ISSN 2277-2685
IJESR/Jan-Mar. 2026/ Vol-16/Issue-1/421-428
Suraj Kumar Pal *et. al.*, /International Journal of Engineering & Science Research

Full Length Article

Blockchain-Enabled Comprehensive Security Framework For Industrial Iot

Gande Lavakumar[1],Suraj Kumar Pal[2],Mohammad Azharuddin[3],V Prajval[4]
[1]Assistant Professor ; Department Of Information Technology, Guru Nanak Institutions Technical Campus, Hyderabad, India.
[2,3,4]B.Tech Students; Department Of Information Technology, Guru Nanak Institutions Technical Campus, Hyderabad, India.
Mail Id; Suraj8101007646@gmail.com
Accepted 25-03-2026

Abstract:
The Industrial Internet of Things (IIoT) is rapidly transforming modern industries by enhancing operational efficiency and productivity. However, the exponential growth and diversity of connected devices expose IIoT systems to significant security vulnerabilities. Traditional centralized security mechanisms struggle with scalability, data integrity, and reliability, leaving industrial environments susceptible to cyberattacks. To address these challenges, this research proposes a decentralized, blockchain-based security framework for IIoT systems. By integrating blockchain technology, smart contracts, and SHA-256 encryption, the framework provides robust authentication, authorization, and data integrity without reliance on a central authority. Smart contracts dynamically enforce access policies and respond in real time to suspicious activities, while SHA-256 ensures strong data confidentiality. Moreover, IoT devices are leveraged to manage encryption keys and data access, enabling a lightweight and scalable security solution tailored to dynamic industrial environments. The hybrid use of public and private blockchains ensures both high performance and confidentiality, offering a practical and adaptive approach to securing IIoT infrastructures. Experimental results demonstrate the proposed framework's effectiveness in enhancing data security, scalability, and operational reliability.

Keywords: *Industrial Internet of Things (IIoT), Blockchain Technology, Cybersecurity, Decentralized Security Framework, Smart Contracts, SHA-256 Encryption, Data Integrity, Authentication and Authorization, Secure Data Transmission, IoT Security, Public and Private Blockchain, Access Control, Encryption Key Management, Industrial Automation Security, Scalable Security Solutions.*

INTRODUCTION

The Industrial Internet of Things (IIoT) refers to the integration of interconnected smart devices within industrial environments to enhance efficiency, automation, and data-driven decision-making. These devices—comprising sensors, actuators, and controllers—collect operational data and share it across the network. This data is subsequently analyzed to optimize processes, detect anomalies, and enable predictive maintenance. IIoT represents the industrial application of Internet of Things (IoT) technologies, transforming traditional linear manufacturing into dynamic, interconnected systems.

Industries such as manufacturing, energy, agriculture, oil and gas, logistics, and healthcare benefit from IIoT by enabling real-time monitoring, process optimization, and predictive decision-making. For instance, in manufacturing, IIoT devices track production lines to prevent downtime, reduce defective output, and maintain quality standards. In agriculture, sensors monitor soil, weather, and crop conditions to enable precision farming, enhancing yields while optimizing resource usage. Energy systems leverage IIoT for predictive maintenance and performance optimization in renewable and non-renewable power plants. Oil and gas operations utilize IIoT for remote pipeline monitoring, drilling management, and maintenance prediction, improving safety and operational efficiency. In transportation and logistics, IIoT facilitates fleet management, route optimization, and cargo monitoring to ensure timely delivery and cost reduction. Even in healthcare, IIoT improves patient monitoring, hospital equipment management, and pharmaceutical process tracking, supporting both safety and efficiency.

A key advantage of IIoT is its ability to transform reactive industrial operations into proactive and predictive systems. By enabling data-driven decision-making, IIoT improves productivity, safety, and innovation across industries. However, as industrial networks expand, the proliferation of devices introduces significant security challenges. Each device functions as a node in a large network, requiring secure communication to prevent

unauthorized access or data manipulation. Scalability, data confidentiality, and integrity become critical concerns, particularly as centralized security solutions struggle to manage thousands of interconnected devices efficiently. Without robust security measures, industrial networks remain vulnerable to cyberattacks, operational disruptions, and financial losses.

Scope of the Project

This project aims to develop a secure, efficient, and scalable IIoT framework that addresses both operational optimization and cybersecurity challenges across industrial domains. The framework will enable seamless communication among smart devices, facilitating real-time data collection, analysis, and exchange. It focuses on ensuring confidentiality, integrity, and interoperability while allowing the system to scale to thousands of nodes without compromising performance. By integrating advanced data analytics, predictive maintenance, and secure communication protocols, the project seeks to minimize downtime, optimize industrial processes, and reduce operational costs. Ultimately, it aims to create a robust and secure environment that supports the digital transformation of industrial operations.

Objectives

The primary objectives of this project are:

- To design a scalable and intelligent IIoT framework that enables efficient industrial automation and real-time decision-making.
- To ensure secure data transmission, authentication, and access control among interconnected smart devices.
- To implement security mechanisms that guarantee confidentiality, integrity, and protection against cyber threats.
- To address scalability challenges, maintaining performance as the number of connected devices increases.
- To facilitate predictive maintenance and process optimization, reducing downtime and operational costs.
- To improve overall reliability, safety, and productivity across industrial environments including manufacturing, energy, agriculture, oil and gas, and logistics.

Problem Statement

While IIoT enhances industrial productivity and operational efficiency, it introduces significant security challenges due to the large, heterogeneous networks of connected devices and the sensitive data they generate. Traditional centralized security solutions fail to scale effectively, preserve data integrity, or adapt to diverse protocols. Centralized architectures present single points of failure, leaving systems vulnerable to cyberattacks, data breaches, and operational disruptions. Therefore, there is a critical need for a scalable, decentralized, and secure framework that ensures real-time data protection, interoperability, and resilience against emerging threats.

PROJECT DESCRIPTION

This project aims to develop a secure communication and data protection framework for the Industrial Internet of Things (IIoT), with a particular focus on email-based data exchange. The objective is to ensure that all industrial information transmitted via email remains confidential, authenticated, and tamper-proof. In modern industrial environments, sensitive data—such as operational reports, production analytics, and maintenance logs—is frequently shared over digital channels, making cybersecurity and privacy paramount.

Traditional email systems and centralized security solutions are often vulnerable to cyberattacks, unauthorized access, and data manipulation. To address these challenges, the proposed system integrates robust cryptographic mechanisms, including encryption, digital signatures, and secure key management, to protect data both in transit and at rest. The framework ensures that only authorized recipients can decrypt and access transmitted information while maintaining data integrity. Additionally, it supports real-time authentication of users and messages, establishing trust in industrial communications.

Unlike conventional IIoT solutions that primarily focus on physical sensors, this approach targets the cyber layer of industrial data sharing. It offers a lightweight, scalable, and easily deployable security solution that minimizes computational overhead, making it suitable for resource-constrained industrial environments. Ultimately, the project provides a secure and efficient IIoT communication framework, safeguarding critical industrial data against modern cyber threats and enabling reliable digital collaboration across organizations.

Methodologies

Input and Expected Output

Module	Input	Expected Output
User Interface	Select module from dashboard	Redirect to login or upload pages; display project overview and navigation
Upload Data	Manager uploads file	File encrypted, decryption key generated, metadata stored, upload logged

Module	Input	Expected Output
Blockchain Layer	Encrypted data with ID/key	Stores data in decentralized ledger; ensures immutability and transparency
SHA-256	Encrypted file	Generates hashes for data integrity; validates blocks
Smart Contracts	Client searches file	Validates request, forwards for approval, releases decryption key securely
JSP Dashboard	Login credentials/search queries	Displays encrypted data, approved file requests, blockchain status
MySQL Database	Login details, system logs	Maintains non-critical info; supports system management

2.3 Techniques and Algorithms

SHA-256

SHA-256 secures data integrity by generating a unique 256-bit hash for each file and block. Block hashes are linked, forming a chain similar to a blockchain. Any modification triggers a hash mismatch, ensuring tamper detection. During file access, hashes validate integrity before decryption, guaranteeing authenticity and traceability.

Smart Contracts

Smart contracts automate file management and access control. They govern block creation, hashing, access requests, and key distribution. Requests are validated and approved without manual intervention, creating a transparent, tamper-resistant system. By eliminating intermediaries, smart contracts enhance security, accountability, and efficiency in file transactions.

REQUIREMENTS ENGINEERING

The proposed Blockchain-enabled Industrial IoT security framework relies on smart contracts and blockchain management to ensure secure, transparent, and tamper-proof data handling. The system integrates modules such as Manager, Blockchain Network, SHA-256, Smart Contracts, JSP Dashboard, and MySQL Database. Together, these components provide functionalities including real-time encryption, blockchain-based verification, fine-grained access control, and immutable data storage. The framework is designed to deliver efficient, scalable, and policy-driven secure data management in industrial environments.

3.1 Hardware Requirements

To develop and deploy the system efficiently, the following hardware specifications are recommended:

Minimum Requirements:

- **Processor:** Intel Core i3 or higher
- **RAM:** 4 GB DDR4 or higher
- **Monitor:** 15.6" LED display
- **Storage:** Minimum 100 GB
- **Peripherals:** Keyboard, Mouse, Webcam (for QR/facial features)

These specifications ensure smooth development, testing, and operation of the system under realistic industrial scenarios.

3.2 Software Requirements

The software environment defines the platform and tools required for development, testing, and deployment.

- **Front-End:** Java EE (JSP, Servlets)
- **Back-End:** MySQL 5.5 or higher
- **Operating System:** Windows 10 / 11
- **Web Server:** Apache Tomcat 7.0
- **Browser Support:** Google Chrome, Mozilla Firefox
- **IDE:** Eclipse IDE or IntelliJ IDEA

This environment supports seamless interaction between system modules and ensures compatibility across typical industrial setups.

3.4 Non-Functional Requirements

Security

- SHA-256 hashing ensures immutability and integrity for all file blocks.
- Smart contracts manage approvals and key distribution, preventing unauthorized access.
- All module communication occurs over secure HTTPS/TLS channels.

Performance

- Supports multiple simultaneous uploads and encryptions without degradation.
- Maintains minimal response times for file search and retrieval.
- Block creation and hash computation are optimized for real-time operation.

Usability

- Provides a user-friendly, consistent interface for Managers, Admins, and Clients.
- File search, request, and approval workflows are intuitive.
- Feedback and status messages (upload success, approval, download readiness) are clearly displayed.

Scalability

- System supports additional users, blocks, and ledgers without performance impact.

- Smart contracts and blockchain storage scale efficiently with increasing data volume.
- Architecture allows integration of future enhancements with minimal restructuring.

Reliability

- Blockchain ensures consistent, fault-tolerant storage even in node failures.
- All transactions are permanently recorded and verifiable through hashes.
- Automated integrity checks restore and validate corrupted data using stored hash values.

DESIGN ENGINEERING

Design engineering serves as a blueprint for visualizing, specifying, constructing, and documenting software systems. In this project, the **Blockchain-Enabled Comprehensive Security Framework for Industrial IoT** leverages design engineering to represent interactions between system actors, modules, and data flows while ensuring secure collection, encryption, storage, and visualization of industrial data.

The system uses **UML diagrams** to model behavior, structure, and workflows:

- **Use Case Diagrams:** Capture interactions between actors (Manager, Admin, Client) and system modules, including file uploads, encryption, storage, validation, and dashboard visualization.
- **Class Diagrams:** Depict classes such as File, Block Ledger, Smart Contract, EncryptionModule, Dashboard, and Database and their relationships, illustrating how encrypted data flows from upload to retrieval.
- **Sequence Diagrams:** Show stepwise communication between modules; for example, file upload → SHA-256 encryption → smart contract validation → blockchain storage → dashboard retrieval.
- **Activity Diagrams:** Represent workflows including data collection, encryption, blockchain recording, policy enforcement via smart contracts, and dashboard visualization.
- **Component Diagrams:** Demonstrate dependencies and interfaces among components like Manager Upload, SHA-256, Smart Contracts, Blockchain Layer, JSP Dashboard, and MySQL Database.
- **Deployment Diagrams:** Illustrate deployment of software components across servers and clients, including blockchain nodes, smart contract servers, and database servers.

UML diagrams serve as essential tools guiding developers, testers, and stakeholders throughout the software development life cycle, ensuring secure and functional design while supporting efficient data handling.

4.1.1 Use Case Diagram

The Use Case Diagram illustrates functional interactions between system actors and modules.

- **Manager:** Uploads files, automatically encrypted with SHA-256 and divided into multiple blocks linked by cryptographic hashes to form a secure ledger.
- **Admin:** Oversees system integrity by verifying and approving decryption or access requests.
- **Client:** Searches files and sends access requests.
- **JSP Dashboard:** Displays uploaded data, key statuses, and facilitates key generation and secure download.
- **MySQL Database:** Maintains user metadata and non-critical information.

This diagram captures secure, policy-driven workflows between actors and system modules.

4.1.2 Class Diagram

The class diagram represents the structural design of the system.

- **Manager:** Handles file uploads.
- **File:** Represents each uploaded file, encrypted using SHA-256 and divided into blocks.
- **Block Ledger:** Stores each block with current and previous hash and timestamps.
- **Client:** Searches files and generates decryption requests.
- **Admin:** Validates and approves access requests.
- **Smart Contract:** Automates blockchain transactions and enforces access control policies.
- **Request Handler:** Processes client requests in coordination with smart contracts.

Relationships among these classes ensure smooth, secure, and traceable data flow.

4.1.3 Object Diagram

The object diagram shows runtime instances of classes such as Manager1, Client1, Admin1, File, and Block1–BlockN.

- A file uploaded by Manager1 is divided into multiple Block objects.
- Each block stores a SHA-256-generated current and previous hash.
- Client1 generates a request to access a file, which is approved by Admin1.
- The Smart Contract automatically releases the decryption key, maintaining security, trust, and transparency.

4.1.4 State Chart Diagram

The state chart diagram captures the lifecycle of a file within the system:

1. **File Uploaded** → SHA-256 encryption →
2. **File Encrypted** → Divided into blocks →
3. **Block Added to Ledger** → Client sends access request →
4. **Request Pending** → Admin/Manager Approval →
5. **Decryption Key Issued** → Client downloads file →
6. **File Accessed / Completed**

This diagram ensures data confidentiality, authorization, and blockchain integrity throughout the process.

4.1.5 Sequence Diagram

The sequence diagram illustrates interactions for file upload and retrieval:

1. Manager uploads file → encrypted via SHA-256.
2. File divided into blocks → stored in blockchain via Smart Contract.
3. Client searches for file → sends access request.
4. Admin and Manager approve → Smart Contract releases decryption key.
5. Client downloads original file.

This diagram visualizes control flow and secure communication among all entities.

4.1.6 Collaboration Diagram

The collaboration diagram highlights dynamic interactions:

- Manager uploads files → triggers Smart Contract to divide and store blocks.
- Each block references the previous block to ensure integrity.
- Client searches and requests decryption → Smart Contract validates hash chain.
- Admin and Manager approve → decryption key delivered securely.

It emphasizes communication, responsibilities, and delegation among modules.

4.1.7 Activity Diagram

The activity diagram depicts workflow for secure file management:

- File upload → block division → SHA-256 hashing → blockchain storage.
- Client search → integrity validation → request approval → key release or rejection.

Decision nodes represent conditions such as approval/rejection, showing dynamic interaction across modules.

4.1.8 Component Diagram

The component diagram represents the high-level system structure:

- **Manager Module:** Uploads and divides files, triggers smart contract functions.
- **Admin Module:** Approves access requests, monitors blockchain ledger.
- **Client Module:** Searches files, requests decryption.
- **Smart Contract Engine:** Central logic controller for access rules, verification, and key distribution.
- **Block Ledger:** Stores file blocks and transaction records.

It emphasizes modularity and clear responsibilities.

4.1.9 Entity-Relationship (E-R) Diagram

The E-R diagram models data entities and relationships:

- **Entities:** Manager, Admin, Client, Uploaded File, File Block, Decryption Request.
- Manager uploads files → stored in Uploaded File → divided into multiple File Blocks with SHA-256 hashes.
- Client creates Decryption Requests → Admin approves/rejects → triggers smart contract actions.
- Relationships enforce one-to-many mappings (Manager → Files, File → Blocks, Client → Requests).

This ensures secure, traceable, and efficient database design.

4.1.10 Data Flow Diagram (DFD)

The DFD demonstrates how data moves across modules:

- Manager uploads file → divided into blocks → SHA-256 hash generated → stored in blockchain.
- Smart Contract enforces rules and validates requests.
- Client searches and requests decryption → Admin/Manager approval → decryption key delivered.

It emphasizes secure, traceable information flow and clearly defines data stores, inputs, and outputs.

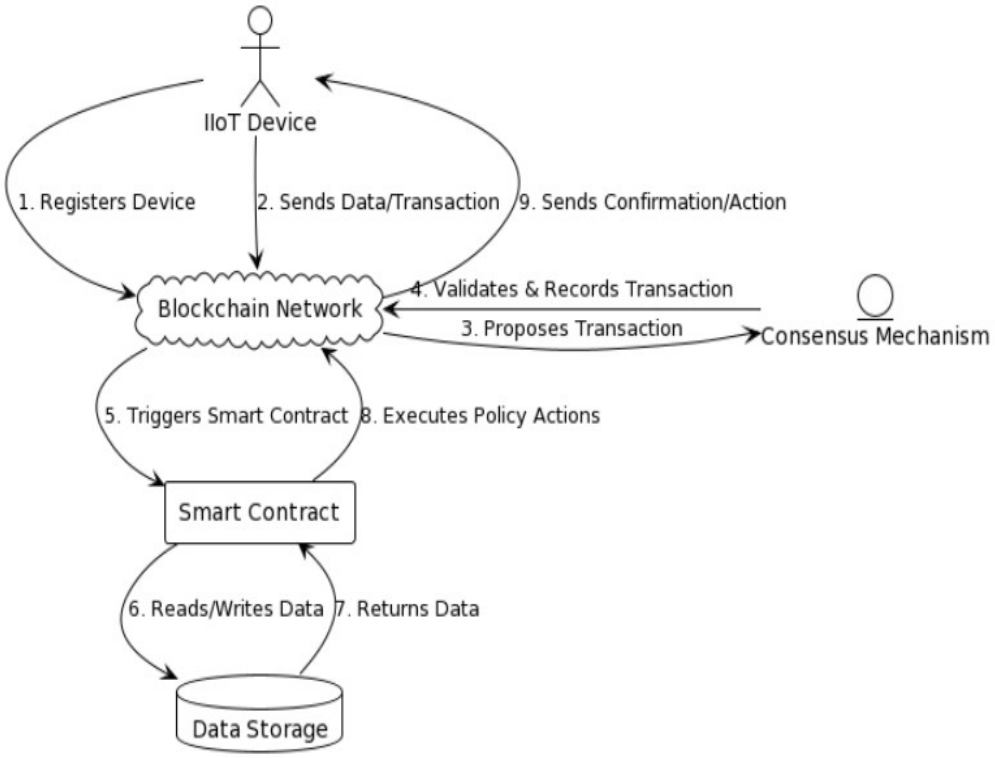
4.1.11 Deployment Diagram

The deployment diagram illustrates component distribution:

- **Client Device:** Interface for searching files and requesting decryption.
- **Manager/Server Node:** Uploads files, divides blocks, interacts with Smart Contract Engine.
- **Admin Node:** Monitors requests and approves/rejects decryption requests.
- **Blockchain Ledger Node:** Stores immutable blocks and transaction records.

It highlights interactions, dependencies, and secure deployment across nodes.

4.2 System Architecture



The system architecture presents a secure, decentralized, and automated framework for file management:

- **Modules:** Manager, Admin, Client, Smart Contract & Block Ledger.
- **Manager:** Uploads files, divided into blocks linked via SHA-256 hashes.
- **Admin:** Approves/rejects decryption requests.
- **Client:** Searches and requests files.
- **Smart Contract Engine:** Automates integrity checks, access control, and key distribution.
- **Block Ledger:** Securely stores all file blocks and transactions.

Data flows are encrypted and governed by smart contracts, ensuring authorized operations only. The system supports scalability, modularity, and real-time secure interactions. Web-based interfaces allow seamless monitoring, logging, and management while preserving integrity, transparency, and accountability across all operations.

DEVELOPMENT TOOLS

This chapter discusses the software languages and tools employed in the development of the project. The primary platform chosen is **Java**, specifically using **J2EE** for implementation. Java provides a versatile, platform-independent environment suitable for building enterprise-level applications with robust security, scalability, and database interaction capabilities.

5.2 Features of Java

5.2.1 Java Framework

Java, originally developed by **James Gosling** at Sun Microsystems in 1995, is an object-oriented, class-based programming language designed to be platform-independent. Its syntax is influenced by C and C++, but it simplifies memory management and low-level operations. Java applications are compiled into bytecode, executable on any **Java Virtual Machine (JVM)**, enabling the principle of *write once, run anywhere*.

Java is widely used for desktop applications, web development, mobile applications, and large-scale distributed systems. Its strengths include versatility, efficiency, platform portability, and robust security, making it ideal for network computing, from personal devices to cloud-based infrastructures.

5.2.2 Objectives of Java

Java's extensive developer community, exceeding 6.5 million members worldwide, has refined and validated the language over decades. Key benefits include:

- Developing software on one platform and running it across multiple environments.
- Creating applications compatible with web browsers and web services.
- Developing server-side applications for e-commerce, forums, and HTML form processing.

- Integrating applications and services to build highly customized solutions.
- Building efficient applications for mobile devices, embedded systems, and digital appliances.

Java learning resources are widely available through online tutorials, virtual courses, and university programs.

5.2.3 Object-Oriented Features

Java is a fully object-oriented language, characterized by:

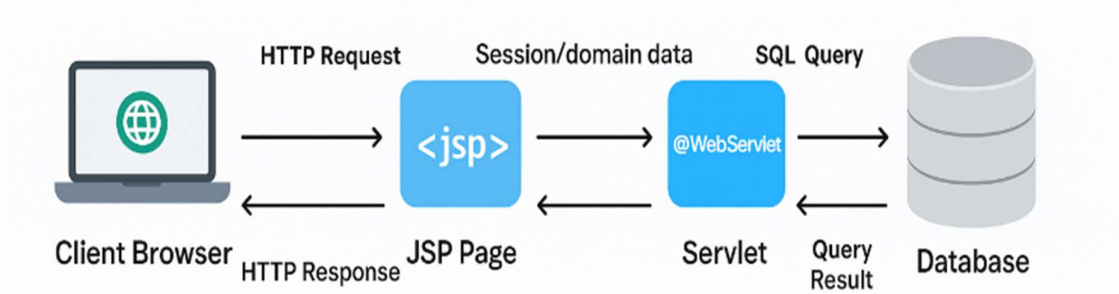
1. **Inheritance:** Enables reuse of existing code by extending classes.
2. **Encapsulation:** Combines data and methods, providing abstraction.
3. **Polymorphism:** Allows methods with the same name to perform different functions based on input parameters.
4. **Dynamic Binding:** Determines object types at runtime, enhancing flexibility and functionality.

5.3 Advanced Java Technologies

Advanced Java extends core Java to support web-based, enterprise, and distributed applications. Key components include:

- **Servlets:** Server-side programs handling HTTP requests and responses. Lifecycle methods include init(), service(), and destroy().
- **JavaServer Pages (JSP):** Embeds Java code in HTML for dynamic web content, compiled into servlets at runtime.
- **Java Database Connectivity (JDBC):** Standard API for database interaction, enabling SQL execution, connection management, and result processing.
- **JavaBeans:** Reusable components encapsulating multiple objects.
- **MVC Architecture:** Separates business logic (Model), user interface (View), and request handling (Controller).
- **Filters & Listeners:** Pre- and post-processing of requests, and event-driven responses within web applications.
- **Session Management:** Maintains user state across multiple pages using cookies, HttpSession, or URL rewriting.

Apache Tomcat is used as the servlet container, supporting Servlets, JSP, and HTTP protocols. It provides a lightweight, easy-to-deploy environment suitable for enterprise and web applications.



SOFTWARE TESTING

Software testing is a systematic process aimed at identifying errors, faults, and weaknesses in a system. It ensures that software components, sub-assemblies, and the complete system meet the specified requirements and user expectations, and that they operate reliably without unacceptable failures. Testing encompasses multiple approaches, each addressing different verification and validation objectives.

8.2 Testing Methodology

A structured testing methodology begins with a comprehensive plan that evaluates the software's functionality across different platforms and configurations. Rigorous quality control procedures are applied to ensure the application aligns with the system requirements and remains free of critical defects. The methodology involves planning, executing, and validating tests to identify and correct errors efficiently.

8.3 Types of Tests

8.3.1 Unit Testing

Unit testing validates individual software components to ensure that their internal logic produces the correct output for given inputs. Each decision branch, function, and code path is verified to ensure accurate operation. Unit tests are performed after completing a component but before system integration, providing early detection of defects at the component level.

8.3.2 Functional Testing

Functional testing confirms that the system performs as intended according to business and technical requirements. It evaluates:

- **Valid Inputs:** Proper handling of acceptable inputs.
- **Invalid Inputs:** Correct rejection of inappropriate inputs.
- **Functions:** Verification of all system functions.
- **Outputs:** Assessment of expected output classes.
- **System Procedures:** Proper interfacing with other systems or processes.

8.3.3 System Testing

System testing assesses the integrated software as a whole to verify that it meets the requirements. This

includes configuration testing and integration testing to ensure predictable and consistent outcomes.

CONCLUSION

The integration of blockchain and lightweight cryptography in **Industrial Internet of Things (IIoT)** systems demonstrates substantial improvements in security, efficiency, and scalability. Lightweight cryptographic algorithms reduce encryption and decryption times, decreasing energy consumption while maintaining strong protection of confidentiality and data integrity. Smart contracts provide dynamic, trust-based access control, enhancing security measures for both devices and data sources.

Scalability assessments indicate that blockchain-enabled IIoT systems can accommodate increasing numbers of devices without significant performance degradation, while energy efficiency is maintained through lightweight cryptography. Although simulations confirm the feasibility and benefits, real-world deployments may present additional challenges requiring further optimization. Overall, the study highlights the potential of combining blockchain and lightweight cryptography to secure and optimize IIoT systems effectively.

References

1. O. Peter, A. Pradhan, and C. Mbohwa, "Industrial Internet of Things (IIoT): Opportunities, challenges, and requirements in manufacturing businesses in emerging economies," *Proc. Comput. Sci.*, vol. 217, pp. 856–865, Apr. 2023.
2. R. Kumar and N. Agrawal, "Analysis of multi-dimensional industrial IoT (IIoT) data in edge–fog–cloud based architectural frameworks: A survey on current state and research challenges," *J. Ind. Inf. Integr.*, vol. 35, Oct. 2023, Art. no. 100504.
3. H. Alasmary, "RDAF-IIoT: Reliable device-access framework for the industrial Internet of Things," *Mathematics*, vol. 11, no. 12, p. 2710, Jun. 2023.
4. V.-T. Truong, D.-B. Ha, A. Nayyar, M. Bilal, and D. Kwak, "Performance analysis and optimization of multiple IIoT devices radio frequency energy harvesting NOMA mobile edge computing networks," *Alexandria Eng. J.*, vol. 79, pp. 1–20, Sep. 2023.
5. J. Singh, I. S. Ahuja, H. Singh, and A. Singh, "Application of quality 4.0 (Q4.0) and industrial Internet of Things (IIoT) in agricultural manufacturing industry," *AgriEngineering*, vol. 5, no. 1, pp. 537–565, Mar. 2023.
6. B. Babayigit and M. Abubaker, "Industrial Internet of Things: A review of improvements over traditional SCADA systems for industrial automation," *IEEE Syst. J.*, vol. 18, no. 1, pp. 120–133, Mar. 2024.
7. O. T. Sanchez, D. Raposo, A. Rodrigues, F. Boavida, R. Marculescu, K. Chen, and J. Sá Silva, "An IIoT-based approach to the integrated management of machinery in the construction industry," *IEEE Access*, vol. 11, pp. 6331–6350, 2023.
8. D. Berestov, O. Kurchenko, L. Zubyk, S. Kulibaba, and N. Mazur, "Assessment of weather risks for agriculture using big data and industrial Internet of Things technologies," in *Proc. Cybersecur. Providing Inf. Telecommun. Syst.*, 2023, pp. 1–13.
9. J. V. Arputharaj and S. K. Pal, "Transforming Industry 5.0: Real-time monitoring and decision making with IIoT," in *Sustainability in Industry 5.0*, Boca Raton, FL, USA: CRC Press, 2024, pp. 76–106. [Online]. Available: https://www.taylorfrancis.com/chapters/edit/10.1201/9781032686363-5
10. M. Kumar, G. K. Walia, H. Shingare, S. Singh, and S. S. Gill, "AI-based sustainable and intelligent offloading framework for IIoT in collaborative cloud-fog environments," *IEEE Trans. Consum. Electron.*, vol. 70, no. 1, pp. 1414–1422, Feb. 2024.
11. D. Kumar, P. Pawar, H. Gonaygunta, and S. Singh, "Impact of federated learning on industrial IoT—A review," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 13, no. 1, pp. 1–12, Dec. 2023.
12. K. S. Hawaou, V. C. Kamla, S. Yassa, O. Romain, J. E. N. Mboula, and L. Bitjoka, "Industry 4.0 and industrial workflow scheduling: A survey," *J. Ind. Inf. Integr.*, vol. 38, Mar. 2024, Art. no. 100546.
13. H. Alshahrani, A. Khan, M. Rizwan, M. S. A. Reshan, A. Sulaiman, and A. Shaikh, "Intrusion detection framework for industrial Internet of Things using software-defined network," *Sustainability*, vol. 15, no. 11, p. 9001, Jun. 2023.
14. A. Sasikumar, S. Vairavasundaram, K. Kotecha, V. Indragandhi, L. Ravi, G. Selvachandran, and A. Abraham, "Blockchain-based trust mechanism for digital twin empowered industrial Internet of Things," *Future Gener. Comput. Syst.*, vol. 141, pp. 16–27, Apr. 2023.
15. A. Aljuhani, P. Kumar, R. Alanazi, T. Albalawi, O. Taouali, A. K. M. N. Islam, N. Kumar, and M. Alazab, "A deep learning integrated blockchain framework for securing industrial IoT," *IEEE Internet Things J.*, vol. 11, no. 5, pp. 7817–7827, Mar. 2024.
16. A. Sasikumar, L. Ravi, M. Devarajan, A. Selvalakshmi, A. T. Almaktoom, A. S. Almazyad, G. Xiong, and A. W. Mohamed, "Blockchain-assisted hierarchical attribute-based encryption scheme for secure information sharing in industrial Internet of Things," *IEEE Access*, vol. 12, pp. 12586–12601, 2024.
17. Y. Guo, Y. Guo, P. Xiong, F. Yang, and C. Zhang, "A provably secure and practical end-to-end authentication scheme for tactile industrial Internet of Things," *Pervas. Mobile Comput.*, vol. 98, Feb. 2024, Art. no. 101877.