INFORMATION SECURITY FOR SHARED IoT INFRASTRUCTURES

A major project report submitted in partial fulfillment of the requirement for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering / Information Technology

Submitted by

Parth Kr Khare (201517)

Kartik Joshi (201524)

Under the guidance & supervision of

Mr. Arvind Kumar, Assistant Professor (Grade-II)

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology,

Waknaghat, Solan - 173234 (India)

CERTIFICATE

This is to certify that the work which is being presented in the project report titled **'Information Security for Shared IoT Infrastructures'** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by **Parth Kr Khare (201517) and Kartik Joshi (201524)** during the period from August 2023 to May 2024 under the supervision of **Mr. Arvind Kumar** (Assistant Professor Grade - II, Department of Computer Science & Engineering and Information Technology).

(Student Signature with Date)
Student Name: Parth Kr Khare
Roll No.: 201517

(Student Signature with Date)
Student Name: Kartik Joshi
Roll No.: 201524

The above statement made is correct to the best of our knowledge.

(Supervisor Signature with Date)
Supervisor Name: Mr. Arvind Kumar
Designation: Assistant Professor (Grade-II)
Department: CSE & IT
Dated:

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this report entitled **'Information Security for Shared IoT Infrastructures'** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from January 2024 to June 2024 under the supervision of **Mr. Arvind Kumar** (Assistant Professor Grade - II, Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature with Date)
Student Name: Parth Kr Khare
Roll No.: 201517

(Student Signature with Date)
Student Name: Kartik Joshi
Roll No.: 201524

This is to certify that the above statement made by the candidate is true to the best of our knowledge.

(Supervisor Signature with Date)
Supervisor Name: Mr. Arvind Kumar
Designation: Assistant Professor (Grade-II)
Department: CSE & IT
Dated:

ACKNOWLEDGEMENT

With immense gratitude, we extend our heartfelt thanks to our esteemed Supervisor, Mr. Arvind Kumar, Assistant Professor (Grade-II) in the Department of CSE at Jaypee University of Information Technology, Wakhnaghat. We are deeply indebted for his tireless support, patient mentorship, constructive criticism, and unwavering encouragement. His keen interest in our work has truly been a driving force behind the project's completion.

We would also like to express our gratitude to Mr. Arvind Kumar from the Department of CSE for his valuable assistance, which significantly contributed to the successful conclusion of our project.

Our sincere thanks extend to all individuals, whether directly or indirectly, who played a role in the triumph of this project. We acknowledge the support of the entire staff, both teaching and non-teaching, whose timely assistance and facilitation greatly aided our endeavor.

In closing, we wish to recognize and appreciate the enduring support and patience of our parents. Their unwavering encouragement has been a source of strength throughout this journey.

With gratitude,

Parth Kr Khare
(201517)

Kartik Joshi
(201524)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The proposed project deals with securing shared IoT systems. Shared internet of things involves collaboration by various parties through a single network of interrelated devices. It is perfect for efficiency and even generates grave questions on safety matters but we must tackle this and secure information data safely.

This project starts with defining shared IoT and noting how different categories of participants work together to enjoy services such as smart sensors and industrial machines.

Security in IoT is vital as it has become an everyday necessity, from health care services, smart cities and through industries among others. Such a scenario may entail unlawful entry into the system, leaking of sensitive data or even a malicious attack that can cause financial losses and destruction of equipment. Therefore, the purpose of this project is to design a robust security framework that applies and works well with shared IoT systems.

Such a type of security plan involves secure channels for communication between devices, encrypting of data, control over access granting as well as unusual behavior detection systems. The project proposes taking some preventive measures to create a strong protection against cybercrime.

It must involve theory-based research, simulations, and even test its approach on a set of common IoT pieces in order to achieve this. The goal of this project is to provide not only an understanding of how to construct and manage shared IoT, but also practical value to those involved in security concerns involving IoT sharing. In essence, this should pave the way for broader collaborative IoT applications in a variety of fields, allowing cooperative systems to completely emerge inside our digital world.

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

Nowadays with every part of life becoming integrated with technology, we have seen the emergence of a new mode of connectedness known as the Internet of things. IoT—from basic household goods such as refrigerators and toasters to advanced industrial sensors—has transformed how we engage in our environment. The impact if IoT is significant ranging from the smart home that adjusts with the person' preference to the industry setup optimizing the efficiency through the data acquired in real time.

Though, this revolutionary wave of connectedness gives rise to an urgent issue – protection of inter-IoT infrastructures. This is essentially what IoT means involving networking and sharing of information among machines leading to a complex web of communication process. However, this connectedness leads to an array of security problems although promotes enhanced functionalities and enhanced efficiency.

The different types of IoT devices have unique specifications as well as security models, hence making it difficult to come up with one universal security framework. However, securing a single device is not the only concern; rather, the entire universe of information, comprising a myriad of connections between different devices, must be protected. In today's era of technological integration, security issues associated with shared Internet of Things (IoT) infrastructures must be addressed as we set foot.

In addition, this interdependent structure of the devices expands the attack space while worsening any possible disruption following a security violation. It involves more than just breaching an individual's right to privacy and losing confidential information. In fact, such actions may lead to breakdown of essential processes and systems including utilities and transport. With such increased responsibility in IoT devices such as monitoring of our health or controlling of industrial processes, an effective and flexible security structure is necessary. This is the entry into an analysis of the complex security issues associated with sharing IoT infrastructure. This acts as a catalyst for change that acknowledges the power of IoT and warns about the need to secure its bases against changing cyber security threats. The

following sections explore the problems, aims and motives which make this project concentrate on creating a robust Information Security Framework towards the shared IoT infrastructures.

1.2 PROBLEM STATEMENT

A new era of ubiquitous interconnectivity is being witnessed as a result of increased adoption of IoT devices. The integration of these devices is transforming homes, industries and cities into “smart” places. On the other hand, increased connectivity involves a maze of security problems. IoT devices vary greatly across the spectrum, from simple home heat control systems such as smart thermostats to complex sensor networks used in industry. This leads to different protocols and corresponding standards in use for each individual device. Such heterogeneity opens up an attack surface against which hostile actors can attack the data’s confidentiality and integrity in multi domain (shared IoT network infrastructures).

Thus, the question is how to ensure security in such an extensive and complex network of numerous interrelated units? The conventional security provisions developed for isolated systems fail to match pace with the flexibility and change associated with IoT ecosystems. These systems’ vulnerability to cyber threats result from inappropriate mechanisms for authentication, communication that is not end-to-end encrypted, and lacking standardized security protocols. The compromise of a single device could potentially trigger data breaches, violate privacy, and lead to physical outcomes in case of vulnerabilities related to critical infrastructures.

The challenge is not only a purely technological one. At its core is the cooperative characteristic of IoT that promotes additional services and capabilities. The cooperation needs a good balance between security and compatibility. Robust security while at the same time allowing for information’s hassle-free interaction across devices is a tricky business that calls for an all-round approach.

1.3 OBJECTIVES

The major goal of this project is to solve the complex security difficulties inherent in shared IoT infrastructures by integrating robust data integrity, encryption, and cloud security protections. Among the particular goals are:

- **Identifying Vulnerabilities**: Carry out thorough assessment of possible hazards in communal IoT platforms concerning information authenticity, vulnerable encryption, and cloud safety.

- **Developing Comprehensive Security Protocols**: Developing a dedicated security framework for shared IoT infrastructures. Thus, in order to maintain high data integrity, the framework will incorporate various data verification procedures, put up in-transit as well as at-rest encryption methods, and establish cloud security mechanisms aimed at protecting the system against cyber attacks.

- **Integrating Advanced Encryption Techniques**: Use modern day encryption methods to encrypt communication linkages of IoT devices at all times so as to ensure that private data cannot be accessed by third parties.

- **Ensuring Cloud Security**: Develop and enforce robust security policies in the cloud infrastructure, which house the IoT data storage and processing mechanisms. This covers access control, encryption of stored data, and the need for continuous monitoring that will help to detect and respond to threats.

- **Adopting Best Practices for Data Integrity**: Create and implement procedures that will protect the authenticity of data during its lifetime. This entails the use of check sums, digital signatures as well as any other way of identifying any form of data alteration.

- **Ensuring Interoperability with Security Measures**: Create standard and efficient mechanisms integrating security measures in multiple approaches to achieve secure operation of the IoT devices and their interoperation.

The set of objectives fits within this project and is related to securing the shared IoT infrastructures; threats to data integrity, vulnerability or weaknesses in the crypto system, cloud security risks among others. This will help realize an IOT-enabled environment that is secure and dependable in many sectors, giving confidence towards its acceptance by various groups.

1.4 SIGNIFICANCE AND MOTIVATION OF THE PROJECT WORK

1.4.1 Significance of Information Security for Shared IoT Infrastructures:

- Data Protection: It is important to keep information security while using common IoT (Internet of Things) infrastructures to protect sensitive data. The confidentiality, the integrity, as well as availability of the data must be ensured that multiple devices can interconnect and share information without creating a vulnerability gateway for intruders to access sensitive business data and cause data loss.

- Privacy Preservation: Many devices collect and exchange user data through shared IoT infrastructures.E These measures include using encryptions, access control, and secure communication protocols among others so as to keep users' details confidential.

- Preventing Cyber Attacks: Widening of the IoT environment into IoE and subsequently makes it a target for hackers. Such information security protocols are needed in order to be able to track down and combat malicious activities against the common resources, like hacking, malware, and Dos attacks.

- Maintaining Trust: In order for users and stakeholders to have faith in shared IoT environments, their data should be confidently managed in a secure manner. Information security ensures this trust is established and maintained so that more people are confident to adopt IoT technologies in different sectors.

- Compliance with Regulations: There are strict rules on data protection in many places. Enforcing strict information security policies would prevent non-compliance and any legal implications towards mismanagement of confidential information.

1.4.2 Motivation for Information Security Research in Shared IoT Infrastructures:

- Risk Mitigation: The study addresses the information security research that focuses on identifying and preventing risk occurrence in shared IoT infrastructures. It is imperative to understand vulnerabilities and develop robust counter measures so as to secure the IoT ecosystems.

- Adaptability to Evolving Threats: The aim of information security research remains the same: to be a step ahead of constantly emerging cyber threats. This is because, over time, researchers continue to innovate on security practices and technology with an aim of addressing emerging challenges and vulnerabilities arising out of a rapidly changing IOT infrastructure.

- Enhancing Resilience: A key motivator of information security research is building resistance towards the cyber-attacks. In addition, developing ways of recovering from possible security breaches implies that hostile attacks may not hinder the effective functioning of shared IoT infrastructures.

- Promoting Sustainable Development: Sustainability in Internet of things (Iot) technology development is enhanced through information security research in IoT shared infrastructure. Through dealing with security issues and making sure IoT is deployed properly and responsibly in different spheres, the scientists give the

beginning to IoT implementation.

- Facilitating Interoperability: They try to develop security standards, which would enable cross-system communications between different IoT devices and platforms. Interoperativity is of a great significance toward ensuring smooth and secured experience shared IoT environments.

1.5 ORGANIZATION OF PROJECT REPORT

The project report is organized to give a thorough examination of the essential components involved in safeguarding shared IoT infrastructures, with an emphasis on data integrity, encryption, and cloud security. The following is how the report is organized:

Chapter 2: Literature Survey

This chapter conducts a thorough analysis of the available literature on IoT security, data integrity, encryption techniques, and cloud security. It provides a solid overview of current difficulties, trends, and best practices for safeguarding shared IoT infrastructures.

Chapter 3: System Development

This chapter is the highlight of our project report. This part defines the algorithms and techniques used to apply Data integrity, data security during transmission, and the future technologies and approach being followed to integrate more technology in the project.

Chapter 4: Testing

This chapter tells about the testing methods and processes used to confirm the correctness of the algorithms used, which is an important step in the project. This chapter shows various testings namely, Unit test, Integration testing, System testing methods, and Performance testing methods used to test our code for the desired output and also use of the Cooja Simulator for Computation.

Chapter 5: Results and Evaluation

This chapter attaches the results of the applied security algorithms, Cooja Simulator, MATLAB simulation results. It assesses the Algorithms used in terms of data integrity and encryption type, and the general structure of the code.

Chapter 6: Conclusions and Future Scope

The last chapter summarizes the project's important findings, and tells about the efficacy of the applied security measures, and explores their implications. It also defines areas for future study and development, taking into account the dynamic nature of IoT technology and evolving security issues.

The organized structure allows for a systemic investigation process – the basic findings collected in the literature survey up to the actual implementation, testing and assessment of the suggested security framework. The chapter on conclusions summarizes the key achievements and serves as a springboard for efforts directed towards improved security for Shared IoT infrastructure.

CHAPTER 2: LITERATURE SURVEY

For comprehensive insight into the state of knowledge concerning the issues regarding the information security of the shared IOT infrastructures, it is crucial to conduct an extensive review of literature. There is an extensive evaluation of different scholarly sources such as research journals, papers, and records concerning the IoT security aspects.

The survey is used as a basis of the study and reveals different security problems and possible approaches analyzed by academic and trade society. The paper aims to provide a basis for understanding how security protocols and technologies have evolved historically for shared IoT environments.

A literature review also helps to establish gaps in existing knowledge and identify areas of the study that require more research. The comprehension of this is vital in developing research problems and objectives that will address or improve on current security concerns not yet resolved by other security systems.

2.1 OVERVIEW OF RELEVANT LITERATURE

E. Kim et al. [1] developed efficient role-based security frameworks for shared IoT infrastructures. Recognizing the complexity and interconnectedness of IoT settings, their objective was to create models that use role-based access control techniques to improve overall security posture. The emphasis was on creating strong and scalable solutions capable of accommodating the various roles and permissions inherent in shared IoT networks.

D. Garcia et al. (2022) [2] researched and proposed privacy-preserving strategies for safe data exchange in the Internet of Things. They wanted to address the rising concerns about individual privacy and sensitive information in the context of the Internet of Things. Their primary focus was on inventing mechanisms for exchanging important IoT data while limiting the danger of privacy breaches.

A. Patel et al. (2021) [3] carried out a thorough comparison of existing security frameworks built for shared IoT settings. The study sought to examine the frameworks' strengths and shortcomings in tackling the specific security concerns provided by networked IoT devices in collaborative settings.

C. Wang et al. (2020) [4] used blockchain technology to research and propose a unique authentication strategy for shared IoT networks. The goal was to improve the security of IoT devices in collaborative situations by exploiting blockchain's decentralized and tamper-resistant characteristics for authentication and access management.

The major goal of H. Gupta et al. (2020) [5] was to propose secure communication protocols adapted to the particular issues faced by shared IoT applications. The objective was to create protocols that provide strong security while also allowing the dynamic and heterogeneous character of common IoT applications.

D. Singh et al. (2020) [6] focused on identifying and evaluating possible vulnerabilities, recognizing the vital role that security plays in the reliable functioning of smart city infrastructures. It sought to provide insights that may be used to design effective security measures adapted to the particular problems faced by smart city IoT applications.

The primary goal of T. Rodriguez et al. (2018) [7] was to design encryption algorithms that may provide a balance of security and efficiency. The purpose was to protect sensitive data shared by various organizations in IoT applications by combining complementary encryption techniques.

R. Tadayoni et al. (2017) [8] sought to present a comprehensive overview of the vast spectrum of vulnerabilities confronting IoT ecosystems, covering both applications and broader service domains. The purpose was to provide insights that may be used to design effective cybersecurity measures adapted to the particular issues posed by the linked nature of IoT.

2.1.1 TABLE STATING OBJECTIVES OF VARIOUS RESEARCH PAPERS ANALYSED DURING OUR RESEARCH ON THIS PROJECT

Table 2.1.1

S No.	Paper Title	Year	Tools/Techniques	Results
1.	Role-Based Security Models in Shared IoT Infrastructures	2023	Role-Based Security model	Not specified
2.	Privacy-preserving Techniques for shared IoT Data: A survey and analysis	2022	Privacy-Preserving Techniques	Survey and analysis
3.	Security Framework for Shared Environments: IoT Comparative Study	2021	Security frameworks	Comparative study
4.	Blockchain-Based Authentication in Shared IOT Networks	2020	Blockchain-based Authentication	Authentication in shared IoT networks
5.	Secure Communication Protocols for Shared IoT Applications: A Survey Comparison	2020	Secure Communication protocol	Survey and comparison

S No.	Paper Title	Year	Tools/Techniques	Results
6.	Identifying Analyzing Potential vulnerabilities in Smart Infrastructures: A Systematic Literature Review	2020	Vulnerability analysis	Systematic literature review
7.	Developing Encryption Methods for Shared IoT Data: Balancing Sectrity and Efficiency	2018	Encryption Methods	Balancing Security and Efficiency
8.	Efficient and Secure Communication in IoT Using Contiki and RPL	2018	Contiki OS, Cooja simulator, RPL protocol	Demonstrated efficient and secure communication in IoT networks using Contiki OS and RPL. Key Finding: Improved reliability and security in IoT deployments. Drawback: Limited evaluation in real-world scenarios.
9.	Security Analysis of Contiki-based Systems in IoT Environment	2017	Contiki OS, Cooja simulator, Security analysis techniques	Identified security vulnerabilities in Contiki-based IoT systems. Key Finding: Importance of security measures in IoT deployments. Drawback: Limited focus on specific attack scenarios.
10.	Wireless Sensor Network Security: A Survey	2017	Survey methodology, Literature review	Comprehensive overview of security issues and existing solutions in WSNs. Focuses on general WSN security, may not delve deeply into specific implementations on Contiki OS.

S No.	Paper Title	Year	Tools/Techniques	Results
11.	Secure Key Management Scheme for IoT Infrastructures using MATLAB	2017	MATLAB, Key management techniques	Proposed a secure key management scheme for IoT infrastructures using MATLAB. Key Finding: Enhanced security for shared IoT environments.
12.	Privacy-preserving Data Aggregation in Shared IoT Networks using Simulink	2016	Simulink, Privacy-preserving data aggregation techniques	Presented a privacy-preserving data aggregation method for shared IoT networks using Simulink. Key Finding: Improved data privacy without sacrificing efficiency.
13.	Detection and Mitigation of Insider Threats in Collaborative IoT Environments with MATLAB	2015	MATLAB, Insider threat detection techniques	Developed a framework for detecting and mitigating insider threats in collaborative IoT environments using MATLAB. Key Finding: Enhanced security against insider attacks.
14.	Secure Communication Protocol for Shared IoT Infrastructures using Simulink	2014	Simulink, Secure communication protocols	Proposed a secure communication protocol for shared IoT infrastructures using Simulink. Key Finding: Enhanced data confidentiality and integrity in shared IoT environments.
15.	Anomaly Detection in Shared IoT Infrastructures using Simulink	2013	Simulink, Anomaly detection techniques	Proposed an anomaly detection method for identifying malicious activities in shared IoT infrastructures using Simulink. Key Finding: Early detection and mitigation of security threats.

2.2 KEY GAPS IN THE LITERATURE

Looking across various analyzed studies, there exist common lacks in comprehension and practice on safe IoT apps. The absence of standardized frameworks and taxonomies remains an issue which is highlighted by “Security Issues in IoT and their countermeasures in smart city application.” In fact, it indicates that it is highly crucial to establish uniform classification schemes to deal with diverse security issues associated with IoT.

This section also touches on another prevalent hole which is that of adapting security solutions to the constrained nature of IoT devices. It further says that there exist only some hybrid encryption means which can be applied for data security sharing applications.

In addition, a common gap noted in the research, such as "Cybersecurity Threats to IoT Applications and Service Domains," is the lack of globally acknowledged best practices. This disparity indicates a lack of defined criteria for protecting various IoT applications, which impedes the creation of uniform and scalable security mechanisms. The development of comprehensive models that smoothly incorporate numerous security techniques, as suggested by "Hybrid Encryption Techniques," is critical to closing this gap.

It is important to note that these key gaps are the necessity for standard taxonomies, adaptation of security measures according to scarce resources, and establishment of universally acknowledged good practices. One should overcome these rifts in order to improve the area of IoT safety, guaranteeing consistency among products and developing hard security against the various changing face of IoT applications.

CHAPTER 3: SYSTEM DEVELOPMENT

3.1 REQUIREMENTS AND ANALYSIS

The rigorous identification and analysis of requirements at the foundational phase of system development is critical for the success of the "Information Security for Shared IoT Infrastructures" project. The requirements are divided into two categories: functional and non-functional requirements.

3.1.1 FUNCTIONAL REQUIREMENTS:

- **Secure Data Transmission:**

The system must facilitate secure communication between shared IoT devices. It should implement robust end-to-end encryption protocols such as TLS/SSL to protect data during transit. Additionally, the system should employ secure communication libraries to ensure the confidentiality and integrity of shared information.

- **Key Management:**

A safe mechanism for creating, storing, and exchanging cryptographic keys should be provided by the system. Using key management systems and hardware security modules (HSM) can help to ensure the security of sensitive cryptographic information by contributing to a strong key lifecycle management process.

- **Data Integrity Assurance:**

It is critical to ensure the integrity of shared data. To perform regular integrity checks during data transmission, the system should use hash functions, digital signatures, and checksums. This guarantees that data is unaltered and trustworthy throughout its existence.

- **Node Setup and Configuration:**

In this step we define the prerequisites for building up and configuring IoT nodes in the Contiki-Cooja simulator and specify parameters such as node ID, communication range, transmission power, and so on.

- **Broadcast Messaging:**

Nodes should be able to broadcast messages to all other nodes in the network and specify the format and structure of these messages. Also, specify the maximum message size and how messages are split (if necessary) during transmission.

- **Routing Protocol:**

In this there is the need to define the routing protocol for broadcast communications. Specify how nodes find nearby nodes and maintain routing tables.

- **Energy Efficiency:**

There is need to Implement energy-efficient communication protocols to reduce node energy consumption and require to Define the sleep modes and duty cycling techniques to save energy while nodes are idle.

- **Node Simulation:**

Create a MATLAB model to simulate IoT nodes. We need to specify factors such node ID, location, transmission power, and communication range.

- **Testing and Validation:**

In this there is a need to define test scenarios and methods to validate the system's functionality and performance and even ensure that the system satisfies the required standards under a variety of situations and settings.

Features:

- End-to-end encryption techniques (e.g., TLS/SSL) should be used for safe data transport.
- To validate and maintain the integrity of shared data, use hash functions and digital signatures.
- Use secure communication libraries to prevent data alteration while in transit.

3.1.2 NON-FUNCTIONAL REQUIREMENTS:

- **Scalability:**

The system's capacity to manage a rising amount of devices, data, and user interactions efficiently and effectively while preserving maximum performance and security. It is critical to guarantee that the security infrastructure can adapt and develop to meet the changing needs and demands of the shared IoT environment. Scalability in information security is critical for fitting the dynamic nature of shared IoT infrastructures, allowing them to scale while maintaining security, performance, and user experience.

- **Performance:**

Security measures should be applied without impairing system performance considerably. The system should handle cryptographic activities effectively, and performance bottlenecks should be identified and addressed on a regular basis.

- **Usability:**

While maintaining security, the system should also be user-friendly. To guarantee that security measures do not impede the use of the shared IoT infrastructure, intuitive interfaces for device administration and clear user instructions for security-related activities are critical.

- **Reliability:**

  Refers to the system's capacity to offer secure services consistently and predictably, to maintain continuous availability, and to reduce interruptions. It is critical to ensure the dependability of security measures in order to foster confidence among stakeholders and ensure the constant protection of shared IoT devices and data. Information security reliability for shared IoT infrastructures is critical for building a safe and trustworthy environment. The dependability of security measures contributes considerably to the overall success of the shared IoT ecosystem by prioritizing continuous availability, proactive issue response, and user confidence.

The "Information Security for Shared IoT Infrastructures" project seeks to create a safe and dependable environment for the shared use of IoT devices by addressing both functional and non-functional criteria.

3.1.3 INFORMATION SECURITY FOR INTERNET OF THINGS (IOT) DEVICES

3.1.3.1 What is Information security with respect to IoT devices?

In today's linked world, protecting our smart devices and equipment from possible attacks is critical. These technologies, whether in our homes to make our lives easier or in factories to power industrial operations, play an important part in our everyday lives. Keeping them safe entails not only safeguarding the data they collect, but also ensuring that only authorized devices may connect with one another. It's like giving each machine a secret code to ensure they're communicating with the correct partners. Furthermore, we must consider who has physical access to these gadgets, just as we would with our own things. The networks that connect these devices function as superhighways, and adding security measures such as firewalls is analogous to having watchful guards defending these digital roads from possible attacks.

Because our privacy is vital, the design of these gadgets prioritizes the confidentiality of our personal information. Consider it a digital veil erected to protect our personal information from prying eyes. In the case of a security breach, it is important to have a thorough plan to ensure a quick resolution and a comprehensive investigation to determine what went wrong. Safeguarding these devices is a continuous process that includes coordination, adhering to

established security procedures, and occasionally investigating new and inventive technologies to keep one step ahead of any threats. The necessity of safeguarding the security and integrity of these smart gadgets in our linked world grows as our dependence on them grows.

The deployment of procedures to secure the confidentiality,and integrity of data inside the IoT ecosystem is information security for IoT (Internet of Things) devices. Given the growing use of IoT devices in various of industries, including households, businesses, and healthcare, guaranteeing the security of these devices is critical.

3.1.3.2 Why is Information Security Needed in IoT Devices?

1. **Data Confidentiality:**
    - IoT devices frequently acquire sensitive data, such as personal, medical, or industrial information. It is critical to ensure the security of this data in order to prevent illegal access and safeguard user privacy.Fig. [3.1]

2. **Prevention of Unauthorized Access:**
    - Unauthorized access to IoT devices can result in a variety of dangers, including data tampering, service disruption, and possible abuse. Unauthorized parties are kept out of these gadgets thanks to information security procedures. Fig. [3.1]

3. **Integrity of Data:**
    - Maintaining data integrity is crucial. Data security protections prevent data from being tampered with or altered during transmission or storage, ensuring that the data is correct and trustworthy. Fig. [3.1]

4. **Protection Against Cyber Attacks:**
    - Malware, ransomware, and distributed denial-of-service (DDoS) assaults are all possibilities for IoT devices. Strong information security procedures protect against these attacks, minimizing possible harm and interruption. Fig. [3.1]

5. **Prevention of Data Theft:**
    - The data collected and processed by IoT devices has the potential to be lucrative. Unauthorized entities are prevented from obtaining or utilizing this data for malevolent reasons such as identity theft or industrial espionage by information security measures. Fig. [3.1]

6. **Reduced Risk of Exploitation:**
    - IoT devices might become targets for exploitation if information security is not properly implemented. These devices' vulnerabilities can be used to obtain unauthorized control, interrupt operations, or compromise the entire IoT network. Fig. [3.1]

7. **Trust in the Ecosystem:**
    - IoT's interconnectedness provides an ecosystem in which gadgets communicate and exchange data. Within this ecosystem, information security fosters trust, ensuring that devices may communicate with one another in a reliable and secure manner. Fig. [3.1]

8. **Prevention of Denial of Service (DoS) Attacks:**
    - Information security helps to avoid DoS attacks that have try to interrupt the availability or the performance of IoT devices or the whole network. Without adequate security measures, broadcast communication channels are vulnerable to flooding assaults and other types of network congestion, resulting in service outages. Fig. [3.4]

9. **Prevention of Replay Attacks:**
    - Security systems prevent from the replay attacks, in which an attacker intercepts and retransmits legitimate communications in order to obtain unauthorized access or disrupt network operations. Replay attacks in broadcast communication may result in unauthorized data access or malicious modification. Fig. [3.4]

10. **Authorization:**
    - Authorization methods govern access to resources and capabilities in the IoT network. By implementing adequate authorization policies, security guarantees that only authorized devices have the permissions to conduct certain tasks, lowering the risk of illegal network resource manipulation. Fig. [3.4]

Summarizing everything, information security is vital for IoT devices for protecting data, stop unauthorized access to the data, and ensure that the IoT Devices communicate with each other smoothly. As IoT evolves, considering and solving security problems becomes more and more important.

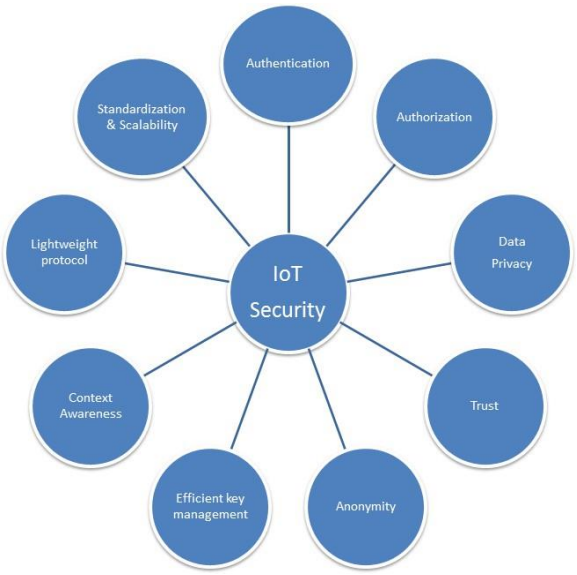

Figure 3.1: IoT Security Features

3.1.3.3 How to achieve Information security IoT devices?

To get information security in shared IoT devices, a detailed set of algorithms must be applied to ensure data integrity and data encryption and prevent data from unauthorized access. Following are some ways in which security in shared IoT devices can be increased/achieved:

1. **Device Authentication:**
    - Use of Strong authentication measures will guarantee that only authorized and valid devices connect and communicate within the network.
    - Implementing Key management programs can improve security.

2. **Data Encryption:**
    - To prevent the data from being modified by someone without access, encrypting data during transmission and storage is a good practice. To protect the communication channel, strong encryption methods like AES/RSA can be applied.

3. **Access Controls:**
    - Implement strong access controls to limit who can access and modify data.
    - To assign particular rights depending on user roles, apply role-based access control.

4. **Network Security:**
    - Protecting the IoT network from unwanted access and attacks includes using firewalls, NIPS/NIDS systems, and network security.
    - To safely send data between devices, we can use secure TSL/SSL communication protocols.

5. **Physical Security:**
    - Implement physical security measures to protect IoT devices from illegal access or manipulation.

6. **Privacy Protection:**
    - IoT devices should be designed with privacy in mind, adopting privacy-by-design principles.
    - To safeguard user privacy, anonymize and aggregate data whenever possible.

7. **Blockchain Technology:**
    - Investigate the use of blockchain for safe and tamper-resistant record-keeping. By offering a decentralized and distributed ledger, blockchain can improve data integrity.

8. **Secure Communication Protocols:**
    - Encrypt data during transmission using secure communication protocols such as MQTT with TLS/SSL to protect the confidentiality and integrity of exchanged information.

Organizations may develop a strong information security framework for shared IoT devices by following these techniques, supporting a safe and trustworthy environment for data sharing and collaboration.

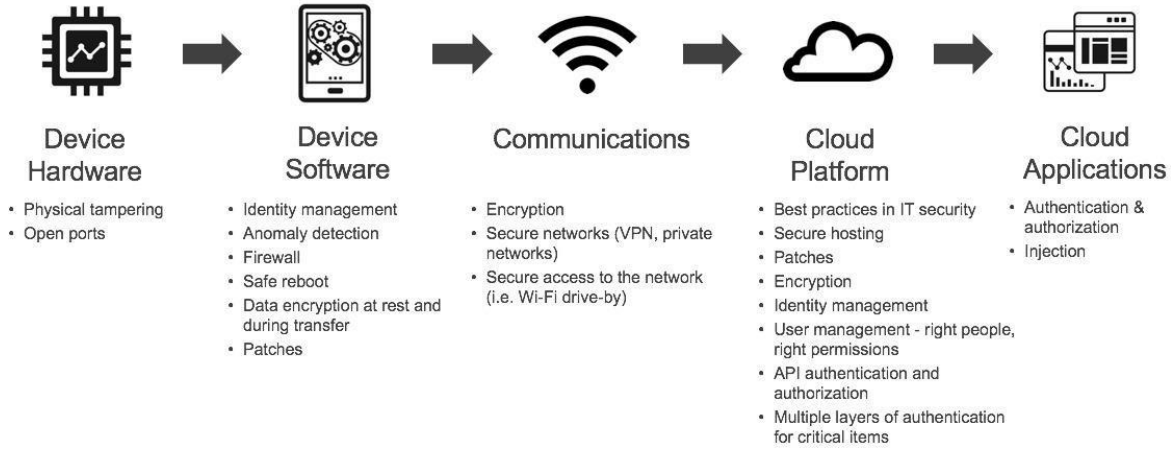

Figure 3.2: IoT Techniques at each step of IoT communication

3.2 PROJECT DESIGN AND ARCHITECTURE

The basic concept of our "Secure IoT Framework" project is to integrate strong security mechanisms to protect common IoT infrastructures. The project aims to improve the security of shared IoT infrastructures by implementing the TLS/SSL protocol, data integrity protections, MQTT transmission, broadcast Communication and the RSA algorithm. The design stresses a strong architecture that can adapt to changing security requirements and include cloud technologies in the future.

1. **Data Integrity Assurance:**
    - Use data integrity methods to check and protect the correctness of data sent between devices. Hash functions and checksums will be used to identify any illegal data changes.

2. **RSA Algorithm for Encryption:**
    - Encrypt data using the RSA technique, which provides a safe way of data security. This asymmetric encryption technique ensures that only authorized parties have access to and decipher the delivered data.

3. **Cloud Technology Integration:**
    - Plan for the incorporation of cloud technologies to improve the security infrastructure's scalability, flexibility, and accessibility. Cloud-based security administration, upgrades, and real-time monitoring may all be facilitated by cloud-based systems.

4. **Access Controls and Authentication:**
    - To restrict device interactions, use strict access controls and authentication systems. This covers protocols for user authentication, device identification, and permission to guarantee that only authorized organizations have access to sensitive data.

5. **Node Design:**
    - Every node in the network represents a sensor device. Nodes will be configured to sense data on a regular basis and broadcast messages to their neighbors. Nodes will also process received messages and take necessary actions depending on their content.

6. **Routing and Network Formation:**
    - The network will employ a routing protocol (such as RPL) to ensure effective data forwarding and network construction and the nodes will use routing tables to identify the next hop for broadcasting messages.

7. **Monitoring and Analysis:**
    - The project will include tools for monitoring network performance and assessing simulation findings. The broadcast communication system's efficacy will be assessed using metrics such as message delivery ratio, latency, and energy usage.

8. **Energy Management:**
    - Nodes will use energy-efficient communication mechanisms to preserve battery power. MATLAB and Simulink can simulate energy-saving processes including duty cycle and sleep modes.

9. **Simulation Environment:**
    - The simulation environment will be created using MATLAB and Simulink. The network topology, node characteristics, and simulation settings will be configured in Simulink.

The goal of combining these features into the project design and architecture is to create a safe, adaptable, and future-ready infrastructure for shared IoT settings that prioritizes information security.

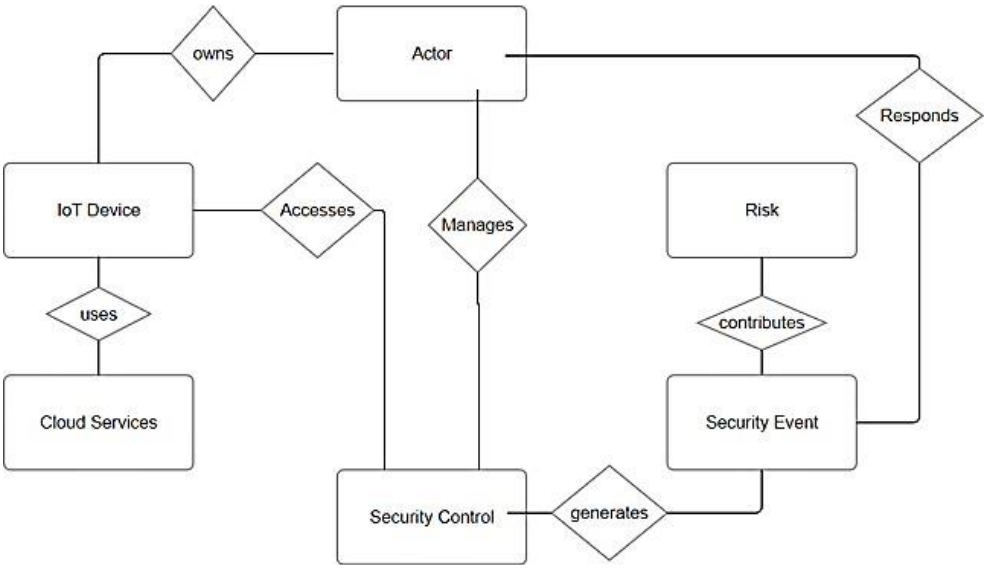

Figure 3.3: Entity Relationship Diagram of IoT Device connection with user and control

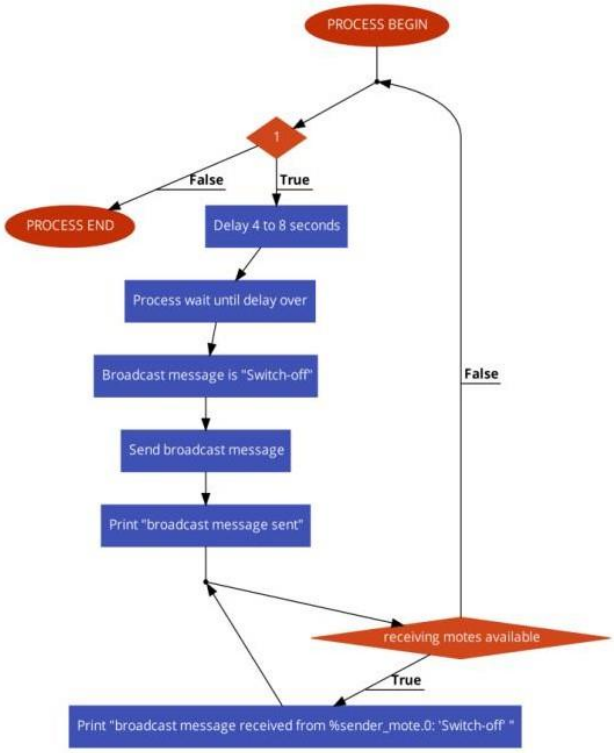

Figure 3.4: Broadcast Communication

3.2.1 Current Progress on the Architecture

Our current project's primary focus is on ensuring the security of data shared in IoT installations. We have successfully introduced functionality for verifying and safeguarding the integrity of shared data. In addition, we used the RSA method to make the information secure during transmission.

Our methodology is adaptable, allowing us to continue to improve as we learn more and as security requirements evolve. In the current scenario, we use the Cooja Simulator along with the Contiki OS in order to compute the secure transmission between the communication of messages. This will provide us with a full solution that will cover everything from data security to data transport safety.

We're now employing explicit tracking measures to keep a careful check on the security aspects. This enables everyone participating in the project to make sound judgments and collaborate efficiently. As we progress, the combined power of data integrity, the RSA algorithm, and the TLS/SSL protocol will provide a robust security basis for shared IoT infrastructures, we also use MATLAB and SIMULINK respectively in order to get results about the script information of virtual IoT that allowing us to accomplish our project aim of keeping everything safe and secure.

3.3 DATA PREPARATION

3.3.1 Data Preparation for "Information Security in Shared IoT Infrastructures"

In the context of our project concentrating on safeguarding common IoT infrastructures, thorough data preparation is a critical step to assure sensor data integrity and confidentiality. We used data files in CSV and XLSX formats with two crucial columns: Value and State, for this purpose.

1. **File Formats and Structure:**
    - **CSV and XLSX:** We took use of the adaptability of both CSV and XLSX file formats to accommodate the various data processing choices. Each file included two columns: Value and State, which contained sensor data.

2. **Data Cleaning and Formatting:**
    - A thorough data cleansing procedure was done prior to implementing security measures. This included dealing with missing numbers, deleting duplicates, and dealing with any outliers in the data. The purpose was to create a uniform and trustworthy dataset for further study.

3. **Data Integrity Measures:**
    - We used data integrity techniques to confirm the sensor data's correctness and consistency. Implementing hash functions and checksums to produce unique signatures for each dataset was part of this. We may detect any unlawful adjustments or corruption in the data by comparing these signatures.

4. **Application of the RSA Algorithm:**
    - We used the RSA technique to ensure the secrecy of sensor data during transmission. Only authorized organizations with the relevant decryption key could access and understand the data using this asymmetric encryption technology. This additional degree of protection was critical for preventing unwanted access to sensitive information.

5. **Cross-Format Consistency:**
    - Consistency across CSV and XLSX formats was a top objective. To eliminate differences that may jeopardize data integrity and security, we ensured that the sensor data remained consistent in both forms.

6. **Key Management for RSA:**
    - For the RSA algorithm, proper key management methods were adopted. This comprised the creation, distribution, and storage of secure keys. The meticulous handling of encryption and decryption keys guaranteed that the security measures used were successful.

7. **Key Management for COOJA Simulator & MATLAB with Simulink:**
    - Create and handle cryptographic keys securely. Implement key exchange methods such as Diffie-Hellman or pre-shared keys. Create key management methods in Simulink or MATLAB that handle key creation, distribution, and storage securely.

8. **Packetization:**
    - Divide data into packets that may be sent via a network. Use Simulink blocks or MATLAB routines to divide the data into frames or packets.

9. **Documentation:**
    - The data preparation methods, integrity checks, and encryption processes were meticulously documented.

Figure 3.5: Data Files of 2 IoT Devices

Value	State
10	In
10	Out
10	Out
10	In
10	Out
25	In
60	Out
40	Out
32	In
32	Out
9	In
26	Out
18	Out
17	In

10	In
10	Out
45	In
10	In
42	In
25	In
60	Out
40	Out
3	In
32	Out
9	In
26	Out
18	Out
17	In
5	Out
28	In

Figure 3.6: Data Values of 2 IoT Devices in csv and xlsx

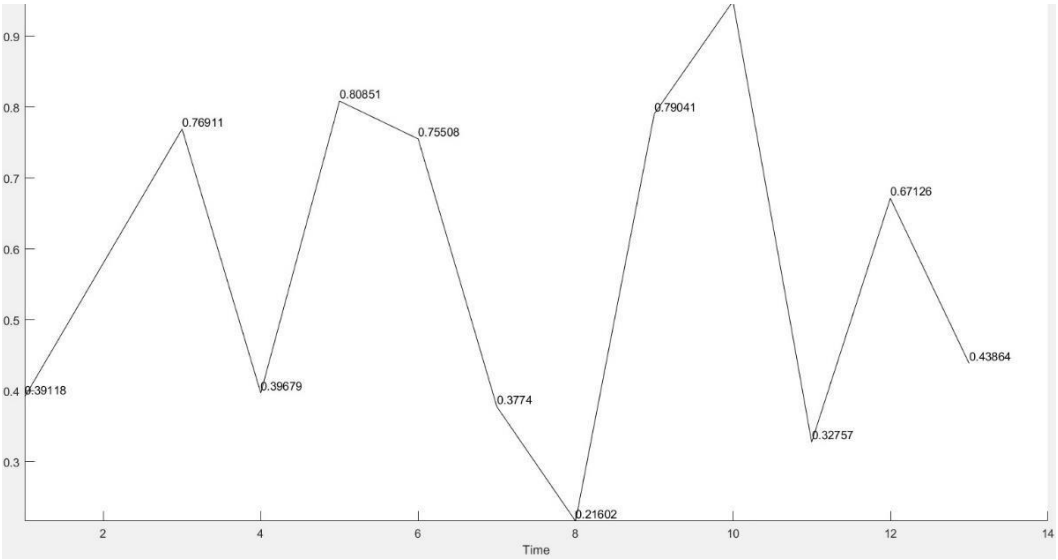

Figure 3.7: Generating data from the IoT sensor and displaying it through a graph

3.4 IMPLEMENTATION

In the ongoing execution of our project on "Information Security for Shared IoT Infrastructures," we are pursuing a holistic approach to data integrity and confidentiality. As demonstrated by the accompanying Python script, the project leverages a strong coding structure to protect data integrity using hashing. This script use the SHA-256 hashing technique to compute hash values for datasets acquired from various IoT devices, which are then compared for verification. Furthermore, the project contains enhanced security measures by utilizing the RSA method to encrypt and safeguard critical information transfer. The RSA method uses asymmetric encryption, which means that only authorized organizations with the associated decryption key may access and decipher the sent data. The combined use of data integrity checks and the RSA algorithm creates a multi-layered security architecture that protects the integrity and confidentiality of shared IoT data. The algorithm not only validates data integrity using hash comparisons, but it also displays particular rows where disparities occur, allowing for further in-depth study of any data mismatches. Current iterations of our project includes additional security measures, such as the use of COOJA Simulator for the secure transmission protocol and also the use of MATLAB with the Simulink tool to get more secure and enhance computational security and safety results followed by integrity and consistency.

3.4.1 TOOLS AND TECHNIQUES USED

1. **PYTHON:**

   Python is frequently used in information security for a variety of reasons, most notably its versatility, simplicity of use, and huge ecosystem of libraries and frameworks. Python's adaptability, large library, community support, and simplicity of integration make it an invaluable tool in information security applications. Python may be useful in automating processes, applying cryptographic approaches, evaluating security data, and smoothly integrating security measures into the larger IoT ecosystem in the context of a project requiring security algorithms for IoT applications, such as ours.

Python is useful for information security because of the following characteristics:

1. **Scripting and Automation:**
    - Python's scripting features enable security experts to automate tedious activities. Automation is critical in security operations for duties such as scanning, monitoring, and reaction actions.
    - Python may be used in the project to automate security operations such as algorithm execution. Python scripts, for example, might simplify the deployment of these techniques to huge datasets or in real-time data processing in a project concentrating on encryption/decryption (as demonstrated in "Hybrid Encryption Techniques for Data Security in Shared IoT Applications").
2. **Extensive Libraries and Frameworks:**
    - Why Python is Beneficial: Python provides a large number of modules and frameworks related to information security. Cryptography, PyCryptoDome, and other libraries give tools for developing cryptographic algorithms.
    - In the Project: The cryptography modules provided by Python may be used to create the hybrid encryption strategies outlined in the project. High-level interfaces for secure communication and data protection are provided by these libraries.
3. **Data Analysis and Visualization:**
    - Python's data analysis libraries (e.g., Pandas, NumPy) and visualization tools (e.g., Matplotlib, Seaborn) are useful for analyzing security logs, recognizing trends, and visually presenting findings.
    - Python is used in the project to view security issues created during the implementation of security algorithms. Visualization tools also help in the presentation of important trends.
4. **COOJA Simulator:**
    - Cooja is the main tool for modeling IoT networks. It offers a graphical interface for simulating large-scale IoT networks, enabling developers to test

and analyze their protocols and applications. Cooja interfaces with Contiki OS, a popular operating system for IoT devices. Contiki offers a lightweight, event-driven kernel that is ideal for resource-constrained devices.

5. **Broadcasting:**
    - Cooja enables broadcasting, which is when a node sends a packet that is received by all other nodes within the radio range. This is a critical strategy for effectively spreading information across IoT networks.

6. **Energy Consumption Analysis:**
    - Cooja allows us to analyze the energy usage of virtual IoT devices during broadcast communications. This research is critical for determining the influence of communication protocols on the network's overall energy efficiency.

7. **MATLAB/Simulink:**
    - MATLAB and Simulink offer a complete platform for modeling, simulating, and evaluating dynamic systems, such as IoT networks. Simulink enables graphical modeling of system dynamics, making it appropriate for IoT communication simulations.
    - The MATLAB IoT Toolbox includes functions and tools for developing, modeling, and evaluating IoT systems. It comprises blocks and functions for simulating wireless communication, sensor networks, and Internet of Things protocols.

8. **Wireless Communication Blocks:**
    - Simulink has blocks for simulating a variety of wireless communication protocols, including broadcast communication. These blocks model the transmission and receipt of data packets between IoT devices in a simulated environment.

9. Documentation:

- In the Project: Python's community support and documentation can be of great help when we try to troubleshoot issues. We can view documentation on the algorithm implementation.

10. Integration with different Technologies:

- Why Python is Helpful: Python's smooth compatibility with other technologies and tools facilitates integration with existing security tools, making it agood language for building projects.
- In the Project: Python can be used to integrate the security algorithms with other components of the IoT devices and network. For example, integrating encryption techniques and data integrity into communication protocols like tsl/ssl or data storage systems.

11. Easy Development:

- In the Project: The Python community can be used to debug bugs, seek assistance on algorithm implementation, and remain updated on security best practices.

Python's adaptability, large library, community support, and smoothness of integration make it an invaluable tool in information security applications. Python can be useful in automating processes, applying cryptographic techniques, assessing security data, and smoothly integrating security measures into the IoT ecosystem for a project including security algorithms for IoT applications.

Other Python libraries that proved to be really useful during the project are listed below:

Libraries:

1. **pandas (import pandas as pd):**
    - Used to manipulate data and perform DataFrame operations.
2. **hashlib (import hashlib):**
    - Used in cryptographic hash functions. For determining hash values, the SHA-256 method is used in particular.

```
import pandas as pd
import hashlib
```

Figure 3.8: Import statement of Python pandas library

3. **cryptography.hazmat.primitives.asymmetric (from cryptography.hazmat.primitives.asymmetric import rsa):**
    - Using the cryptography library's RSA module to implement the RSA algorithm.
4. **cryptography.hazmat.primitives import serialization (from cryptography.hazmat.primitives import serialization):**
    - Serializes and deserializes cryptographic objects.
5. **cryptography.hazmat.primitives.asymmetric import padding (from cryptography.hazmat.primitives.asymmetric import padding):**
    - Padding techniques for asymmetric encryption, a critical component of the RSA algorithm, are provided.

```
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import serialization, hashes
from cryptography.hazmat.primitives.asymmetric import rsa, padding
```

Figure 3.9: Import statements of cryptography.hazmat files.

Special Functions:

1. **calculate_hash(data):**

   - Custom function that uses the SHA-256 method to compute the hash value for a given dataset.

```
    # Calculate hash of the data
    data_hash = calculate_hash(data_string)

    return data_hash, df
```

Figure 3.10: Calculate_hash function of SHA-256 algorithm

2. **read_and_verify_data(file_path, file_format):**

   - Custom function that reads data from CSV or Excel files, converts it to a string, computes the hash, and returns the hash result as well as the DataFrame.

```
# Function to read data from CSV or Excel file and calculate hash
def read_and_verify_data(file_path, file_format):
    try:
        # Read data from file using pandas
        if file_format == 'csv':
            df = pd.read_csv(file_path)
        elif file_format == 'xlsx':
            df = pd.read_excel(file_path)
        else:
            print("Unsupported file format. Please provide a CSV or Excel file.")
            return
```

Figure 3.11: User input prompt function for csv or xlsx data file

3. **RSA Algorithm Functions:**

- The RSA algorithm is used for key creation, encryption, and decryption. These functions include the generation of RSA key pairs, the encryption of data with a public key, and the decryption of data with a private key.

```
def generate_key_pair():
    private_key = rsa.generate_private_key(
        public_exponent=65537,
        key_size=2048,
        backend=default_backend()
    )
    public_key = private_key.public_key()

    return private_key, public_key
```

```
def encrypt_data(data, public_key):
    encrypted_data = public_key.encrypt(
        data.encode(),
        padding.OAEP(
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )
    return encrypted_data
```

Figure 3.12: Functions for Key generation, Encryption and decryption for Rsa Algorithm

The employment of these libraries and bespoke functions in tandem in the project's code guarantees a strong information security foundation. The project's main objective of safeguarding common IoT infrastructures is helped by the combination of data integrity checks and the RSA algorithm.

2. DATA INTEGRITY

- **Data Integrity in General:**

Data integrity is the quality, consistency, and dependability of data during its entire lifespan. It is the guarantee that data will remain intact and uncorrupted during storage, transport, or processing. Maintaining data integrity is critical for guaranteeing information reliability, and it is a cornerstone of information security and data management procedures.

- **Key Aspects of Data Integrity:**

1. **Accuracy:** It is assurance that the data is correct and error-free.

2. **Consistency:** Ensuring that data is uniform and coherent across systems and throughout time.

3. **Reliability:** It is the assurance that the data can be trusted and will be available when needed.

4. **Completeness:** Ensure that all necessary information is there and accessible.

5. **Data security** refers to safeguarding data from illegal access, alteration, or corruption.

- **Data Integrity in IoT Devices:**

Data integrity is especially important in the context of Internet of Things (IoT) devices because of the decentralized nature of IoT ecosystems and the continual collection, transfer, and processing of data by many associated devices. Here are some particular considerations for data integrity in IoT:

1. **Real-Time Monitoring:**
    - IoT devices frequently work in real-time contexts, necessitating continual data integrity monitoring. Any errors or contradictions in the data might have immediate and serious effects.

2. **Secure Communication:**
    - It is critical to ensure data integrity during transmission. Insecure communication routes are prone to data modification or interception, jeopardizing the integrity of data transferred between IoT devices.

3. **Diverse Data Sources:**
    - Sensors, actuators, and other devices are common data sources in IoT contexts. To ensure the integrity of data from these many sources, consistent methods and meticulous validation processes are required.

4. **Security Protocols:**
    - Strong security mechanisms, such as cryptographic hashing and encryption, must be implemented. These safeguards protect sensitive data from unauthorized alterations and guarantee that only authorized entities may access or edit it.

5. **Data Lifecycle Management:**
    - Effective data integrity management requires taking into account the full data lifecycle, from production to transmission to storage and processing. Establishing defined rules for each stage protects the data's ongoing integrity.

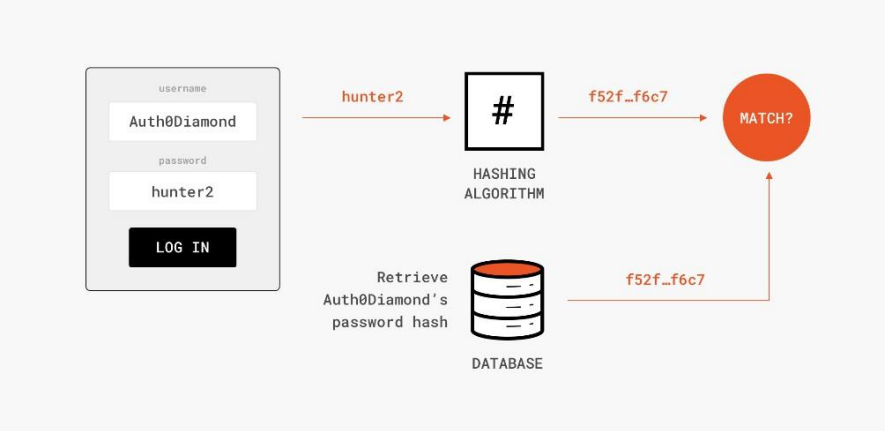

Figure 3.13: Data Integrity Methodology Diagram and Hashing

- **Data Integrity Applied in This project following these steps:**
    1. **User Input:**
        - The user is asked to provide the file locations and formats for data from two IoT devices (Devices 1 and 2). This data is essential for later data reading and integrity checks.

```
# User input for file paths and types
file_path_device1 = input("Enter file path for data from Device 1: ")
file_format_device1 = input("Enter file format for data from Device 1 (csv or xlsx): ").lower()

file_path_device2 = input("Enter file path for data from Device 2: ")
file_format_device2 = input("Enter file format for datFa from Device 2 (csv or xlsx): ").lower()
```

Figure 3.14: User Input code and File format decision query

    2. **Reading Data from Files:**
        - Using the pandas library, the code reads data from CSV or Excel files using the user-supplied file locations and formats. Two DataFrames are constructed (df_device1 and df_device2), each containing sensor data from a different device.

```
# Function to read data from CSV or Excel file and calculate hash
def read_and_verify_data(file_path, file_format):
    try:
        # Read data from file using pandas
        if file_format == 'csv':
            df = pd.read_csv(file_path)
        elif file_format == 'xlsx':
            df = pd.read_excel(file_path)
        else:
            print("Unsupported file format. Please provide a CSV or Excel file.")
            return
```

Figure 3.15: Data reading and Data file integration code

3. **Data to String Conversion:**
    - The to_string function is used to transform the data in each DataFrame to a string. This step guarantees that the entire dataset is represented as a single string, which simplifies the hashing procedure later on.

```
# Convert DataFrame to a string for hashing
data_string = df.to_string(index=False)
```

Figure 3.16: to_string function for DataFrame conversion to string

4. **Hash Calculation:**
    - Each data string is subjected to the calculate_hash function. This method employs the hashlib library's SHA-256 hashing technique to produce unique hash values (hash_device1 and hash_device2) for the datasets from Devices 1 and 2.

5. **Comparison of Hash Values:**
    - The hash values acquired from the two devices are compared in the code. If the hash values are the same, the data from both devices is likely to be the same.

```
# Verify data integrity for both devices
hash_device1, df_device1 = read_and_verify_data(file_path_device1, file_format_device1)
hash_device2, df_device2 = read_and_verify_data(file_path_device2, file_format_device2)
```

Figure 3.17: Hash values comparison statement.

6. **Data Comparison (Optional):**
    - If the hash values match, suggesting that the data is potentially intact, the algorithm compares the actual data in the DataFrames (df_device1 and df_device2). This extra step ensures a more complete check of data integrity.

```
# Compare DataFrames to check data integrity
if hash_device1 and hash_device2:
    if df_device1.equals(df_device2):
        print("\nData integrity verified: Data from both devices is identical.")
```

Figure 3.18: Message when data is identical after data integrity check

```
else:
    print("Data integrity check failed: Data from devices does not match.")
```

Figure 3.19: Message when data integrity fails

7. **Displaying Results:**
    - The function returns the results of the data integrity check. If the hash values match and the data in the DataFrames is identical, a success message is presented. If there are any discrepancies, the algorithm locates and displays the exact rows where the data differs.

```
print("\nDisplaying the whole data (from Device 1):")
print(df_device1)
```

Figure 3.20: Output when data is identical

```
# Show the rows where data integrity fails
diff_rows = df_device1[df_device1.ne(df_device2).any(axis=1)]
print("\nRows where data integrity fails:")
print(diff_rows)
```

Figure 3.21: Output when data integrity fails

8. **User Interaction:**
    - The code communicates with the user throughout the process by asking file locations and formats, presenting results, and offering feedback in the event of mistakes or anomalies.

3. HASHING

In Information Security, hashing is the process of encoding the data entered in a hashvalue for information security purposes. A hash function is applied and used to produce a singular hash output of the input. Hash value also has constant size, which provides easiness in its storage and comparison. Hashing is one of the most amazing techniques, as it’s not possible to reverse engineer the initial input from the hash value output. This feature plays a vital role in protecting confidential details like passwords. The hash values generated are used for comparison to ensure integrity of data by preventing alteration in data after transmission or storage. Cryptographic hash functions like SHA-256 are most commonly applied in security applications such as digital signatures, password storage, and checksums. Hashing in information security is important because it helps in verification of data, authentication and protection against unauthorized changes.

- **Usage of Hashing in our project:**

  The SHA-256 hash function is the hashing method [Figure 3.19] used in our project for data integrity. The "calculate_hash" function creates a SHA-256 hash object and codes the hash values of the input dataset, which is changed to string datatype for better results, to guarantee compatibility with the hashing method. The encoded data is then sent through the hash object using the "update()" function, which results in a unique and fixed-length hash value of the input data. This encoding serves as the input dataset's unique digital signature, enabling data integrity verification throughout the process. The easiness of the SHA-256 method increases security by making it hard to reverse the operation and recover the original data from its hash value. This hashing method maintains the consistency of data sent between the IoT devices in our project.

Figure 3.22: How hashing works in our project

4. ENCRYPTION METHOD

We applied the RSA Algorithm to encrypt data transferred between two IoT devices. The RSA algorithm is a popular asymmetric data encryption technique used for secure data transmission. It is based on the computationlal problem of calculating huge values, making recovering the original data from its RSA encrypted version impossible. RSA creates a key pair, namely a public key for encryption and a private key for decryption. The public key is made by calculating modulus of exponent n and an exponent e, whereas the private key is made by calculating modulus of n and a an exponent d.

During encryption, the sender encrypts the data using the recipient's public key, and only the recipient with the compatible and authorized private key can decode and retrieve the original data. The difficulty of factoring the product of two large prime numbers makes RSA's security strong, providing a good approach for ensuring data integrity during communication.

- **Integration into the project:**

1. **Generating RSA Key Pairs:**

   Using the cryptography library, the "generate_key_pair" function generates a private-public key pair for each device.

```
def generate_key_pair():
    private_key = rsa.generate_private_key(
        public_exponent=65537,
        key_size=2048,
        backend=default_backend()
    )
    public_key = private_key.public_key()

    return private_key, public_key
```

Figure 3.23: Key pair Generation Function for RSA

2. **Encrypting Data:**

The "encrypt_data" function encrypts the DataFrame (converted to a string) with the other device's public key.

```
def encrypt_data(data, public_key):
    encrypted_data = public_key.encrypt(
        data.encode(),
        padding.OAEP(
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )
    return encrypted_data
```

Figure 3.24: Encryption function of RSA

3. **Decrypting Data:**

Using the private key, the "decrypt_data" function decrypts the received encrypted data.

```
def decrypt_data(encrypted_data, private_key):
    decrypted_data = private_key.decrypt(
        encrypted_data,
        padding.OAEP(
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )
    return decrypted_data.decode()
```

Figure 3.25: Decryption function of RSA

4. **Integration with Data Integrity Check:**

To verify data integrity, the encrypted data is decrypted and the decrypted data is compared across devices.

- **Steps in the Code:**
    1. The user enters file directories and data types for Devices 1 and 2.
    2. For both devices, RSA key pairs are created.
    3. Each dataset's data integrity is validated using SHA-256 hashing.
    4. Each device's data is encrypted using the public key of the other device.
    5. The relevant private keys are used to decode encrypted data.
    6. To check data integrity, decrypted data is compared with original data.
    7. Output shows whether the data integrity has been checked or not and displays original encrypted data from Device 1.

The use of the RSA algorithm ensures secure communication between IoT devices with protecting the integrity of data.

5. **Simulink access GUI:**

   The Simulink Access GUI includes controls for initiating, pausing, and halting simulations. Users may launch simulations and control their execution directly using the GUI interface. The GUI includes sliders, input boxes, and dropdown menus that allow users to interactively alter simulation settings. This enables the real-time adjustment of model parameters during simulation. Simulink Access GUI enables users to view simulation results in real time. It generates charts and displays to show signals, states, and other essential factors during simulation. Scopes or displays inside the GUI allow users to monitor certain signals or variables during simulation. This aids troubleshooting and understanding of system behavior. The Simulink Access GUI enables interactive testing and validation of Simulink models by allowing users to change inputs, see outputs, and evaluate the system's reaction in real time. The GUI may be adjusted to meet unique simulation requirements. Users can create unique controls, displays, or plots based on their application requirements.

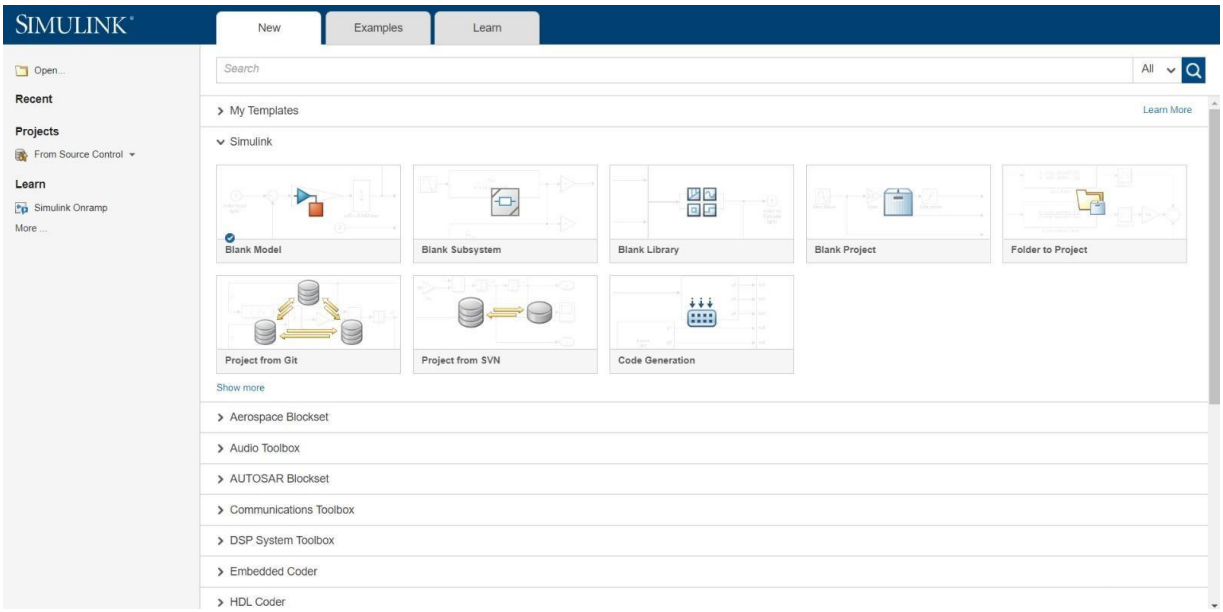

Figure 3.26: Simulink Access GUI

Usage:

1. **Start Simulation:**

   Click the "**Start**" button in the GUI to begin the simulation.

2. **Control Parameters:**

   Adjust simulation parameters using sliders, input boxes, or dropdown menus.

3. **Monitor Signals:**

   Use scopes or displays to monitor signals and variables during simulation.

4. **Stop Simulation:**

   Click the "**Stop**" button to halt the simulation at any time.

5. **Visualize Results:**

   View simulation results in real-time through plots and displays in the GUI.

```
% Load Simulink model
simulink_model = 'your_simulink_model'; % Specify the name of your Simulink model
open_system(simulink_model);

% Set simulation parameters in Simulink
set_param(simulink_model, 'StopTime', num2str(duration));
set_param(simulink_model, 'FixedStep', num2str(sampling_rate));

% Run simulation
simOut = sim(simulink_model);

% Access simulation data from Simulink output
sensor_data = simOut.sensor_data;
filtered_sensor_data = simOut.filtered_sensor_data;

% Print simulation parameters
fprintf('Simulation Parameters:\n');
fprintf('Duration: %d seconds\n', duration);
fprintf('Sampling Rate: %d second(s)\n', sampling_rate);
```

Figure 3.27: Boilerplate Simulink model program

```
% Define simulation parameters
duration = 3600; % Duration of simulation in seconds (1 hour)
sampling_rate = 1; % Sampling rate in seconds (1 sample per second)

% Define sensor names
sensor_names = {'Temperature Sensor', 'Humidity Sensor', 'Light Sensor'};
num_sensors = numel(sensor_names);

% Print simulation parameters
fprintf('Simulation Parameters:\n');
fprintf('Duration: %d seconds\n', duration);
fprintf('Sampling Rate: %d second(s)\n', sampling_rate);
fprintf('Number of Sensors: %d\n', num_sensors);

% Function to generate synthetic data for a sensor
generate_sensor_data = @(sensor_name) sprintf('%s: Data reading at time t', sensor_name);
```

Figure 3.28: Version 1 of the MATLAB program of our project

```c
/**
 * \file
 *         Testing the broadcast layer in Rime
 * \author
 *         Adam Dunkels <adam@sics.se>
 */

#include "contiki.h"
#include "net/rime.h"
#include "random.h"

#include "dev/button-sensor.h"

#include "dev/leds.h"

#include <stdio.h>
/*---------------------------------------------------------------------------*/
PROCESS(example_broadcast_process, "Broadcast example");
AUTOSTART_PROCESSES(&example_broadcast_process);
/*---------------------------------------------------------------------------*/
static void
broadcast_recv(struct broadcast_conn *c, const rimeaddr_t *from)
{
  printf("broadcast message received from %d.%d: '%s'\n",
         from->u8[0], from->u8[1], (char *)packetbuf_dataptr());
}
static const struct broadcast_callbacks broadcast_call = {broadcast_recv};
static struct broadcast_conn broadcast;
/*---------------------------------------------------------------------------*/
PROCESS_THREAD(example_broadcast_process, ev, data)
{
  static struct etimer et;

  PROCESS_EXITHANDLER(broadcast_close(&broadcast);)

  PROCESS_BEGIN();

  broadcast_open(&broadcast, 129, &broadcast_call);

  while(1) {

    /* Delay 2-4 seconds */
    etimer_set(&et, CLOCK_SECOND * 2 + random_rand() % (CLOCK_SECOND * 2));

    PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et));
```

Figure 3.29: Example-broadcast.c

This is the file used by 3 motes with a delay of 8-10 seconds in sending their signals with the message “Everything is fine”

```c
#include "contiki.h"
#include "net/rime.h"
#include "random.h"

#include "dev/button-sensor.h"

#include "dev/leds.h"

#include <stdio.h>
/*---------------------------------------------------------------------------*/
PROCESS(example_broadcast_process, "Broadcast example");
AUTOSTART_PROCESSES(&example_broadcast_process);
/*---------------------------------------------------------------------------*/
static void
broadcast_recv(struct broadcast_conn *c, const rimeaddr_t *from)
{
  printf("broadcast message received from %d.%d: '%s'\n",
         from->u8[0], from->u8[1], (char *)packetbuf_dataptr());
}
static const struct broadcast_callbacks broadcast_call = {broadcast_recv};
static struct broadcast_conn broadcast;
/*---------------------------------------------------------------------------*/
PROCESS_THREAD(example_broadcast_process, ev, data)
{
  static struct etimer et;

  PROCESS_EXITHANDLER(broadcast_close(&broadcast);)

  PROCESS_BEGIN();

  broadcast_open(&broadcast, 129, &broadcast_call);

  while(1) {

    /* Delay 2-4 seconds */
    etimer_set(&et, CLOCK_SECOND * 2 + random_rand() % (CLOCK_SECOND * 2));

    PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et));

    packetbuf_copyfrom("Hi im good wbu?", 16);
    broadcast_send(&broadcast);
    printf("broadcast message sent\n");
  }

  PROCESS_END();
}
```

This is the file used by 2 malicious motes with a delay of 2-4 seconds in sending their signals with the message “ATTACK”

Figure 3.30: Types of motes used in the simulation

Though we have many powerful motes, we have used the most lightweight mote (SKY) for the simulation

3.5 KEY CHALLENGES

The project: "Information Security for Shared IoT Infrastructures" applied latest technologies and approaches to enhancing the security of shared IoT devices. However, some issues occurred during the research and implementation of the project that demanded solutions to ensure the project's success. It was a huge barrier trying to shifting from standard security processes for normal data to the complexities and limitations of shared IoT ecosystems. To address this, the project emphasized the creation of an intuitive and strong algorithm to facilitate the rising issue of IoT devices.

Another challenge was maintaining the consistency of data during transmission. This was addressed by the use of RSA encryption technique to ensure safe data transfer and reduce risks. Moreover, the need for integrity checks inside shared IoT data presented accuracy challenges. This was solved by employing secure data integrity verification algorithm, which reduced the possibility of errors and increased overall system and data consistency.

In summary, the primary challenges in implementing "Information Security for Shared IoT Infrastructures" included secure communication protocols, automated integrity checks, device interaction verification, and security protocol customization. The project solved each challenge step by step in order to produce a reliable and effective information security solution designed for shared IoT ecosystems.

CHAPTER 4: TESTING

4.1 TESTING STRATEGY

The compatibility of the "Information Security for Shared IoT Infrastructures" project with security algorithms was crucial, making it necessary for thorough testing steps to test our applied algorithm. The testing comprised a detailed evaluation of all factors and constraints, as well as testing functional requirements to ensure that critical features such as secure data integrity and encryption worked properly. The program's responsiveness and scalability were also put to a test during performance testing to ensure smooth functioning. The testing technique and steps were designed to detect and correct any errors, ensuring that the project ran smoothly and reliably.

- Functional testing guaranteed that critical features such as secure communication, data integrity, and encryption worked properly.
- Security testing was critical for detecting and correcting any flaws in data processing and encryption methods.
- Performance testing assessed the system's responsiveness and scalability under shifting loads, assuring optimal operation.
- The iterative testing strategy intended to find and resolve any faults, ensuring that the project performed smoothly, securely, and reliably.

1. **Unit Testing:**
    - Unit testing was about analyzing individual parts of the program to verify their results.
    - Improved software quality, lower development and maintenance costs, and increased developer trust.
2. **Integration Testing:**
    - Integration testing evaluated software components that were integrated together, detecting faults that may occur as a result of their interaction.
    - This manual testing by testers verified that security procedures were integrated seamlessly.

- Reduced development and maintenance costs and improved trust in the integrated security features.

3. **System Testing:**
   - The complete integrated system was evaluated during system testing.
   - Faults that occurred when the system interacted with other systems or the environment were detected, ensuring that overall system requirements were satisfied.
   - Improved software quality, cheaper development and maintenance costs, and higher system confidence were among the benefits.
4. **Performance Testing:**
   - The system's responsiveness and scalability were tested under varied loads to ensure maximum efficiency.
   - System trust was increased, development and maintenance expenses were reduced, and software quality was enhanced.

All of the above testing and validation were carried out on a Jupyter Google Colab notebook including the necessary files and our Python script. So far, the code has performed admirably in all forms of testing.

4.2 TEST CASES AND OUTCOMES

1. Our initial test case was for data integrity in the.xlsx data format. By mounting Google Drive, two separate.xlsx files were grabbed and incorporated into the Jupyter Notebook.

   - **OUTCOME**
   - For both data integrity test situations, the algorithm produced the proper result.

2. The second test case was for data integrity in the.csv data format. By mounting Google Drive, two different files in.csv format were obtained and incorporated into the Jupyter Notebook.

   - **OUTCOME**
   - For both data integrity test situations, the algorithm produced the proper result.

3. The following test case concerned data integrity with several file format types. One file in.xlsx format and another in.csv format were used for this. When uploading, the algorithm was meant to inquire for the format of each file and was anticipated to produce accurate results even with cross-format data files.

   - **OUTCOME**
   - For both data integrity test situations, the algorithm produced the proper result. When the data was identical, the algorithm presented the entire dataset appropriately, and when the data integrity failed, it displayed the rows that differed in data integrity.

4. Our test case was putting the RSA Algorithm to the test for secure data transmission via keys.

For communication, both devices created their own public and private keys. The keys were then used to decode the data transferred between the devices.

- **OUTCOME**
- RSA method provided the proper result and correctly decrypted the message.

5. We test the Cooja Simulator for the Broadcast communication by the use of virtual IoT motes that we use as a replica of virtual IIoT devices.

- **OUTCOME**
- Simulation Provides Secure and energy efficient results in order to predict the secure transmission**.**

6. We test the MATLAB/Simulink for the secure transmission and transportation by the use of virtual IoT motes generally known as Scripts that we use as a replica of virtual IIoT devices having MATLAB codes inside it.

- **OUTCOME**
- This Simulation provides good idea as compared to Cooja Simulator as this is fast and used in more secure WNS protocols.

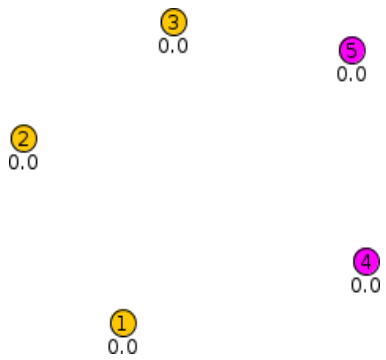

Figure 4.1: 5 motes and their positioning in the simulation environment

CHAPTER 5: RESULTS AND EVALUATION

5.1 RESULTS

1. Result when 2 xlsx format files were taken

```
Enter file path for data from Device 1: drive/MyDrive/Major project/Temp1.xlsx
Enter file format for data from Device 1 (csv or xlsx): xlsx
Enter file path for data from Device 2: drive/MyDrive/Major project/Temp1.xlsx
Enter file format for datFa from Device 2 (csv or xlsx): xlsx

Data integrity verified: Data from both devices is identical.

Displaying the whole data (from Device 1):
    Temp State
0     10    In
1     10   Out
2     10   Out
3     10    In
4     10   Out
5     25    In
6     60   Out
7     40   Out
8     32    In
9     32   Out
10     9    In
11    26   Out
12    18   Out
13    17    In
14     5   Out
15    28    In
16    56    In
17    40   Out
18    21   Out
```

Figure 5.1: Displaying data when both files of same format match

2. Result when 2 files of xlsx format dont match

```
Enter file path for data from Device 1: drive/MyDrive/Major project/Temp1.xlsx
Enter file format for data from Device 1 (csv or xlsx): xlsx
Enter file path for data from Device 2: drive/MyDrive/Major project/Temp2.xlsx
Enter file format for datFa from Device 2 (csv or xlsx): xlsx
Data integrity check failed: Data from devices does not match.

Rows where data integrity fails:
   Temp State
2    10   Out
4    10   Out
8    32    In
```

Figure 5.2: Displaying differences in data when both files of same format fail data integrity check

3. Result when 2 csv format files were taken

```
Enter file path for data from Device 1: drive/MyDrive/Major project/Temp1csv.csv
Enter file format for data from Device 1 (csv or xlsx): csv
Enter file path for data from Device 2: drive/MyDrive/Major project/Temp1csv.csv
Enter file format for datFa from Device 2 (csv or xlsx): csv

Data integrity verified: Data from both devices is identical.

Displaying the whole data (from Device 1):
    Temp State
0     10    In
1     10   Out
2     10   Out
3     10    In
4     10   Out
..   ...   ...
95    31   Out
96    51   Out
97    10    In
98    52   Out
99     1   Out

[100 rows x 2 columns]
```

Figure 5.3: Displaying data when both files of same format (csv) match

4. Result when 2 files of xlsx format dont match

```
Enter file path for data from Device 1: drive/MyDrive/Major project/Temp1csv.csv
Enter file format for data from Device 1 (csv or xlsx): csv
Enter file path for data from Device 2: drive/MyDrive/Major project/Temp2csv.csv
Enter file format for datFa from Device 2 (csv or xlsx): csv
Data integrity check failed: Data from devices does not match.

Rows where data integrity fails:
    Temp State
0     10    In
1     10   Out
8     32    In
13    17    In
20    23   Out
```

Figure 5.4: Displaying differences when both files of same format (csv) fail data integrity check

5. Results when two different file formats are checked

```
Enter file path for data from Device 1: drive/MyDrive/Major project/Temp1.xlsx
Enter file format for data from Device 1 (csv or xlsx): xlsx
Enter file path for data from Device 2: drive/MyDrive/Major project/Temp1shortcsv.csv
Enter file format for datFa from Device 2 (csv or xlsx): csv

Data integrity verified: Data from both devices is identical.

Displaying the whole data (from Device 1):
    Temp State
0     10    In
1     10   Out
2     10   Out
3     10    In
4     10   Out
5     25    In
6     60   Out
7     40   Out
8     32    In
9     32   Out
10     9    In
11    26   Out
12    18   Out
13    17    In
14     5   Out
15    28    In
16    56    In
17    40   Out
18    21   Out
```

Figure 5.5: Displaying data when 2 different file formats pass data integrity check.

6. When 2 different file formats fail data integrity check

```
Enter file path for data from Device 1: drive/MyDrive/Major project/Temp1.xlsx
Enter file format for data from Device 1 (csv or xlsx): xlsx
Enter file path for data from Device 2: drive/MyDrive/Major project/Temp1csv.csv
Enter file format for datFa from Device 2 (csv or xlsx): csv
Data integrity check failed: Data from devices does not match.

Rows where data integrity fails:
Empty DataFrame
Columns: [Temp, State]
Index: []
<ipython-input-12-c3c4fb1f507d>:54: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
  diff_rows = df_device1[df_device1.ne(df_device2).any(axis=1)]
```

Figure 5.6: When 2 different file formats fail data integrity check.

7. Data encryption and decryption check (RSA)

```
Enter the path of the data file (CSV or Excel): drive/MyDrive/Major project/Temp1shortcsv.csv
Encrypted Data:
b'\x88\x06\'&_\x83\xf4~\xf1HnF\xc38A?\x92\xda\xac\x0c\xdce9n\xc7\x1b\xb3\xa7O\xd8D~F\x1dT\xa2G\xd4

Decrypted Data:
    Temp State
0     10    In
1     10   Out
2     10   Out
3     10    In
4     10   Out
5     25    In
6     60   Out
7     40   Out
8     32    In
9     32   Out
10     9    In
11    26   Out
12    18   Out
13    17    In
14     5   Out
15    28    In
16    56    In
17    40   Out
18    21   Out
```

Figure 5.7: Results of the data file shared being encrypted and decrypted and matching the data.

8. Cooja Simulator Instance making and creating

```
Applications  Places
File  Edit  View  Search  Terminal  Help
     [java]     at java.awt.EventQueue$4.run(EventQueue.java:706)
     [java]     at java.security.AccessController.doPrivileged(Native Method)
     [java]     at java.security.ProtectionDomain$1.doIntersectionPrivilege(Prot
ectionDomain.java:76)
     [java]     at java.awt.EventQueue.dispatchEvent(EventQueue.java:705)
     [java]     at java.awt.EventDispatchThread.pumpOneEventForFilters(EventDisp
atchThread.java:242)
     [java]     at java.awt.EventDispatchThread.pumpEventsForFilter(EventDispatc
hThread.java:161)
     [java]     at java.awt.EventDispatchThread.pumpEventsForHierarchy(EventDisp
atchThread.java:150)
     [java]     at java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.j
ava:146)
     [java]     at java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.j
ava:138)
     [java]     at java.awt.EventDispatchThread.run(EventDispatchThread.java:91)
     [java]  INFO [Thread-7] (Simulation.java:252) - Simulation main loop starte
d, system time: 1710698111450
     [java]  INFO [Thread-7] (Simulation.java:311) - Simulation main loop stoppe
d, system time: 1710698163482   Duration: 52032 ms      Simulated time 339286 ms
Ratio 6.520718019680197
```

Figure 5.8: Results of the data file shared being encrypted and decrypted and matching the data.

9. Cooja Simulation obtained Output

```
00:27.661   ID:1   broadcast message received from 3.0: 'Everything is fine'
00:27.750   ID:4   broadcast message received from 3.0: 'Everything is fine'
00:27.765   ID:2   broadcast message received from 3.0: 'Everything is fine'
00:28.941   ID:5   broadcast message sent
```

Figure 5.9: The output of the IoT devices is displayed while working normally

10. Attack Detection Output

```
00:29.014   ID:2   broadcast message received from 5.0: 'ATTACK'
00:30.059   ID:4   broadcast message sent
00:30.159   ID:1   broadcast message received from 4.0: 'ATTACK'
00:30.177   ID:3   broadcast message received from 4.0: 'ATTACK'
00:32.317   ID:4   broadcast message sent
00:32.387   ID:5   broadcast message sent
00:32.514   ID:2   broadcast message received from 5.0: 'ATTACK'
00:34.864   ID:4   broadcast message sent
00:34.909   ID:1   broadcast message received from 4.0: 'ATTACK'
00:34.926   ID:3   broadcast message received from 4.0: 'ATTACK'
00:36.371   ID:5   broadcast message sent
00:36.389   ID:2   broadcast message received from 5.0: 'ATTACK'
00:36.918   ID:4   broadcast message sent
```

Figure 5.10: The output when the malicious motes start to ATTACK

11. DOS Attack Output

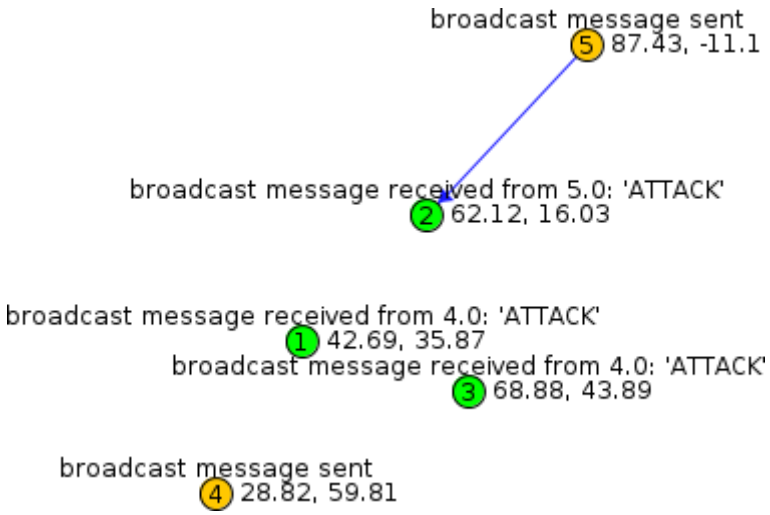

Figure 5.11: Malicious motes trying to perform DOS attack

12. Synthetic data generation and time-stamp communication graph of 3 sensors

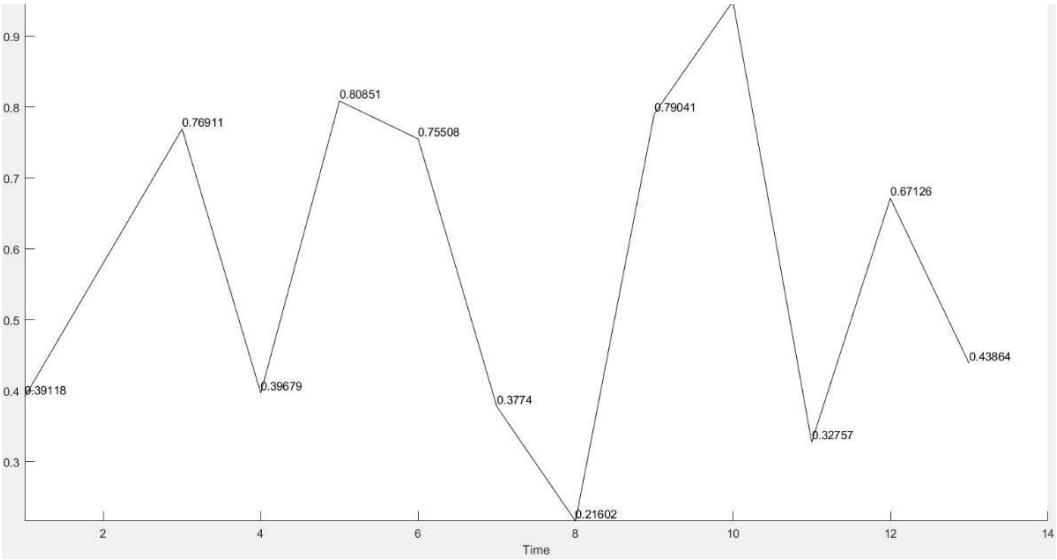

Figure 5.12: Communication graph with time-stamp and values shared

13. Replica of Industrial IoT Sensors

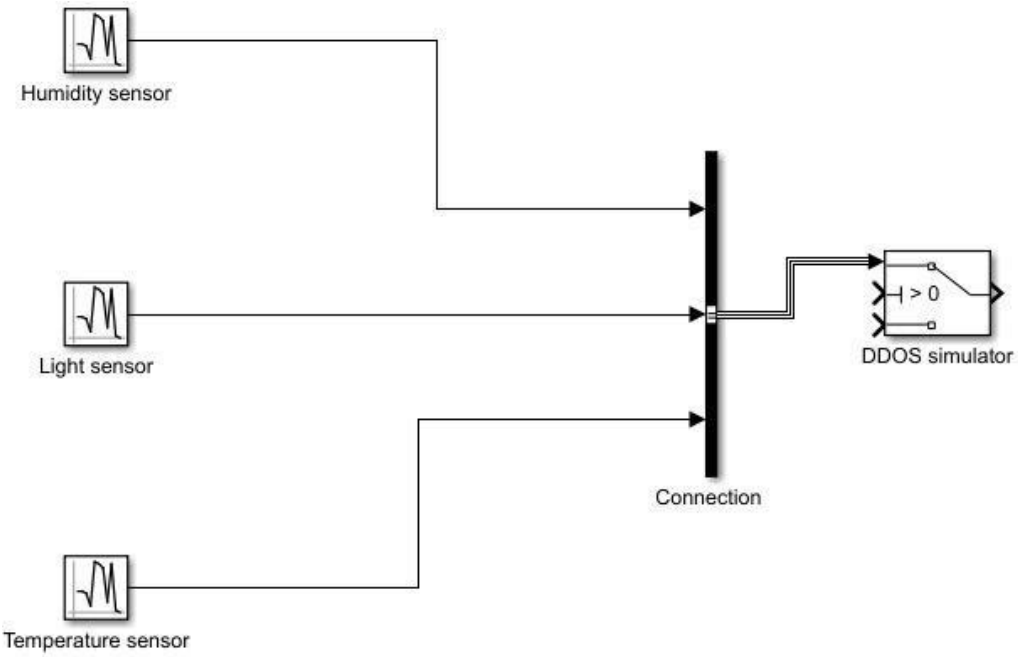

Figure 5.13: First model and structure of our Industrial IOT using 3 different sensors

CHAPTER 6: CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

In conclusion, the investigation of information security inside shared IoT infrastructures has given major insights and paved the way for comprehensive security solutions. The use of the RSA algorithm, data integrity checks and the use of Cooja Simulator and MATLAB with Simulink has helped to ensure the secrecy and integrity of data sent between networked devices.

6.1.1 Key Findings:

This project's use of modern technologies and algorithms yields crucial discoveries that show significant gains over standard methodologies in the field of information security for shared IoT infrastructures.

1. **RSA Algorithm Advantages:**
    - *Enhanced Encryption Strength:* When compared to older approaches, the use of the RSA algorithm gives substantially better encryption. Its dependence on mathematical difficulties and the usage of public and private keys improves data secrecy, outperforming traditional encryption solutions.
    - *Secure Key Exchange:* The efficiency of the RSA algorithm in secure key exchange overcomes a significant flaw in older techniques. In contrast to symmetric key techniques, RSA enabled secure communication channels without the need for a shared secret, reducing the dangers associated with key distribution.

2. **Data Integrity Checks:**
    - *Tamper-Resistant Data:* Traditional data integrity measures frequently fail to identify and prevent manipulation. The project's use of data integrity checks, such as hashing, assures tamper-resistant data. This outperforms existing checksums and CRC approaches, resulting in a more robust mechanism for

ensuring the integrity of shared data in IoT situations.

3. **Planned Integration of Network Technology:**
    - *Scalability and Flexibility:* The intended incorporation of network technologies will provide even more scalability and flexibility than traditional on-premise systems. Contiki OS systems having COOJA simulation strategies allow dynamic resource allocation, effortlessly adapting the shifting demands of shared IoT infrastructures, which is difficult for conventional infrastructures to do.
    - *Policy Enforcement:* Centralized security management guarantees that all IIoT devices follow the security regulations established by the enterprise. Policies may include access control restrictions, data encryption standards, and communication methods. These regulations may be designed and simulated using MATLAB and Simulink, and then enforced across all devices by a centralized system.

4. **MATLAB with Simulink Advantages:**
    - *Simulation Capabilities:* Simulink provides sophisticated simulation features, enabling us to simulate complicated systems such as communication networks and control algorithms. Simulink allows us to replicate the behavior of virtual IIoT devices and examine their performance under different situations.

Finally, the major findings show that the technologies and methods used in this research surpass existing approaches in critical areas such as encryption strength, data integrity, scalability, access control, and real-time threat detection. These advances contribute to a more robust and adaptable information security framework for shared IoT infrastructures, overcoming shortcomings in traditional security techniques.

6.1.2 Limitations:

While the RSA algorithm and data integrity checks are excellent advances toward improving security, certain limits must be acknowledged. In latency-sensitive IoT applications, the computational complexity associated with RSA encryption may have an influence on real-time responsiveness. Furthermore, the efficacy of security measures is contingent on the widespread adoption and compliance of all devices inside the shared infrastructure.

6.1.3 Contributions to the Field:

By tackling the special issues of shared infrastructures, this initiative makes a substantial contribution to the field of IoT security. Cloud technologies, access control, and an NIDS are among the planned integrations that will provide flexibility to new security threats.

Furthermore, the contributions of this project go beyond individual security components, encompassing a collaborative, adaptive, and multi-layered security approach. The initiative contributes to the continuing discussion of information security for shared IoT infrastructures by harmonizing with ideas from other research articles. The planned integrations and adaptive security measures demonstrate a forward-thinking strategy, contributing to IoT security framework robustness and sustainability in the face of increasing threats.

6.2 FUTURE SCOPE

The intended integration of cloud technologies, access and authorization management, and an NIDS opens up interesting possibilities for future research and development. The cloud will improve scalability, while access control measures will improve device authorization and authentication. The NIDS will aid in proactive threat detection, strengthening the security architecture.

Finally, the combination of existing security mechanisms and future integrations places this project at the forefront of assuring data security in shared IoT infrastructures. These results and recommended upgrades provide a solid basis for preserving the integrity and confidentiality of shared data across networked devices as the IoT environment grows. Addressing the highlighted restrictions and moving forward with the planned integrations will be critical for progressing the field and maintaining a safe and collaborative IoT ecosystem.

Expanding the scope of our project on "Information Security in Shared IoT Infrastructures" can involve incorporating additional technologies to address emerging challenges and

enhance overall security. Here are some technologies that we plan to incorporate in future:

1. **Blockchain Technology:**
    - Investigate the use of blockchain for safe and transparent transaction records as well as decentralized identity management. Blockchain technology can improve the integrity and reliability of data transferred between IoT devices.

2. **Machine Learning for Anomaly Detection:**
    - Use machine learning methods to spot anomalies in IoT data streams. ML models may learn typical behavior patterns and recognize abnormalities, assisting in the early detection of threats.
3. **Homomorphic Encryption:**
    - Investigate homomorphic encryption approaches for computing on encrypted data. This permits data to stay encrypted while being processed, adding an extra degree of secrecy.
4. **Biometric Authentication for IoT Devices:**
    - Integrate biometric authentication techniques to boost device security. Biometrics such as fingerprint or iris scanning can help to improve access control methods.

By investigating these technologies in the future, we will be able to strengthen the security posture of shared IoT infrastructures while staying ahead of growing threats and technical breakthroughs.

LIST OF PUBLICATIONS

2023 Seventh International Conference on Image Information Processing (ICIIP)

Paper ID: 1570974184

Paper Title: Identification of Phishing Attacks Using Machine Learning

Track Name: Digital Image Forensics & Security

REFERENCES

[1] Kim, E., Huh, K., & Kang, J. (2023). Role-Based Security Models in Shared IoT Infrastructures. IEEE Access, 11, 16581-16593.

[2] Garcia, D., Rodrigues, J. J., & Caldeira, A. (2022). Privacy-Preserving Techniques for Shared IoT Data: A Survey and Analysis. IEEE Internet of Things Journal, 9(17), 12344-12359.

[3] Patel, A., Borcea, C., & Gluhak, A. (2021). Security Framework for Shared IoT Environments: A Comparative Study. In 2021 IEEE International Conference on Smart Grid and Smart Cities (SGSC) (pp. 739-744). IEEE.

[4] Wang, C., Zhang, J., Wang, J., & Shen, J. (2020). Blockchain-Based Authentication in Shared IoT Networks. IEEE Transactions on Computational Social Systems, 7(1), 215-229.

[5] Gupta, H., Liu, A., & Maharjan, S. (2020). Secure Communication Protocols for Shared IoT Applications: A Survey and Comparison. IEEE Internet of Things Journal, 7(12), 16092-16112.

[6] Singh, D., Kwak, R., & Rehmani, M. H. (2020). Identifying and Analyzing Potential Vulnerabilities in Smart City Infrastructures: A Systematic Literature Review. IEEE Internet of Things Journal, 7(2), 1135-1147.

[7] Rodriguez, T., Lopez, J., & Roman, R. (2018). Developing Encryption Methods for Shared IoT Data: Balancing Security and Efficiency. IEEE Transactions on Cloud Computing, 6(4), 1023-1031.

[8] Tadayoni, R., Mahmud, R., & Alawadhi, S. (2017). Understanding the Diverse Range of Threats Faced by IoT Ecosystems: A Review. IEEE Communications Surveys & Tutorials, 19(1), 413-435.

[9] Dorri, A., Kanjorski, B., & Jurdak, R. (2018). Blockchain technology for the internet of things: An overview. IEEE Communications Surveys & Tutorials, 20(3), 1617-1649.

[10] Ekdahl, O., & Jonsson, F. (2008). The Internet of Things: A strategic overview. Cisco Systems White Paper.

[11] Evans, D. (2011). The Internet of Things: How it works and how you can design things that matter. O'Reilly Media.

[12] Fett, B., Gupta, V., & Badhiwala, A. (2014). A survey on security issues and challenges in the Internet of Things. In IEEE Internet of Things Journal (Vol. 1, No. 2, pp. 169-179). IEEE.

[13] Gupta, V., & Vasconcells, M. (2013). Security in the internet of things: A taxonomy of attacks and countermeasures. In Computer Networks (Vol. 57, No. 2, pp. 210-234). Elsevier.

[14] Heath, S., & Nikov, A. (Eds.). (2008). Internet of Things: Technology and applications. CRC press.

[15] Hirschfeld, R. (2013). The Internet of Things: An introduction. CRC press.

[16] Hwang, T., & Kim, D. (2016). A survey of fingerprint identification techniques for mobile devices. IEEE Communications Surveys & Tutorials, 18(1), 1-22.

[17] IoT Security for Shared IoT Infrastructures (Ed. U. Taşdemir et al.)

[18] IoT Security: A Comprehensive Guide (Ed. D. C. Sharma et al.)

[19] IoT Security: Privacy, Authentication, and Trust (Eds. J. Gubbi, R. Buyya, S. Marusic, & F. Montyne)

[20] IoT Security: Challenges, Solutions, and Future Directions (Ed. S. N. Bhatti)

[21] Kettler, M. (Ed.). (2014). The Internet of Things: A vision for 2020. Cisco Systems White Paper.

[22] Mohit, H. (2021). Security and privacy in the Internet of Things (IoT). In Handbook of Wireless Networks and Communication Systems (pp. 1343-1362). Springer, Singapore**.**

[23] Nikov, A., & Heath, S. (Eds.). (2010). The Internet of Things: Emerging research directions. CRC press.

[24] Senge, M. (2011). The fifth discipline: Fieldbook. Doubleday Books.

[25] Stankovic, J. A., & Stankovic, J. A. (2014). Research directions in wireless sensor networks. Ad Hoc Networks, 13, 1-1.

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: …….

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: _____ **Department:** _____ **Enrolment No** _____

Contact No. _____ **E-mail.** _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

_____

_____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ..... (%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

<table>
<tr><td>Copy Received on</td><td rowspan="4">• All Preliminary Pages
• Bibliography/Images/Quotes
• 14 Words String</td><td rowspan="2">Similarity Index (%)</td><td colspan="2">Word Counts</td></tr>
<tr><td rowspan="2">Report Generated on</td><td colspan="2">Character Counts</td></tr>
<tr><td rowspan="2">Submission ID</td><td colspan="2">Total Pages Scanned</td></tr>
<tr><td></td><td colspan="2">File Size</td></tr>
</table>

Checked by
Name & Signature **Librarian**

……

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

201517

ORIGINALITY REPORT

3%	3%	0%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	fastercapital.com Internet Source	1%
2	eprints.cs.univie.ac.at Internet Source	<1%
3	i-scholar.in Internet Source	<1%
4	codeberg.org Internet Source	<1%
5	Submitted to University of Wollongong Student Paper	<1%
6	www.ir.juit.ac.in:8080 Internet Source	<1%
7	w-rdb.waseda.jp Internet Source	<1%
8	Submitted to University of Sydney Student Paper	<1%
9	www.cryptopolitan.com Internet Source	<1%

10	bjnbooks.com Internet Source	<1 %
11	github.com Internet Source	<1 %
12	5dok.org Internet Source	<1 %
13	research.monash.edu Internet Source	<1 %
14	www.juit.ac.in Internet Source	<1 %
15	Submitted to Nottingham Trent University Student Paper	<1 %
16	archive.org Internet Source	<1 %
17	dspace.nm-aist.ac.tz Internet Source	<1 %
18	export.arxiv.org Internet Source	<1 %
19	www.coursehero.com Internet Source	<1 %
20	www.frontiersin.org Internet Source	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		