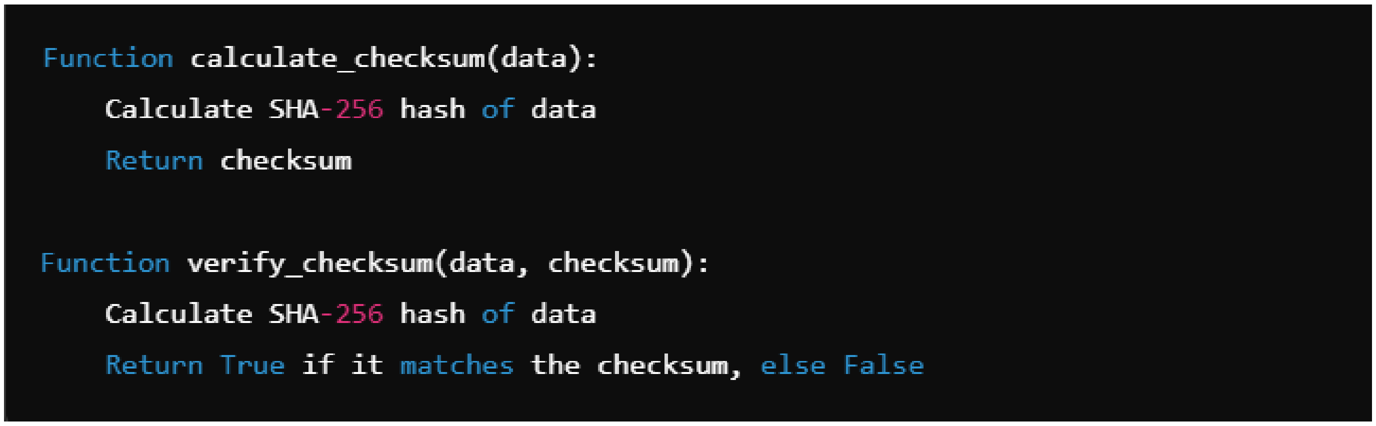
Fig 5.9 Pseudocode of Reed-Solomon Technique

- **Explanation:** These functions handle error correction by adding redundancy with Reed-Solomon encoding, which enables recovery of data even with some errors during transmission. The add_reed_solomon_error_correction function encodes data with additional bytes for error correction, and remove_reed_solomon_error_correction removes these error-correction bytes after successful transmission.

2. **Checksum calculation and verification**

- **Code:** calculate_checksum(data) and verify_checksum(data, checksum)

- **Pseudocode:**



```
Function calculate_checksum(data):
    Calculate SHA-256 hash of data
    Return checksum

Function verify_checksum(data, checksum):
    Calculate SHA-256 hash of data
    Return True if it matches the checksum, else False
```

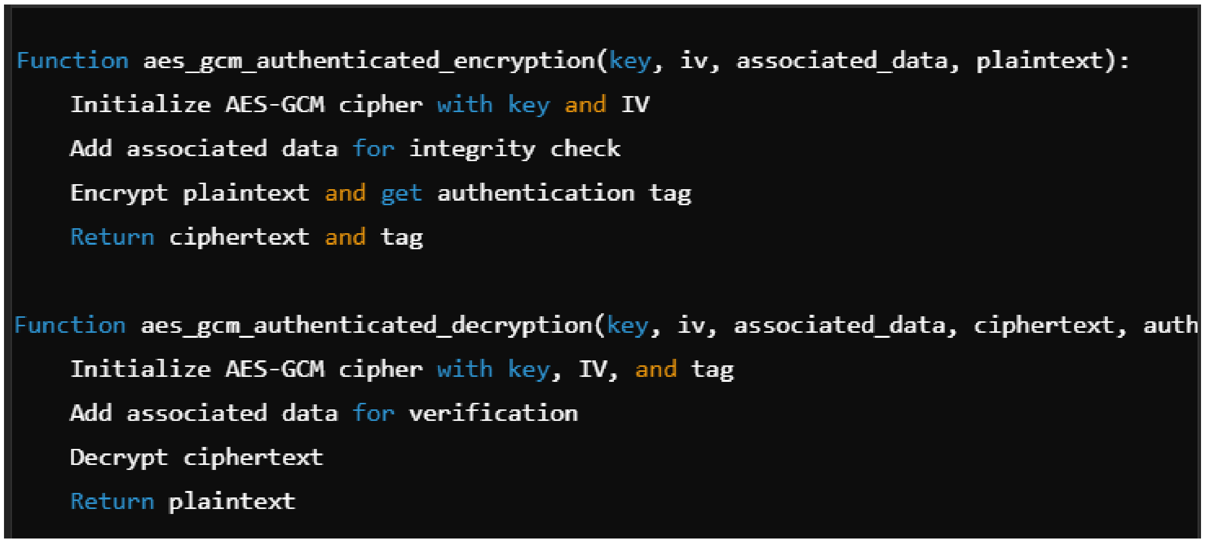
Fig 5.10 Pseudocode of Checksum calculation and verification

- **Explanation:** The calculate_checksum function generates a SHA-256 hash to ensure data integrity, while verify_checksum compares the recalculated hash with the original checksum to detect any tampering or transmission errors. The checksum calculation is an essential part of ensuring data integrity during encryption and decryption. In this implementation, a cryptographic hash function, specifically SHA-256, is used to compute the checksum. The checksum serves as a digital fingerprint of the data, allowing for verification of the data's integrity after it has been transmitted and decrypted.
  During encryption, the checksum of the Reed-Solomon encoded plaintext is calculated before encryption begins. This checksum is securely stored, ensuring that it can be compared later during the decryption process. Once the encrypted data is decrypted, the checksum of the recovered plaintext is computed again using the same SHA-256 hash function. If the calculated checksum matches the stored checksum, it confirms that the decrypted data is identical to the original encoded plaintext, assuring that no corruption or tampering has occurred during transmission.
  The use of checksums in this process enhances the security and reliability of the system, adding an additional layer of protection alongside encryption and error correction.

3. AES-GCM Authenticated Encryption and Decryption

- **Code:** aes_gcm_authenticated_encryption(key, iv, associated_data, plaintext) and aes_gcm_authenticated_decryption(key, iv, associated_data, ciphertext, auth_tag)

- **Pseudocode:**



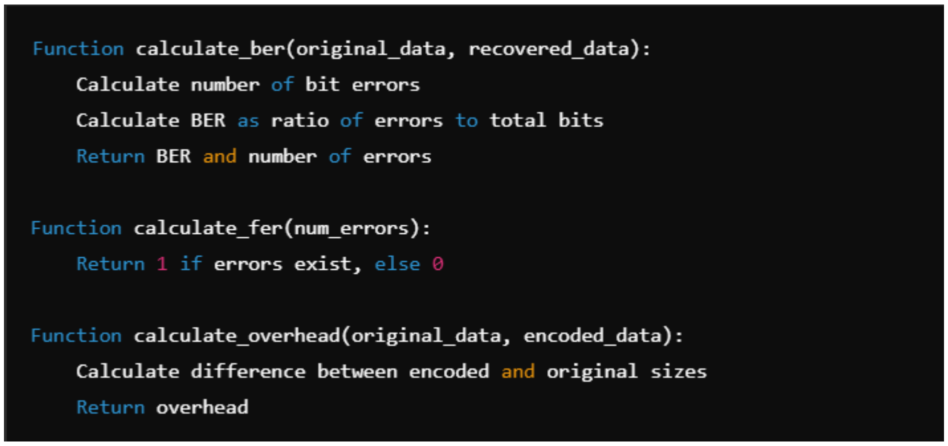
```
Function aes_gcm_authenticated_encryption(key, iv, associated_data, plaintext):
    Initialize AES-GCM cipher with key and IV
    Add associated data for integrity check
    Encrypt plaintext and get authentication tag
    Return ciphertext and tag

Function aes_gcm_authenticated_decryption(key, iv, associated_data, ciphertext, auth
    Initialize AES-GCM cipher with key, IV, and tag
    Add associated data for verification
    Decrypt ciphertext
    Return plaintext
```

Fig 5.11 Pseudocode of AES-GCM

- **Explanation:** These functions perform AES encryption in GCM mode, providing both data confidentiality and integrity. The aes_gcm_authenticated_encryption function uses a key, IV, and associated data to encrypt the plaintext, while aes_gcm_authenticated_decryption decrypts it and verifies authenticity with the authentication tag.

4. **Error Rate and Overhead Calculations**

- **Code:** calculate_ber, calculate_fer, calculate_overhead
- **Pseudocode:**



```
Function calculate_ber(original_data, recovered_data):
    Calculate number of bit errors
    Calculate BER as ratio of errors to total bits
    Return BER and number of errors

Function calculate_fer(num_errors):
    Return 1 if errors exist, else 0

Function calculate_overhead(original_data, encoded_data):
    Calculate difference between encoded and original sizes
    Return overhead
```

Fig 5.12 Pseudocode of Error Rate and Overhead

- **Explanation:** These functions evaluate transmission efficiency by calculating Bit Error Rate (BER), Frame Error Rate (FER), and data overhead. BER measures data transmission accuracy, FER indicates frame-level errors, and overhead shows additional bytes added by error correction.

Code Explanation

This code is designed to send the data securely with error correction and integrity checks. The first operation utilizes Reed-Solomon encoding to add some redundancy to the data in order to allow error recovery. It then applies the checksum on the encoded data, which will produce an independent hash of the data that can be verified when it arrives that the data was not tampered during the transmission.

The encoded data is encrypted in Galois/Counter Mode (GCM) using AES. It provides both encryption and authentication. The encryption parameters will be set using a symmetric 256-bit AES key coupled with a 96-bit initialization vector, IV. GCM associates data, and that adds context to the encryption and strengthens integrity checks. The authentication tag, auth_tag, accompanies Ciphertext and created such that the alterations in the encrypted message are detected.

After receiving, the code decrypts the ciphertext using the AES key and the same IV, verifies the auth_tag against any tampering of the data. The retrieved data is checked using a checksum for integrity and, finally, the Reed-Solomon decoding eradicate the redundant bytes and restore the original data.

Finally, the code calculates performance metrics for Bit Error Rate (BER), Frame Error Rate (FER), and overhead in order to demonstrate that these metrics indeed indicate the quality of data transmission based on how effective this error correction and encryption is in transmitting data securely and accurately.

Chapter 6
OBSERVATION

6.1 INTRODUCTION

Modern data communication and cryptography systems require giving guarantees over the integrity, confidentiality, and reliability of data being transmitted. For this purpose, a combination of multiple techniques is used together for error detection, security, and efficient performance without major latency and overhead. The current chapter gives an in-depth analysis of system performance where error correction is implemented using Reed-Solomon, encryption is done using AES-GCM, and checksum verification is performed. These techniques, when integrated together give an all-around approach towards the guaranteeing of robustness and security of data transmission. However, these advance techniques come with some trade-off, primarily concerning system performance. To assess the performance of such techniques, a set of key performance metrics were considered from which insight into overall performance of both encryption as well as error-correction processes were gained.

The objective of this chapter is to analyze and evaluate the performance characteristics of the implemented hybrid system. Here we are focused on knowing the impact of Reed-Solomon error correction, AES-GCM encryption, and checksum mechanisms on critical parameters such as BER, FER, encryption time, decryption time, and overhead. This analysis will help us to better understand the trade-offs between security and performance and how these technologies together may be optimized for such scenarios like secure data communication over unreliable channels.

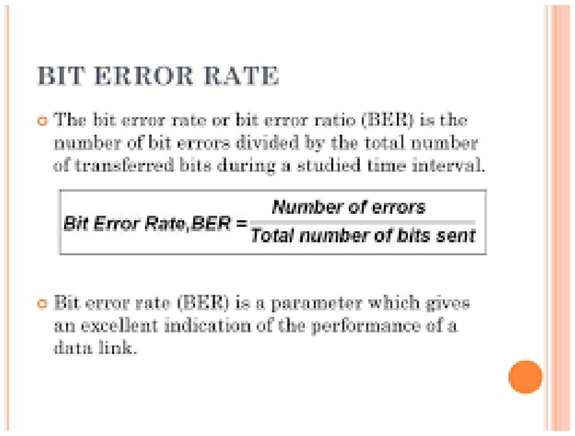
This chapter will also provide a detailed study of how the performance of each one of these component elements, that is Reed-Solomon encoding, AES-GCM encryption, and checksum verification, contributes to overall system effectiveness. Additionally, the relationships of these components shall also be discussed to demonstrate how the collective impacts of these component elements on these systems affect their robustness and efficiency.

6.2 METRICS USED

The main metrics for the performance evaluation of the proposed system are as follows. These include metrics that provide insight into the system's functionality and performance. The following metrics were followed in this study:

a) Bit Error Rate (BER):
Bit Error Rate is defined as the number of bits received in error compared to the total number of bits transmitted. It essentially represents a prime indicator of the good

performance of the error correction mechanisms, such as Reed-Solomon. The higher the BER, the worse a system is at recovering from errors during transmission. In this chapter, BER will be used as a measure for evaluating the efficiency of Reed-Solomon error correction in controlling errors when transmitted with AES-GCM encryption.



BIT ERROR RATE

- The bit error rate or bit error ratio (BER) is the number of bit errors divided by the total number of transferred bits during a studied time interval.

$$\text{Bit Error Rate,BER} = \frac{\text{Number of errors}}{\text{Total number of bits sent}}$$

- Bit error rate (BER) is a parameter which gives an excellent indication of the performance of a data link.

Fig 6.1 Bit-Error Rate

b) Frame Error Rate (FER):

Frame Error Rate is the other measure of significance, which represents the fraction of erroneous frames-or packets-included in them. Error correction capabilities of the system are augmented, and as such, Reed-Solomon's ability to identify and correct frame level errors is also included in its general error corrections. The study of FER will give an idea about how these hybrid techniques improve the general data transmission reliability over the communication link.

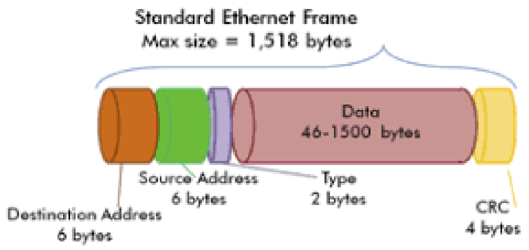


Fig 6.2 Frame-Error Rate

c) Encryption and Decryption Time:

These metrics measure the amount of time needed by encryption and decryption processes, respectively. Encryption time refers to how long it takes to encrypt the data using AES-GCM and any additional steps like error correction, while decryption time refers to the time needed to reverse the encryption process. The time efficiency of AES-GCM encryption and Reed-Solomon encoding are going to be evaluated.

d) Overhead:

Overhead: It refers to additional data, such as error correction, encryption, or integrity checks. Reed-Solomon introduces redundancy overhead as does AES-GCM encryption, which also introduces its own encryption overhead. This metric explains the increment in the size of the data because of using these mechanisms of error correction and encryption and explains the trade-off between security and efficiency in terms of data.

e) Latency:

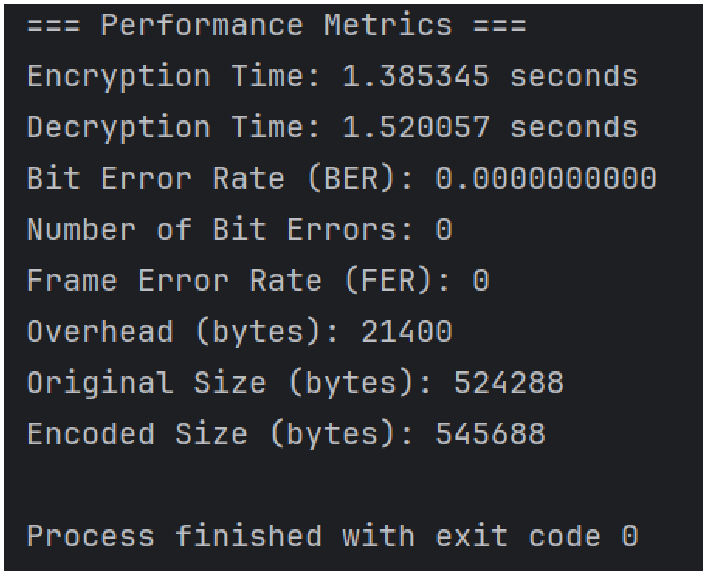
Latency is the time taken by the system to send the data and receive it back. They do not contribute directly toward latency: error correction techniques or checksum calculations, while the encryption process and error correction might introduce some delay. Latency is yet another wonderful variable to gauge the real-time performance of the system affected by the usage of these advanced techniques.

Fig 6.3 Latency

These metrics will be used for the assessment of the overall performance of the system with a basis for comparison of effectiveness for various combinations of encryption, error correction, and integrity verification techniques in a data communication environment. The results shall also be used to identify any potential areas for optimization or improvement in the hybrid system.

6.3 METRICS IMPLEMENTATION

Now we shall look at the observed output of our implementation for a data size of 512 kb.

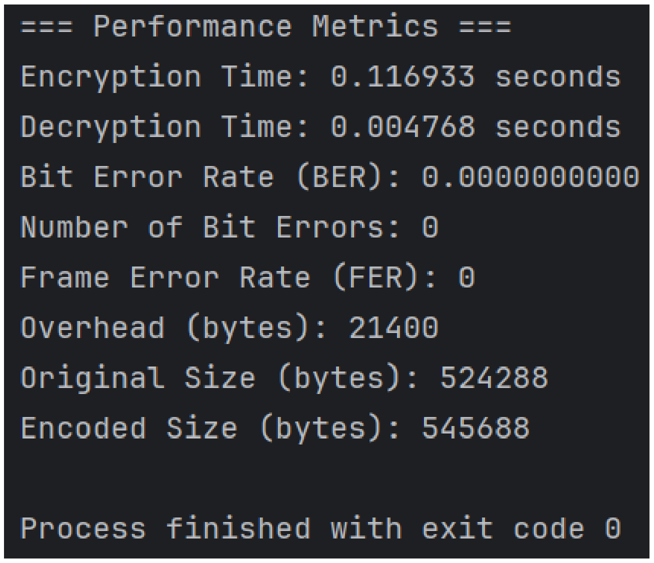


```
=== Performance Metrics ===
Encryption Time: 1.385345 seconds
Decryption Time: 1.520057 seconds
Bit Error Rate (BER): 0.0000000000
Number of Bit Errors: 0
Frame Error Rate (FER): 0
Overhead (bytes): 21400
Original Size (bytes): 524288
Encoded Size (bytes): 545688

Process finished with exit code 0
```

Fig 6.4. Output of RS,AES-GCM on 512kb data

This figure displays the results for a data transmission system utilizing **Reed-Solomon error correction** and **AES-GCM encryption** with 512KB of data. The output provides the following key performance metrics:

- **Encryption Time:** 1.385345 seconds, indicating the time taken to encrypt the data.
- **Decryption Time:** 1.520057 seconds, showing the time taken to decrypt the ciphertext.
- **Bit Error Rate (BER):** 0.0000000000, suggesting that no bit errors were detected during the transmission or decryption process.
- **Number of Bit Errors:** 0, confirming that no errors were encountered.
- **Frame Error Rate (FER):** 0, indicating no frame errors during transmission.
- **Overhead (bytes):** 21,400 bytes, the additional data added by Reed-Solomon error correction for error detection and correction.
- **Original Size (bytes):** 524,288 bytes (512KB), the size of the unmodified data.
- **Encoded Size (bytes):** 545,688 bytes, the size of the data after Reed-Solomon encoding.



```
=== Performance Metrics ===
Encryption Time: 0.116933 seconds
Decryption Time: 0.004768 seconds
Bit Error Rate (BER): 0.0000000000
Number of Bit Errors: 0
Frame Error Rate (FER): 0
Overhead (bytes): 21400
Original Size (bytes): 524288
Encoded Size (bytes): 545688

Process finished with exit code 0
```

Fig 6.5. Output for RS,AES&RSA-OAEP on 512kb data

This figure shows the performance metrics for a system using **Reed-Solomon error correction**, **AES encryption**, and **RSA-OAEP encryption** for the same 512KB of data. The output includes:

- **Encryption Time:** 0.116933 seconds, significantly lower than the AES-GCM scheme, reflecting faster encryption with RSA-OAEP.
- **Decryption Time:** 0.004768 seconds, also very fast due to the RSA-OAEP decryption process.
- **Bit Error Rate (BER):** 0.0000000000, indicating no bit errors during decryption.
- **Number of Bit Errors:** 0, confirming that the data was transmitted without any errors.
- **Frame Error Rate (FER):** 0, showing no frame errors.
- **Overhead (bytes):** 21,400 bytes, like the previous scheme, reflecting the Reed-Solomon error correction's impact.
- **Original Size (bytes):** 524,288 bytes (512KB), unchanged.
- **Encoded Size (bytes):** 545,688 bytes, also identical to the previous output, showing the same level of overhead due to Reed-Solomon.

Attribute	RS + AES + RSA	RS + AES-GCM
Encryption Algorithm	AES (symmetric) + RSA (asymmetric)	AES-GCM (symmetric with authenticated encryption)
Error Correction	Reed-Solomon (RS)	Reed-Solomon (RS)
Encryption Speed	Slower, due to RSA's computational intensity	Faster, as AES-GCM is optimized for speed
Decryption Speed	Slower, RSA requires more processing for key operations	Faster, AES-GCM provides efficient authenticated decryption
Security Level	High (asymmetric RSA for secure key exchange)	High (AES-GCM provides data confidentiality and integrity)
Authentication	Not inherently provided; requires additional MAC or HMAC	Built-in authentication with GCM (Authenticated Encryption)
Error Detection	Reed-Solomon error detection	Reed-Solomon error detection
Error Correction Capability	Strong, Reed-Solomon corrects errors in noisy environments	Strong, same Reed-Solomon-based error correction
Overhead	Higher overhead due to RSA key size and additional security metadata	Lower overhead since AES-GCM is more compact with combined encryption/authentication
Latency	Higher due to RSA’s computational demand	Lower, AES-GCM reduces latency through faster processing
Suitable for Real-Time Use	Limited due to higher latency and overhead	Good, as AES-GCM offers low latency and high speed
Best-Suited Applications	Applications requiring secure key exchange and non-repudiation, e.g., secure financial transactions, medical record transfers, satellite communication	Applications needing low-latency, real-time security, e.g., video streaming, instant messaging, large file transfers
Complexity	Higher complexity due to dual encryption algorithms (AES + RSA)	Lower complexity, single symmetric encryption with authenticated encryption
Resource Requirements	High (RSA needs significant processing power and memory)	Moderate, AES-GCM is lightweight and efficient on resources
Quantum Resistance	Moderate (RSA is vulnerable, AES is resistant with longer keys)	Good (AES-GCM is generally more resistant than RSA, particularly for symmetric encryption needs)

Table 3. Algorithms Comparison

Let us understand the above table:

1. The following table compares two security communication frameworks, RS + AES + RSA and RS + AES-GCM, in terms of how they utilize Reed-Solomon (RS) error correction in conjunction with two different encryption techniques.

2. The combination RS + AES + RSA is slower than other options both in encryption and decryption since RSA is an asymmetric algorithm and, hence, quite computationally expensive, thereby contributing to more latency. Even though the result RS + AES-GCM is faster because AES-GCM is optimized for speed and AES-GCM encrypts with authenticated encryption in one go.

3. Security Level: Both the combinations ensure security at a high level but through a different mean. RS + AES + RSA relies on RSA for securely exchanging keys and providing secure protection against interception. It is quite strong in asymmetric encryption. RS + AES-GCM relies on the authenticated mode of AES-GCM providing assurance about data confidentiality and integrity without any need for a separate key exchange protocol, hence very effective and safe for many applications.

4. With AES-GCM, authentication is built into data that can be encrypted and verified with the same step, hence giving it integrity and authenticity. RS + AES + RSA does not have built-in authentication. Additional authentication mechanisms like MAC or HMAC would still be necessary for data integrity.

5. Error Detection and Correction Capability: Both the techniques use Reed-Solomon (RS) for error correction, which is appropriate for noisy channels, where error detection and correction are a necessity. RS resolutely identifies and corrects errors in the data so that both the combinations are highly reliable in terms of maintaining data integrity.

6. Overhead: The overhead is more in RS + AES + RSA because keys related to RSA are much larger in size and the metadata involved with the exchange of keys, which makes the overhead of the message higher. In case of RS + AES-GCM, the overhead is less because AES-GCM produces an encoded bit for both encryption as well as authentication in compact form, making it relatively more efficient regarding data size.

7. Latency: RS + AES + RSA entails greater latency as RSA encryption is slow and, hence, not a favourite in low-latency applications. However, RS + AES-GCM is characterized by much lesser latency, making it appropriate for real-time applications in which AES-GCM offers fast encryption and decryption.

8. RS + AES-GCM would be much more suited to real-time applications, as it does not introduce much latency and is very efficient; thus, it would be well-deployed in video streaming or instant messaging, where speed is of the essence. RS + AES + RSA will be better suited to applications where real-time processing is less critical but key secure exchange and secrecy of data are important.

9. Appropriate Scenarios. RS + AES + RSA is the suitable choice for those applications where the confidentiality and integrity of data are very important, such as financial transactions, medical records, and satellite communications. RS + AES-GCM will be considered for high-speed, low-latency applications video broadcasting, large-sized file transfers, and real-time messaging of voice and text messages in which fast processing is much more important than secure key exchange.

10. Complexity: RS + AES + RSA is more complex because it has symmetric and asymmetric encryption, creating layers in the encryption process. RS + AES-GCM is less complex. This one symmetric encryption algorithm is easier for a person to apply and manage than trying to apply three single encryption algorithms.

11. Resource Requirements: RS + AES + RSA has a larger processing and memory usage; therefore, the later has resource-intensive nature due to high computational requirements for RSA. RS + AES-GCM is relatively lightweight and much better suited to be used on devices with minimal resources.

12. Quantum Resistance While asymmetric encryption (RSA) is much more sensitive to quantum computing attacks, symmetric encryption such as in AES-GCM is generally more resilient, particularly when the key lengths are longer. As a result, RS + AES-GCM is slightly better quantum-resistant compared to RS + AES + RSA, though both combinations will likely require at least minor adjustments in the future to stay secure in a post-quantum world.

13. RS + AES + RSA is robust, in case of secure applications needing both secure key exchange along with high data confidentiality. RS + AES-GCM is optimal when highly high-speed, in real-time application requires encrypting and correcting errors simultaneously. Each of these combinations satisfies uniquely different sets of applications and balances the factors of security, efficiency, and resources.

Chapter 7
RESULTS AND CONCLUSIONS

7.1 RESULT

Indeed, the integration of error correction algorithms such as Reed-Solomon (RS) and encryption algorithms like AES-GCM and RSA-OAEP increases the reliability and security associated with the transfer of data. All of them provide a different feature in terms of error detection, error correction, and efficiency in encryption.

- Reed-Solomon + AES-GCM: Suitable for applications where simple error detection is desired along with high strength error correction with minimal overhead. The AES-GCM encryption mode would ensure both confidentiality as well as integrity, hence it is a viable solution for most applications implemented in real-time. Examples include video streaming, message system and file transfer up to massive size, as error correction due to Reed-Solomon is assured regarding the detection as well as correction of data transmission error conditions, thus forming an all-around reliable mechanism for error-free communication. This system is very efficient and does quite well even at high-speed data transfer conditions.

- Reed-Solomon + AES + RSA-OAEP: This uses in addition an added layer of security, specifically the RSA-OAEP encryption used for key exchange with secure and detailed error analysis. Contrary to the previous combination, this setup, however, introduces slightly higher complexity and latency. This makes it suitable for environments that require precise error correction and safe key management. It is thus quite useful for those systems requiring transfers of extremely sensitive data. This includes small network communications, financial transactions, medical records, and broadcasting systems. In these scenarios, extra computational overhead for RSA encryption is worthwhile as the data being exchanged demand significant confidentiality and authenticity.

- Reed-Solomon + AES + RSA + Turbo codes is also very reliable for high-priority data, where accuracy in error correction is crucial. This scheme is used in niche fields like wireless communication systems, small network communication, secure file transfer protocol, and DRM. These systems now enjoy Turbo code's features of error correction with AES and RSA's added security mechanism.

While these combinations offer robust security and reliability, they are not without limitations. The added overhead due to error correction and encryption leads to increased latency. In cases where **speed and real-time processing** are crucial, such as in **video streaming**, the more straightforward **Reed-Solomon + AES-GCM** combination is preferred for its faster processing and minimal latency. However, in systems where data **integrity** and **security** are of utmost importance, such as small network communication or critical infrastructure, the second combination of **Reed-Solomon + AES + RSA-OAEP** is the better choice despite its added complexity and computational demands.

7.2 CONCLUSION

Hybrid combination with error correction and encryption provides a means of overall and secure protection of data communications. Using Reed-Solomon for error correction, and AES-GCM or RSA-OAEP for encryption, it establishes a strong framework that is going to reveal and correct errors while providing data secured, integral protection from unauthorized access attempts. This powerful combination provides double-layer protection against error-prone transmission attempts as well as unauthorized access attempts.

Among the alternatives suggested is Reed-Solomon + AES-GCM, which does better with systems that must have maximum speed and minimal latency, such as in real-time communications and high-speed data transfers. For applications with extremely secret data or requiring secure key exchange and detailed analysis of errors, a stronger alternative at a higher computational price is Reed-Solomon + AES + RSA-OAEP.

The Reed-Solomon + AES + RSA + Turbo codes combination offers the best data security and integrity levels only for mission-critical communications like small network communications or financial transactions but this comes at the cost of higher overhead and latency.

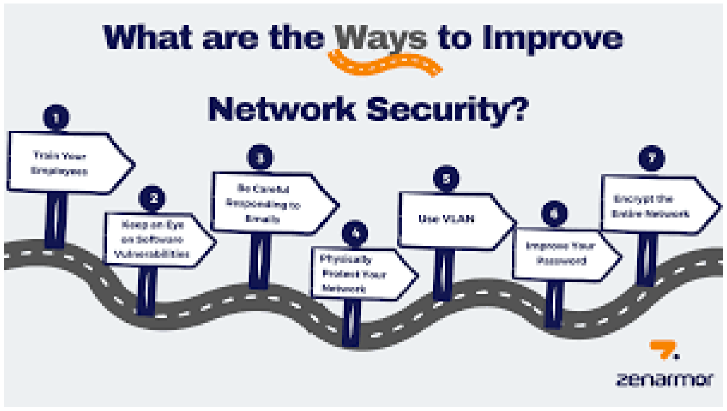

What are the Ways to Improve Network Security?
1
Train Your Employees
2
Keep an Eye on Software Vulnerabilities
3
Be Careful Responding to Emails
4
Physically Protect Your Network
5
Use VLAN
6
Improve Your Password
7
Encrypt the Entire Network
zenarmor

Fig 7.1 Better Network Security

In conclusion, the above integration of error correction and encryption techniques enhances the trustworthiness and reliability of data networks with the guarantee that even if errors occur or there are suspected security threats, the data transmitted will not only remain accurate but also secure.

7.3 FUTURE WORK AND SCOPE

As the requirement for even more secure and efficient communication continues to increase, the incorporation of techniques that include error correction along with encryption may have good potential to be followed in the future. These might be some potential directions for future work:

1. Optimization of Performance: Although the currently available combination provides excellent security and error correction, latency and overhead introduced due to these techniques can be optimized. Future studies may include lowering the computational complexity, especially in RSA-OAEP and Turbo codes, so these techniques are more appropriate in real-time applications without compromising their efficiency.

2. Applications in Emerging Technologies: With the advent of 5G and IoT, there is an increased demand for low latency with the ability to have high-throughput communication systems that are not just secure but also reliable. There is a need for further study into Reed-Solomon with advanced techniques such as Post-Quantum Cryptography to be applied to address emerging technology that necessitates the safety and error correction among its applications.

3. Hybrid Systems with Machine Learning: This includes the application of machine learning techniques toward dynamically choosing and adapting error correction and encryption schemes based on conditions in the channel and characteristics of the data. Such systems would then be real-time adaptive, intelligent, and performing at the best possible level.

4. Low-Resource Systems. For mobile device and embedded systems, lightweight encryption and error correction techniques would be key concerns for resource-poor environments. Balancing security capability with error-correcting ability and performance has a huge potential breakthrough area for these types of systems.

Addressing these areas, further work will thus help push the envelope for more scalable and adaptive error correction and encryption techniques while being able to provide greater security in a constantly evolving landscape of data communication.

Chapter 8
REFERENCES

8.1 References

[1] A. M. Abukari, An enhanced error detection and correction scheme for enterprise resource planning (ERP) data storage, Journal of Advances in Computer Science and Maths 36 (2021). doi:10.9734/jamcs/2021/v36i930405.
URL https://journaljamcs.com/index.php/JAMCS/article/view/1602

[2] A. Couvreur, M. Lequesne, On the security of subspace subcodes of Reed–Solomon codes for public key encryption, IEEE Transactions on Information Theory 68 (1) (2021) 632–648.

[3] S. Mahmood, S. M. Mohsin, S. M. A. Akber, Network security issues of data link layer: An overview, in: 2020 3rd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), 2020, pp. 1–6. doi:10.1109/iCoMET48670.2020.9073825.

[4] N. C, Survey on network security with cryptography, https://www.ijser.org/researchpaper/Survey-on-Network-Securitywith-Cryptography.pdf (2019).

[5] I. B. Djordjevic, Physical-layer security and quantum key distribution, Springer, 2019.

[6] A. M. Abdelaziz, E. Abdelwanees, A. D. Elbayoumy, Securing the space data link communication protocol of earth observation small networks (2019). doi:10.1109/ICICIS46948.2019.9014846.

[7] G. Yang, L. Dai, Z. Wei, Challenges, threats, security issues and new trends of underwater wireless sensor networks, Sensors 18 (11) (2018) 3907.

[8] D. Purwanto, Optimization of data security system control with CRC (cyclic redundancy check) algorithm, Budapest International Research and Critics Institute-Journal (BIRCI-Journal) 4 (3) 4635–4642.

[9] Z. Chen, L. Yin, Y. Pei, J. Lu, Codehop: physical layer error correction and encryption with LDPC-based code hopping, Science China Information Sciences 59 (2016) 1–15.

[10] E. Bertino, Data security and privacy concepts (2016). doi:10.1109/COMPSAC.2016.89.
URL https://ieeexplore.ieee.org/document/7552042

[11] M. Dener, O. F. Bay, Teenysec: a new data link layer security protocol for WSNs, Security and Communication Networks 9 (18) (2016) 5882–5891.

[12] N. Islam, Z. Shahid, W. Puech, Denoising and error correction in noisy AES-encrypted images using statistical measures (2016). doi:https://doi.org/10.1016/j.image.2015.11.003. URL https://www.sciencedirect.com/science/article/pii/S0923596515002003

[13] W. H. Jeong, B.-G. Yeo, K.-H. Kim, S.-H. Park, S.-W. Yang, J.-S. Lim, K.-S. Kim, Performance analysis of the encryption algorithms in a small network communication network based on h-ARQ, The Journal of The Institute of Internet, Broadcasting and Communication 15 (1) (2015) 45–52.

[14] L. Ning, L. Kanfeng, L. Wenliang, D. Zhongliang, A joint encryption and error correction method used in small network communications (2014). doi:10.1109/CC.2014.6825260.

[15] J. M. Hamamreh, M. Yusuf, T. Baykas, H. Arslan, Cross MAC/PHY layer security design using ARQ with MRC and adaptive modulation, in: 2016 IEEE Wireless Communications and Networking Conference, IEEE, 2016, pp. 1–7.

[16] O. S. Younes, Securing ARP and DHCP for mitigating link layer attacks, Sādhanā 42 (2017) 2041–2053.

[17] T. Rao, Joint encryption and error correction schemes, ACM SIGARCH Computer Architecture News 12 (3) (1984) 240–241.

[18] W. Stallings, Data and Computer Communications (2007).

[19] B. A. Forouzan, Data Communications and Networking (McGraw-Hill) (2007).

[20] K. W. R. James F. Kurose, Computer Networking, A Top-Down Approach (2000).

[21] A. Tahir, S. Schwarz, M. Rupp, BER comparison between convolutional, turbo, LDPC, and polar codes, in: 2017 24th International Conference on Telecommunications (ICT), 2017, pp. 1–7. doi:10.1109/ICT.2017.7998249.

[22] O. Aitsab, R. Pyndiah, Performance of Reed-Solomon block turbo code, in: Proceedings of GLOBECOM'96, 1996 IEEE Global Telecommunications Conference, Vol. 1, 1996, pp. 121–125 vol.1.