# CIA-Triad-Oriented Security Architecture for Mission Critical IoT Environments: A Bank Monitoring Case Study

**Arnold Ochieng'**

**150211**

**A Project Proposal Submitted to School of Computing Engineering Sciences, Strathmore University for the Award of Bachelor of Science in Computer Networks and Cyber Security**

**STRATHMORE UNIVERSITY**

**NAIROBI, KENYA**

**July 2025**

## Declaration and Approval

I declare that this work has not been submitted or previously submitted and approved, in whole or partial, for the award of a degree by this or any other University. To the best of my knowledge and belief, the report contains no material previously published or written by another person except where due reference is made in the dissertation itself.

**Student**

Arnold Ochieng'

150211

Signature ……. Date …….

**Approval**

Mr. Harrison Martini Talo
Lecturer Computer Networks and Telecommunications

School of Computing and Engineering Sciences

Signature ……. Date 14/07/2025…….

## Acknowledgement

I wish to express my gratitude to Strathmore University for the opportunity and provision of necessary materials through the Makerspace Lab to undertake this project. Special thanks to my supervisor, Mr. Harrison Martini Talo, whose insights have been instrumental in the execution and completion of the project, as well as the project coordinator, Dr. Vitalis Ozianyi, whose unwavering support and guidance were indispensable for the duration of the project.

# Abstract

IoT technologies are widely adopted in different sectors including industrial automation, healthcare monitoring, home security, smart city evolution and transportation among others. The CIA triad also features in these environments, to secure the data that flows in and out of these sectors and that it is kept private to only authorized users, the data is not tampered with and that it is available for use when needed. However, because of their limited power, memory, and computational resources, many IoT deployments lack strong security solutions, making it difficult to support conventional security protocols. This affects the confidentiality of transmitted data, integrity of system functions, and continuous availability of services, especially under potential attacks or failures. In high-stakes environments such as banking, even minor breaches can lead to severe financial and operational consequences. This project seeks to tackle the gap by designing a secure, resilient, and scalable architecture that leverages the CIA triad to safeguard IoT driven smart environments. Some of the technologies to be adopted in this project include LoRaWAN for securely sending data from end devices to the database, Firebase for data storage, The Things Stack for node management, MQTT as the data protocol, AES 128-bit and end-to-end encryption for confidentiality, OTPs for authentication and hashing functions for integrity as well as GSM connectivity for availability as a secondary network. Through the practical lens of a bank monitoring system, the project will showcase how a carefully engineered combination of sensors, microcontrollers, and secure communication protocols can uphold the security principles essential to any IoT deployment. The project will be done using ESP32 and Arduino MKR GSM 1400 microcontrollers, with software written in C++ for the embedded firmware and will feature sensors such as the DHT11, ultrasonic sensors, LDRs and buzzers among others. Using Kotlin and Android Studio, a mobile application shall be created for system interaction and testing the real-time transmission thus verifying the real-time transmission, accuracy, and secure access of sensor data to demonstrate how CIA principles can be pragmatically enforced in IoT systems. For the project methodology, it will involve delving into various literature on IoT technologies, networks and security mechanisms employes in IoT deployments, assessing the challenges and vulnerabilities as well as the existing solutions to IoT security and their relevant gaps. To develop and implement the solution, prototyping methodology focusing on evolutionary prototyping will be used since prototyping speeds up the development life cycle and act as milestones showing progress and relevant advancements in the project. The evolutionary method supports reuse of previous prototypes thus saving on time and allows refinement throughout the development phase.



Keywords: *IoT security, CIA triad, mission-critical infrastructures, data integrity, secure communication protocols.*

# Table of Contents

# List of Figures

# List of Tables

## List of Abbreviations and Acronyms

AES – Advanced Encryption Standard

API – Application Programming Interface

ATM – Automated Teller Machines

AuC – Authentication Center

BSC – Base Station Controller

BSS – Base Station Subsystem

BTS – Base Transreceiver Station

CA – Certificate Authority

CCTV – Closed Circuit Television

CIA – Confidentiality, Integrity, Availability

CRC – Cyclic Redundancy Check

DoS – Denial of Service

DDoS – Distributed Denial of Service

ECC – Elliptic Curve Cryptography

EIC – Equipment Identification Register

GE – Gate Equivalencies

GPIO – General Purpose Input/Output

GSM – Global System for Mobile Communications

GNU – GNU's Not Unix

HLR – Home Location Register

IoT – Internet of Things

IDE – Integrated Development Environment

IMEI – International Mobile Equipment Identity

IT – Information Technology

JSON – JavaScript Object Notation

JVM – Java Virtual Machine

LDR – Light Dependent Resistor

LWCHF – Lightweight Cryptographic Hash Functions

LoRaWAN – Long Range Wide Area Network

LPWAN – Low-Power Wide Area Network

MIC – Message Integrity Code

MiTM – Man in the Middle

MQTT – Message Queuing Telemetry Transport

MSC – Mobile Switching Center

NSS – Network Switching Subsystem

OMC – Operation and Maintenance Center

OOP – Object Oriented Programming

OS – Operating System

OSS – Operation Subsystem

OTAA – Over the Air Activation

OTP – One Time Pin

PKI – Public Key Infrastructure

RSA – Rivest Shamir Adleman

RSS – Radio Switching Subsystem

SDK – Software Development Kit

SMS – Short Messaging Service

SQL – Structured Query Language

TDMA – Time Division Multiple Access

TLS/SSL – Transport Layer Security/Secure Sockets Layer

TTS – The Things Stack

UI/UX – User Interface/User Experience

USB – Universal Serial Bus

UTF-8 – Unicode Transformation Format 8-bit

VLR – Visiting Location Register

XOR – Exclusive-OR

## Definition of Terms

Advanced Encryption Standard – Symmetric encryption technique for ciphering and deciphering data.

GSM – a digital cellular network standard used for mobile devices like phones and modems. It's a widely adopted 2G technology that digitizes and transmits voice and data signals across a network.

IoT - a network of physical nodes including wearables, hospital and domestic appliances, vehicles among other devices fitted with sensory devices, network capabilities and embedded firmware to facilitate sharing of data and communication.

LoRaWAN - a protocol for communication based on LoRa technology, which allows nodes to send data over extended distances with little battery exhaustion. For LoRa-based devices, it functions essentially as a network layer, facilitating scalable and effective wireless communication for Internet of Things applications.

MQTT - a low-bandwidth, high-latency, or unreliable messaging protocol that is mostly utilized for IoT deployments that require communication. It uses a publish-subscribe architecture in which interested subscribers can obtain data from nodes and send it to a broker.

# Chapter 1: Introduction

## 1.1 Background to the project

The Internet of Things (IoT) has revolutionized the aggregation, sharing and utilization of data across multiple domains by allowing physical devices to access public networks, perceive their surroundings and carry out automated tasks. As per the views by Nižetić et al. (2020), "The rise of IoT technologies is currently intense and according to projections for the next 10 years, over $125x10^9$ IoT devices are expected to be connected." IoT technologies have proliferated several sectors; from home appliances to industrial automation and smart cities, with some of these technologies being embedded in mission-critical environments. Mission-critical IoT applications are defined as time-critical applications; the kinds of applications whose potential intermission would result not only in potential risk to life but also the disruption of public services, cause a disturbance to public order, imperil operations of organizations as well as cause businesses to suffer relevant losses (Farooq & Zhu, 2021). Delays in exchange of information in such applications fail the first-hand objective of the application. Mission-critical environments require constant up time, uncompromised system behaviour and trusted access control. For instance, a wearable wristband may be used to track a patient's health condition and the failure of the IoT device could end in potential loss of life. In monitoring systems where IoT devices are used to provide physical security such as surveillance cameras, a disruption in their operation could lead to a possible break in, significant financial losses and compromise customer trust in contexts such as banks.

The CIA triad embodies the central element for information security, and it comprises of Confidentiality, Integrity and Availability. It ensures data security in connected networks, protection of sensitive information, accuracy and accessibility only to legitimate users (W. N. Abidde et al., 2025). These principles are mostly employed in traditional IT systems but are not as widely enforced in IoT systems due to their computational resources, power limitations and lightweight communication protocols.

The project concentrates on the development of a secure IoT-based monitoring system fashioned for a mission-critical banking environment. It adopts the principles outlined in the CIA triad to guide the implementation of confidentiality through end-to-end encryption and OTP-based access, integrity via digital hashing mechanisms and availability by supporting the use of LoRa as the

main network for data transfer and redundant transmission via GSM network. Together with microcontrollers (ESP32 and Arduino MKR GSM 1400) and a mobile application for real-time communication, the monitoring system will include several sensors, including PIR for motion detection, ultrasonic sensors for intrusion proximity, LDR for light-based tamper detection, and DHT for environmental monitoring. By concentrating on a banking use case, the study will demonstrate how an architecture driven by the CIA triad may improve IoT environment resilience and encourage safe deployments in more general mission-critical domains.

## 1.2 Problem Statement

Security remains a huge affair even if IoT technologies are being adopted quickly in many important industries. The limited memory and processing capacity of IoT devices are primarily used in driving automation, monitoring, and efficiency enhancement in smart environments, but they also restrict the application of security regulations. Because of this, mission-critical settings that depend on IoT systems are more vulnerable to operational disturbances, illegal access, and cyberthreats. Some security risks and weaknesses coupled with IoT deployments include data leakage, eavesdropping, hacking and DoS (Karie et al., 2021). Reactive measures like perimeter-based firewalls and simple user authentication are the main emphasis of current systems. End-to-end security throughout the data lifecycle, including collection, transport, processing, and storage is frequently not achieved by these methods. This project seeks to meet the security gap by designing a safe, resilient, and scalable architecture that leverages the CIA triad to safeguard IoT systems. Using a bank as a test case, it demonstrates how sensors and intelligent systems can be used for monitoring to provide efficient access to data and provide physical security, as well as how a CIA-compliant design can fortify IoT systems to ensure secure and reliable operations in mission-critical settings.

## 1.3 Objectives of the Study

### 1.3.1 General Objective

The main objective of this project is to design and develop a secure IoT-based framework for mission-critical environments using the CIA triad as a guiding framework and a bank monitoring system as an illustrative example.

### 1.3.2 Specific Objectives

i. To investigate the integration of various IoT technologies and networks in IoT applications as well as their vulnerabilities and security challenges.

ii. To analyze existing solutions, tools and technologies employed in enforcing security in IoT applications.

iii. To design and develop a bank monitoring system using IoT components and an Android mobile application to concretize a real-world mission critical IoT smart environment.

iv. To implement security principles aligned with the CIA triad within the developed system and test their effectiveness in ensuring confidentiality, integrity and availability of data.

## 1.4 Scope and Limitations

### 1.4.1 Scope

The study focuses on designing and implementing a CIA-triad-oriented security framework tailor-made for the deployment of IoT capabilities in mission-critical environments as well as the development of a bank monitoring system as a case study. Through the use of mechanisms such as end-to-end encryption of data, AES 128-bit encryption, hashing, OTP-based access control, and LoRaWAN and GSM networks, the project highlights the evolution of a secure IoT architecture that upholds the CIA principles. The project incorporates both hardware and software components to conceptualize a secure IoT deployment with real-time monitoring and alerting system. The solution prioritizes lightweight, scalable, and cost-effective methods to meet the constraints of typical IoT devices. The mobile application will be instrumental in providing secure access to sensor data and alerts, while the backend infrastructure ensures reliable and secure data transmission and storage, with inclusion of LoRaWAN as the primary network and GSM for availability of data.

Other solutions to security in IoT smart environments include creating dedicated VLANs in enterprise networks to segregate IoT-enabled nodes from other devices on the LAN, implementing security at the network's edge where security policies are applied at the gateway before transmission for storage or use of firewalls and intrusion detection systems and AI driven threat analysis and mitigation to secure these IoT devices and applications. However, this project focuses on implementable, resource-efficient solutions best suited to the constrained nature of IoT

hardware, to enforce the security policies on the devices themselves to avoid compromise of data at the network level while aligning with the practical challenges and limitations of IoT environments.

### 1.4.2 Limitations

The first limitation to the project is the reliance on the campus LoRaWAN gateway. The gateway might suffer significant downtimes which will in turn hinder the use of the LoRaWAN network via the gateway during such times. Furthermore, the free tier limits of the Firebase Realtime database might hinder data storage if they are exceeded due to increasing traffic and the deployment of functions and triggers on the database.

## 1.5 Significance of the project

As IoT technologies continue to mushroom in both consumer and commercial applications, the security of these technologies becomes increasingly pivotal; especially in mission-critical sectors like industrial control systems or security monitoring systems. Despite the key benefits coupled with the innovation in IoT, many of their deployments remain vulnerable due to the devices' constrained resources such as computational power which deters the implementation of conventional security protocols. This poses a great risk in applications where even a minor breach can wind up in severe financial losses, disruption of key services and compromised data integrity.

The relevance of the project stems from its capability to demonstrate how low-cost, real time and secure IoT systems can be developed and deployed without sacrificing fundamental security principles for efficiency and performance. Furthermore, the case study in the context of a bank actualizes the relevance of the solution in protecting a sensitive infrastructure while securing the IoT deployment itself, which highlights the project's relevant contribution towards the security of IoT ecosystems.

# Chapter 2: Literature Review

## 2.1 Introduction

This chapter delves into the main technologies utilized in IoT ecosystems along with their applicability in mission-critical settings. It also looks at how limited resources and insufficient security measures lead to IoT vulnerabilities and challenges that are specific to IoT networks and devices. Additionally, it talks about existing solutions and outlines their strengths and shortcomings in terms of protecting IoT applications. The chapter winds up with a conceptual framework that will inform the design and development of the proposed CIA-triad-based security architecture.

## 2.2 Technology Review

This section explores the key technologies that propel system performance and operational efficiency of IoT applications in mission-critical environments. Each of these technologies, which include hardware, networking protocols, software platforms and data security frameworks, perform essential functions in guaranteeing functionality, connectivity and security of the overall system. This part thus establishes the technological basis upon which the proposed solution will be designed and developed.

### 2.2.1 Microcontrollers and IoT Hardware

A microcontroller is an all-in-one chip which integrates a microprocessor and other components such as timing units, filters, signal converters, memory, among other auxiliary components (Márquez-Vera et al., 2023). They come in a variety of types depending on the architecture e.g. 8-bit, the memory architecture, instruction set, and number of pins. Figure 2.1 indicates the general architecture of microcontrollers.

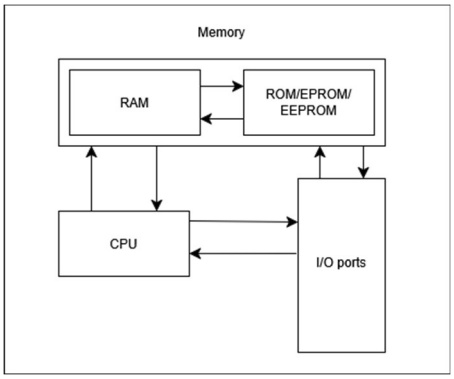



Figure 2.1: General block diagram of microcontrollers

As per the categorization by Birlog et al. (2020), the building blocks of IoT hardware are divided into the asset being managed, the data acquisition module which comprises of sensors used to catch signals and their conversion to digital format, the data processing module which processes and executes operations on data and saves it; but some devices lack this capability and thus transmit data upstream to entities such as cloud applications and finally the communication module which facilitates communication with third party components and cloud specific platforms.

### 2.2.2 Communication Technologies

#### A. LoRa and LoRaWAN

For IoT networks, LPWAN is the desired network choice for networks in IoT chosen for its extended coverage, budget-friendliness and efficient energy use (Jouhari et al., 2023). LoRa has emerged as the preferred LPWAN option because it is easy and cost efficient to deploy, it has off-site maintenance needs as well as worldwide availability which aids deployment of private networks without third-party operators. LoRa functions within unlicensed frequencies, including 433 MHz, 868MHz and 915MHz as per the regional regulations. Figure 2.2 illustrates the typical layout of a LoRa frame. It consists of the preamble consisting of four fixed symbols and other variable symbols, an optional header, payload data and the value for the payload Cyclic Redundancy Check (CRC).

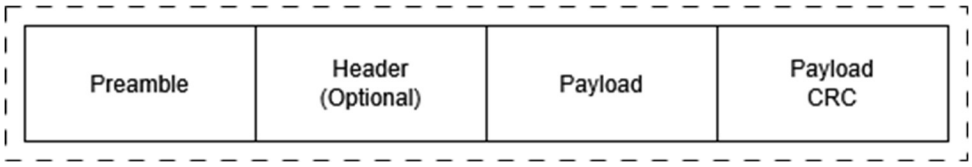



Figure 2.2: Structure of a LoRa frame

As explained by Almuhaya et al. (2022), LoRaWAN is "a cloud-based MAC layer networking protocol that is designed and maintained by the LoRa Alliance to define the upper layers of long-range wide-area networks with a LoRa physical layer. It is an LPWAN technology for battery-operated wireless connection of objects to the Internet in regional, national, or worldwide networks that aims at major IoT needs, including two-way communication, end-to-end protection, mobility, and localization." It is founded on the LoRa physical layer and is used to control flow of traffic between gateways and end nodes on the network layer. Its features include low power optimization of end devices, broad coverage, indoor penetration, unlicensed frequency bands, location tracking capabilities, large capacity, private and public implementations, AES-128 bit encryption, remote

firmware updates, roaming, low cost and certification programs (LoRaWAN, 2025). Figure 2.3 shows the architecture of LoRaWAN.

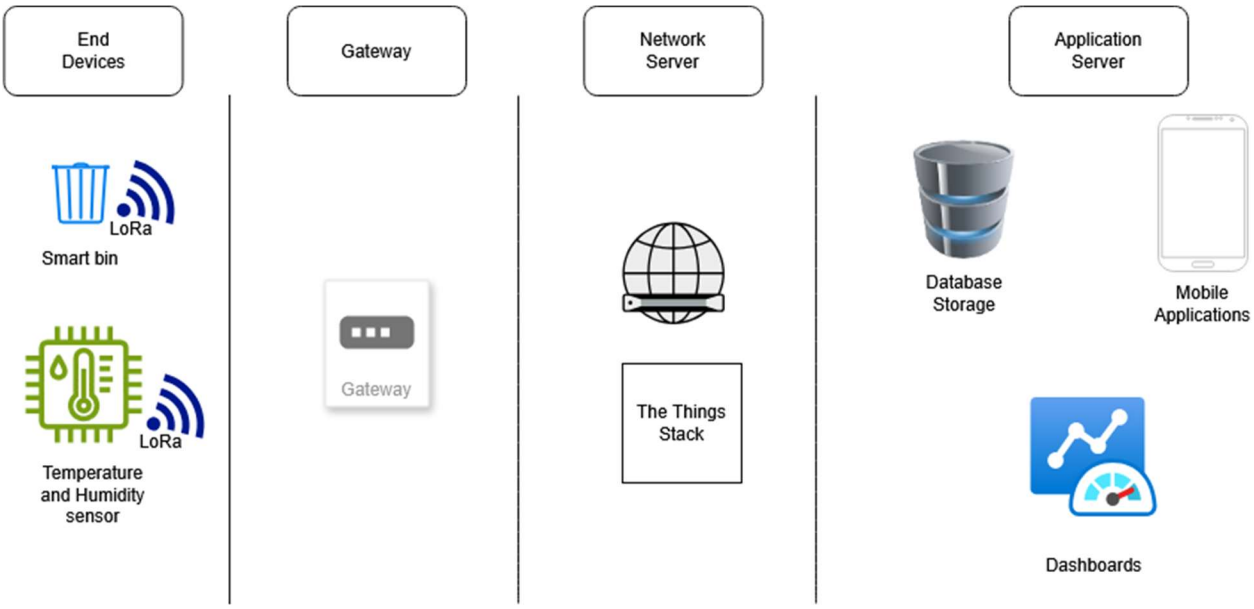



Figure 2.3: LoRaWAN network architecture

It consists of the end devices which transmit LoRa packets to the gateway or receive downlink packets, gateways that facilitate LoRa communication between the Network Server and LoRa-enabled nodes, the Network Server which functions as the core backend of the network in charge of network management and application servers which are responsible for processing of application data.

**B. GSM**

As per the definition by Awati, Ndungu, & Mixon (2025), GSM is "a digital mobile communication standard applied widely in Europe and other parts of the world." It makes use of TDMA where a 200 kilohertz radio channel is divided into 8 slots, each with the same bandwidth to accommodate 8 to 16 users to use a particular frequency for communication. The most common frequency bands used include 850MHz, 900MHz, 1800MHz and 1900MHz. Its network architecture consists of three smaller systems including Operation Subsystem (OSS), Network Switching Subsystem (NSS) and the Radio Subsystem (RSS), (Annamalai, 2024). The RSS is comprised of the Base Station Subsystem (BSS) which handles the signaling of mobile phones including audio, messaging and video with each BSS containing a few portable devices, the Base Transreceiver Station (BTS) which includes a radio transmitter and receiver, and the Base Station Controller

(BSC) for allocation of channels, reception of metrics from mobile devices and facilitation of handovers from one BTS to another and call establishment. The NSS features the Mobile Switching Center (MSC), the Home Location Register (HLR) which is a database for each cell number connected via GSM and the Visiting Location Register (VLR) which is a data repository containing data related to users roaming within the location of the MSC. Finally, the OSS comprises of the Operation and Maintenance Center (OMC) in charge of managing and maintaining the operation of mobile stations, Base Transreceiver Stations, Base Station Controllers and Mobile Switching Centers, the Authentication Center (AuC) responsible for creation and housing of authentication keys for all users in the HLR and the Equipment Identification Register (EIR) which houses records of International Mobile Equipment Identity (IMEI). Figure 2.4 represents the functional architecture of the GSM network.

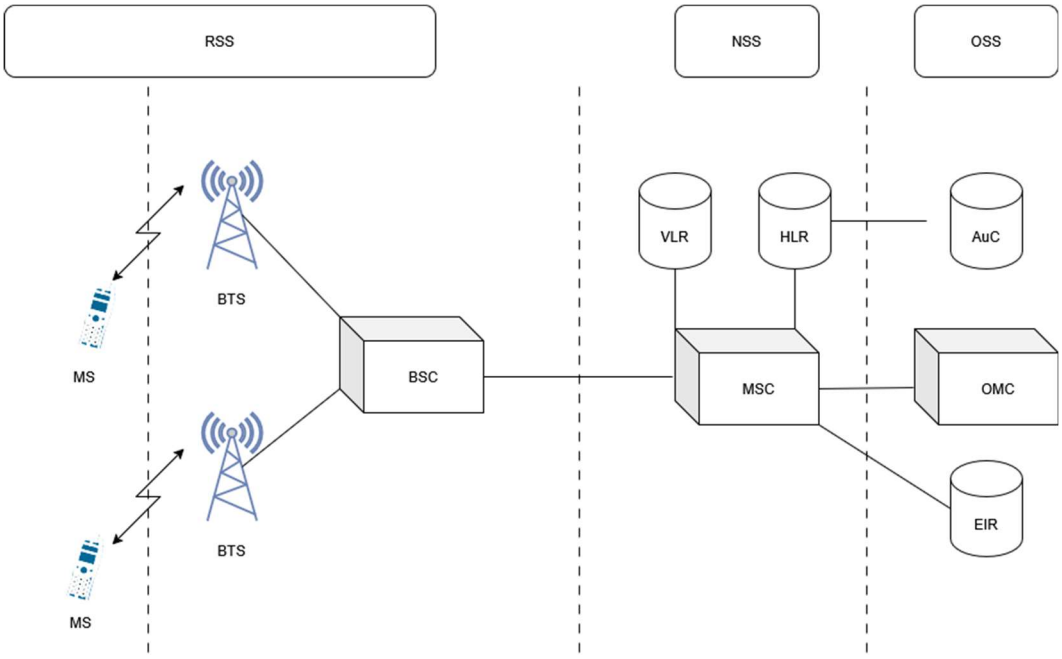



Figure 2.4: GSM functional architecture

## C. MQTT

MQTT is a publish/subscribe model-based lightweight communication protocol designed to support devices with low bandwidth and unstable networks to provide reliable transmission of information thus suitable for IoT applications (Wang et al., 2024). Figure 2.5 shows the publish/subscribe model MQTT. The publisher chooses a topic and writes data to the topic while subscribers subscribe to topics and get messages when availed by the publisher. The broker

functions as a middleman between the publisher and subscriber and is responsible for message delivery to the subscriber. A topic can be described as the classification of messages into different subject domains represented by a UTF-8 string such as /humidity or /kitchen/humidity (Usmani, 2021).

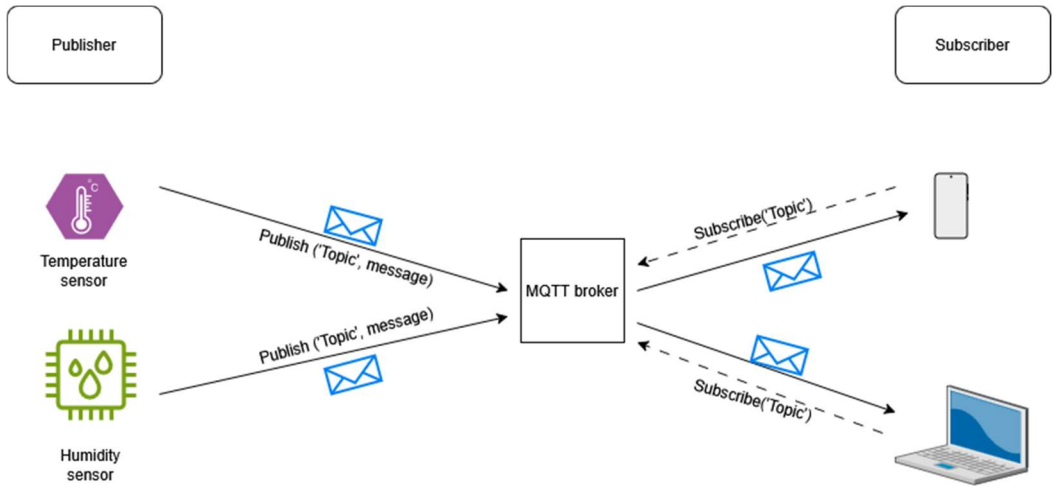



Figure 2.5: MQTT publish/subscribe model

MQTT is advantageous in that it is lightweight with a message size of at least 2 bits, supports TLS/SSL encryption, consumes less power, quick transmission, reliability due to Quality-of-Service levels, ability to connect multiple subscribers without knowledge of each other's existence, message persistence and better congestion control mechanisms (Usmani, 2021).

### 2.2.3 Security Mechanisms

#### A. AES

With a mutable key length of varying bits, AES is a private key algorithm which uses a 128-block cipher and 10,12 and 14 rounds hinged upon the length of the key used (Vimalkumar et al., 2023). Each round applies various permutation and substitution techniques to the cipher to ensure higher levels of confusion and diffusion. It operates quickly as it performs byte-level encryption and is susceptible only to brute force attacks. Its specifications as described by Advanced Encryption Standard (AES) (2023) include the four byte-oriented transformations performed on the state, a two-dimensional array of bytes of a 4x4 order matrix derived from dividing a 128-bit data length into four operational blocks. Table 2.1 provides a summary of these transformations and their respective operations.

Table 2.1: Summary of AES transformations

| Transformation | Operation |
|---|---|
| SUBBYTES() | Invertible, non-linear transformation on the state which singly applies a substitution table to each byte in the state. |
| SHIFTROWS() | Bytes in the state's last three rows are switched in a cyclic manner with the number of shifting positions depending on the row index. |
| MIXCOLUMNS() | All columns in the state undergo a product operation with a fixed matrix. |
| ADDROUNDKEY() | A round key is combined with the state by bit-by-bit XOR operations. |

### B. Hashing

Hashing as defined by Sharma (2024) is "the process of transforming input data of arbitrary size into a fixed-size value, known as a hash value or hash code through the application of a cryptographic hash function. These hash functions are meticulously designed to produce a unique and consistent hash value for any given input ensuring that even a minor change in the input results in a significantly different hash value." Their main function is to uphold the integrity of data, protect information and ensure efficiency when accessing data. These hash functions, in order of their years of introduction, include MD5 introduced in 1992, SHA-1 introduced in 1995, SHA-2 introduced in 2002 and finally SHA-3 introduced in 2015. They are applied in storing passwords and verifying them, digital signatures, hash tables and data structures, blockchain and cryptocurrency and file and data deduplication. An argument tabled by (Windarta et al., 2022) indicates that these current cryptographic functions are not ideal for all IoT environments due to their constrained resources. The paper discusses lightweight cryptographic hash functions (LWCHF) suited for IoT gadgets. The approaches to their design include three popular constructions namely Merkley-Damgard Construction, block ciphers and Sponge Construction which could be used by other researchers as starting points in designing robust and secure LWCHFs.

### C. OTP Based Authentication

OTP authentication generates a password whose validity lasts for a single use which minimizes the possibility of stealing passwords and unwanted access (Labhade-Kumar et al., 2024). They are advantageous in enhancing security, scalability and reliability as well as user-friendliness and data synchronization, but lack when it comes to limited offline access, risks of system downtime, initial setup complexity and dependency on mobile networks. However, they are effective and have varying applications including login and access management, real-time data access and role-based access control which prove their robust nature when included in multi-layered security frameworks and reliability in digital security.

#### 2.2.4 Backend and Cloud Services

**A. Firebase**

Firebase, as discussed by Madaminov & Allaberganova (2023) is described as "a Google database, one of the most popular high-speed systems for storing data in various formats on a server. It includes integration with other systems, use in authorization, high security system, data processing in milliseconds, use in various messaging applications and many other high-level services." It not only stores its data on the server but also the cloud in JSON format. The data is updated automatically when changes are made, and this is reflected in the mobile application as well. Firebase's Realtime partition is a non-relational database used for live communication and integration of data with clients. It utilizes a special API which allows the application to see changes occurring in the database, and the updated data can be observed and managed from different applications such as mobile and web interfaces. It provides other features including real-time data synchronization across connected devices, offline access to data due to data persistence to disk, scalability across multiple databases, flexible and expression-based rules language to define the structure of data and integration with Firebase Authentication for access control (Firebase Realtime Database, 2025).

**B. The Things Stack**

The Things Stack is a LoRaWAN network server which provides network linkage, administration and surveillance of nodes, gateways and applications (The Things Stack, 2025). Its purpose is to guarantee the network's data traversal and transmission security, scalability, and dependability. For production, it features cloud and enterprise deployments depending on whether one wants to handle the management of The Things Stack and their fleet of devices on their own or have The

Things Stack manage it, and for local testing, it offers sandbox and open-source options which are free but relatively limited. The Things Stack includes features such as durable webhooks which allow communication between their service and other third-party services supporting webhooks, gateway outage alerts to provide gateway-related updates to administrators for quick responses and lessen the interference to deployments and a comprehensive network operations center which allows administrators to register and receive insights on gateways, applications with metrics provided such as gateway uptime, radio utilization, end device packet rate and many more (The Things Stack Features, 2025). Figure 2.6 depicts the architecture of The Things Stack.

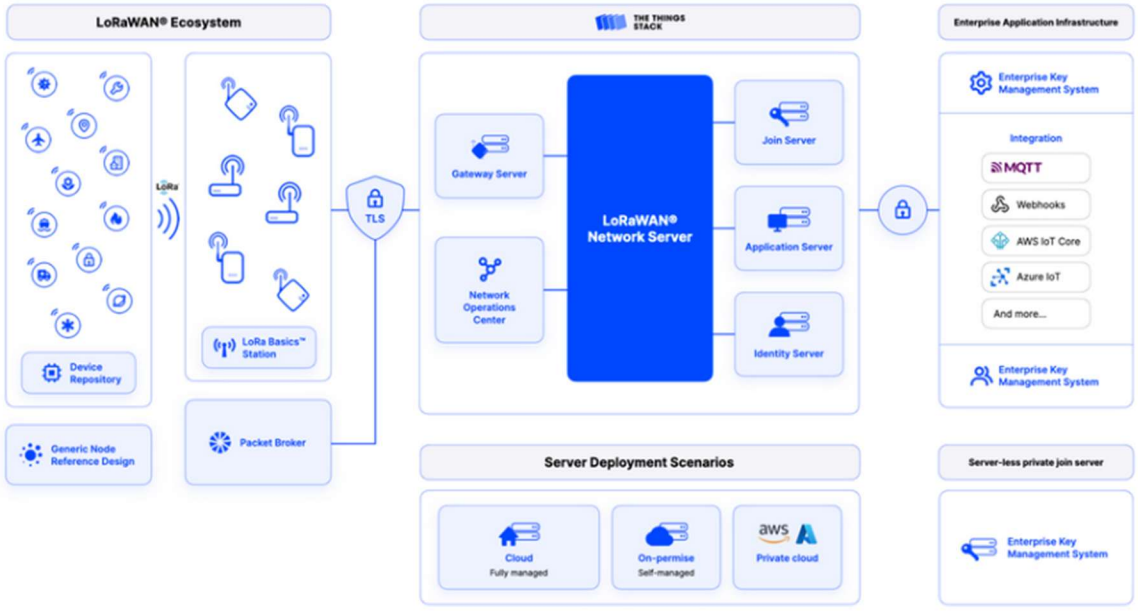



Figure 2.6: The Things Stack Architecture

Source: (The Things Stack Architecture, 2025)

Its main components include the Gateway Server which forwards uplink traffic and schedules downlink traffic, the Network Server which handles the LoRaWAN network layer, the Application Server in charge of the LoRaWAN application layer, the Identity Server which provides registries for storage of entities such as applications, organizations and users, the Join Server which handles the LoRaWAN join flow, the Device Claiming Server in charge of secure claiming of devices and gateways, the Gateway Configuration Server which creates configuration files for UDP gateways and manages gateway configuration, the Packet Broker Agent which connects the platform with

Packet Broker for outbound traffic exchange, the Network Operations Center and the Console for interaction with The Things Stack.

#### 2.2.5 Mobile Interfaces and Firmware Development

### A. Kotlin and Android Studio

According to Oliveira et al. (2020), "Kotlin is a statically typed programming language that runs on the JVM and fully interoperates with Java." Its official support on the Android platform was released by Google in 2017 and it eventually rose in favour among developers with adoption by companies such as Pinterest, Uber and Pivotal. It is a multi-paradigm language i.e. OOP and functional programming paradigms, allocates for non-nullable data types, intelligent type casting and functional programming via higher-order functions and extended class behaviour through extension functions (Oliveira et al., 2020). As per a study conducted by Martinez & Gois Mateus (2022) involving 98 developers to find out their reasons for switching to Kotlin, the supporting reasons were Kotlin provided features such as lambdas and smart casts which were not at their disposal with Java, and to write safer code due to null-pointer exceptions.

The endorsed development environment for Android app development based on developer tools from IntelliJ IDEA is Android Studio (Meet Android Studio, 2025). It includes features such as Gemini AI assistant which aids in code generation, debugging and responding to queries related to Android app development, Android Studio Cloud which is available in developers' browsers which helps in writing code anywhere and avoiding local installations and Jetpack Compose, a declarative UI framework which simplifies and speeds up UI development on Android platforms is powerful, intuitive, accelerates development and has less code which makes code simple and low maintenance. Nonetheless, it also features a smart code editor which allows auto-completion for code written in Kotlin, Java and C++ as well as Live Edit which mirrors changes in code as they happen, a flexible build system driven by Gradle which allows developers to generate different builds for different Android devices from the same project as well as build analysis to identify possible build issues via the Build Analyzer, an Android Emulator that enables one to test their application on different Android devices and it is also fast and feature rich and the Android App Bundle which caters for optimization of an application size before publishing and comparisons between two Android App Bundles to understand change in size between application versions.

## B. Firmware Development Languages

For firmware development, a variety of factors which might influence the choice of programming language to be utilized include developer experience, project requirements, target hardware, complexity and desired control over the IoT devices. From a survey conducted between January and March 2018 which included 502 people, it was found that the popular languages used for IoT included C, C++, Java, JavaScript, Python and PHP (Hong, 2020), with C++ and Java being general purpose languages and languages such as Go and Parasail being embedded specific choices. Over the recent past, the major languages used to develop IoT firmware are the Arduino Programming Language built on C/C++ and MicroPython and CircuitPython which are implementations of Python. Table 2.2 summarizes the evaluation conducted by (Das, 2023) on the three languages.

Table 2.2: Comparison between Arduino (C/C++), MicroPython and CircuitPython

| Comparison Parameters | Programming Language | | |
|---|---|---|---|
| | Arduino(C/C++) | MicroPython | CircuitPython |
| IDE Support | The popular IDE is Arduino IDE. Alternatives include PlatformIO IDE, Atmel Studio, CodeBlocks and EclipseIDE. | Thonny IDE and uPyCraft are popular. Alternatives include Pycharm and microIDE. | Recommended editor is Mu Editor for Adafruit boards with alternatives being GNU Emacs, VSCode and Sublime Text. |
| Code Execution | Code is compiled into machine code. Code can be optimized, and execution is fast and efficient. | Code is interpreted by MicroPython runtime environment which then executed the code. Code runs slower than compiled code. | Code is translated into bytecode then CircuitPython runtime environment takes care of code execution. Code runs slower than compiled code. |

| | | | |
|---|---|---|---|
| Development Boards | Supports an extensive range of boards from varying manufacturers including the Arduino family, ESP32 and ESP8266 MCUs and RP2040 boards from different manufacturers. | Official board is called the pyboard. Other supported boards include Raspberry Pi Pico, ESP32 and ESP8266. | Supports a variety of boards, mostly those from Adafruit. Examples are Raspberry Pi Pico, Arduino Nano RP2040 connect and SeeedStudio XIAO RP2040. |
| Cloud Support | Arduino Cloud allows remote development and monitoring of boards and collaboration. | MicroPython library can be used to connect device to Arduino Cloud, Google Cloud IoT Core and AWS IoT Core. | CircuitPython Adafruit IO module allows interaction with Adafruit IO via HTTP and MQTT APIs. |
| Documentation and Community Support | Large community and extensive documentation. | Bigger community compared to CircuitPython. | Relatively newer thus the community is not so big. |
| Ease of Use | Includes a lot of libraries available, but not very beginner friendly. | Quick and easy project development since Python is simple. | Easier compared to MicroPython since it has simplified syntax, standardized library collection and drag-and-drop functionality. |

### 2.3 IoT Vulnerabilities and Challenges

This section addresses the main weaknesses and difficulties with IoT systems that the project seeks to address, especially in delicate settings, emphasizing how these problems compromise the CIA triad.

### 2.3.1 DoS/DDoS Attacks

DoS and DDoS attacks primarily target devices by flooding them with forged traffic and fake connections to deny service to legitimate requests. This deters the delivery of required information between the IoT devices and the applications that require access to the data for processing. To propagate such attacks, attackers make use of botnets or distributed systems to render network resources unavailable for benign users (Siwakoti et al., 2023). These attacks mainly target the availability of services.

### 2.3.2 Man-in-the-Middle Attacks

MiTM attacks occur when an adversary can place themselves on the communication channel in the middle of communicating devices. The attacker can then monitor and intercept network traffic between the communicating nodes and can further take advantage of this by assuming one of the devices' identities and communicating with the other device to gain access to information (Alqarawi et al., 2023). This attacks the confidentiality of data in IoT applications.

### 2.3.3 Weak Authentication and Authorization Mechanisms

IoT devices lack standard authentication mechanisms and as such, attackers can exploit application updates or inject malicious payloads into IoT ecosystems to penetrate IoT applications and reveal sensitive data (Alqarawi et al., 2023).

### 2.3.4 Failure in Encryption

Lack of encryption exists as one of the major challenges in IoT applications. A report by the Internet of Things Security Foundation highlighted that extremely few IoT devices have proper encryption levels set when sending out data which makes them susceptible to attacks (Madiiarbekova, 2025). An example is the Dyn DDoS attack in 2016 where hackers attacked large sites including Twitter and Netflix by using a botnet of unsecured IoT devices.

## 2.4 Existing Solutions for IoT Security

With the growth of IoT devices, various solutions have been proposed and implemented to tackle the increasing security related issues in IoT environments. This section presents these solutions and their technical approaches to securing IoT applications.

### 2.4.1 LoRa and LoRaWAN Security

LoRaWAN 1.0 includes keys such as the Network Session Key (NwkSKey), the Application Session Key (AppSKey) and the Application Key (AppKey) (Tournier et al., 2021). The keys are

128 bits long with the encryption suite being AES-128 bit (LoRaWAN, 2025). The AppSKey and NwkSKey are yielded when a node joins the network with the NwkSKey being synced with the network and the AppSKey remains confidential. Both are unique to each device for each session. To join a network, nodes can either use over-the-air-activation (OTAA) dynamically or via customized activation which is static. The combination of session keys provides nodes with the ability to encrypt and validate the message integrity in the network (Laghari et al., 2024). The NwkSKey ensures the integrity of messages via a Message Integrity Code (MIC) check which resembles a checksum to prevent message tampering. The AppSKey on the other hand is used for ciphering and deciphering of the payload (LoRaWAN, 2025).

### 2.4.2 Lightweight Cryptography

Security solution tailored for devices with limited resources. There are three types namely private key lightweight cryptographic algorithms, asymmetric key lightweight cryptographic algorithms and hash functions (Litoussi et al., 2020). The criteria when identifying an algorithm's lightweight nature include the software and hardware weights of the cipher used (Williams et al., 2022). The software weight is identified by the duration it takes to successfully encrypt and decrypt messages and the memory required to execute the function of cipher-generation. The hardware complexity is determined by the area of the cipher depicted by the gate equivalencies (GE) and power it consumes during its execution. Such cryptographic solutions are implemented at the perception layer, on the IoT devices themselves to leverage these mechanisms for security.

### 2.4.3 Public Key Infrastructure

PKI systems combine varying mechanisms such as intrusion detection, authentication and authorization as well as RSA-based encryption to generate public keys housed at Base Stations and private keys managed by each device (Litoussi et al., 2020). PKI ensures data is secured from both sides in a communication channel by encrypting and decrypting data at the client's side using the keys for communication to avoid tampering and cyber-attacks (Laghari et al., 2024).

### 2.4.4 IoT Authentication Techniques

To ensure users' identities before being authorized to access data, (Azrour et al., 2021) highlights various authentication mechanisms and schemes. They include:

i. OTP Authentication – protocols reliant on time synchronization, hash functions and cryptographic RSA, with proposals to implement strong OTP IoT mechanisms by using identity-based ECC and Lamport's OTP technique.

ii. ECC-oriented Common Authentication – better than traditional RSA encryption algorithms, Elliptic Curve Cryptography (ECC) mechanisms are viewed to have greater efficiency and secure for devices with limited resources such as IoT devices.

iii. ID and Password Based Authentication – employs attributes that can distinguish people such as usernames, email and phone numbers and requires a server to store user IDs and passwords. However, this is also quite limited where servers may not adequately secure passwords, users might forget their credentials and user IDs transmitted over public networks are vulnerable to sniffing but this can be curbed by using hashing or cryptographic algorithms.

iv. Certificate Based Authentication – proposed to curb issues with ID and Password Based Authentication to verify users' identities which is more secure, but the processing resources are quite high thus not suitable for IoT.

v. Blockchain – different type of database which stores data as a series of blocks connected to each other which have been leveraged to propose authentication protocols for IoT ecosystems.

**2.5 Deficiencies of Existing Solutions and Technologies**

The existing solutions provide relevant security for IoT applications and deployments, but they have varying limitations when it comes to implementation, resource demands or wholeness. For LoRa and LoRaWAN security, the security model places trust in the network server whose compromise could cause potential vulnerabilities. Attackers could pose as the network server and have access to the sensor data as it traverses the network and use advanced techniques to decipher the data in the event it was not encrypted before transmission. With lightweight cryptography, the various proposed algorithms offer resource efficient encryption solutions but lack standardization which dictate their use and guide the steps to be taken in their inclusion in IoT security. They also tend to focus on data confidentiality alone, neglecting the other principles of the CIA triad unless complemented by other techniques and algorithms. PKI uses key management for authentication

and integrity of data but relies on Certificate Authorities (CAs) to handle issuing digital certificates and public key management. Attackers can assume the role of CAs, sign and distribute their certificates and the nodes will obliviously share data with the attackers. Furthermore, handling digital certificates on the nodes might tend to get computationally expensive. For the authentication techniques, OTP schemes are relatively susceptible to replay attacks, ID and password based authentication fail when passwords are insecurely stored or when credentials are shared over the network and they are then vulnerable to network sniffing, certificate based authentication is resource intensive when it comes to handling certificates and processing while blockchain similarly consumes a lot of resources especially power and memory and is also quite complex.

### 2.6 Conceptual Framework

The conceptual framework of the proposed solution shall begin with data collection from the bank monitoring IoT ecosystem. The IoT ecosystem will incorporate buzzers to function as alarms, a light dependent resistor to detect tampering where lighting is required, LEDs as status indicators to blin when alarms are triggered or upon user commands, OLED displays to display various metrics such as temperature and humidity, an ultrasonic sensor mounted on a servo motor to shift through varying degrees to detect objects, PIR sensor to detect motion, GSM module for GSM connectivity, LoRa modules for LoRa connectivity and a DHT11 module for measuring temperature and humidity. Once these various signals are captured, they are encrypted, a hash value is calculated, and an OTP is generated before encapsulation into a LoRa packet and transmission for storage. The primary network for transmission is LoRaWAN since it features 128-bit encryption thus enhancing data confidentiality. The data is aggregated at the LoRaWAN gateway which then forwards the data to Firebase Realtime database for storage. In the event the LoRaWAN network fails, the system uses GSM as a fallback and sends data to the database to enhance availability of data. The data protocol for transmission will be MQTT over port 8883 for encryption of data. Before the data is stored, the hash is verified to ensure only untampered data is stored. The data is then accessible to a user via the mobile application where the OTP will have to be provided before data is accessed. Decryption of data and further hash calculation will occur at the mobile application to ensure confidentiality and integrity of the data before being availed to users. Figure 2.7 represents the conceptual framework for the proposed solution.

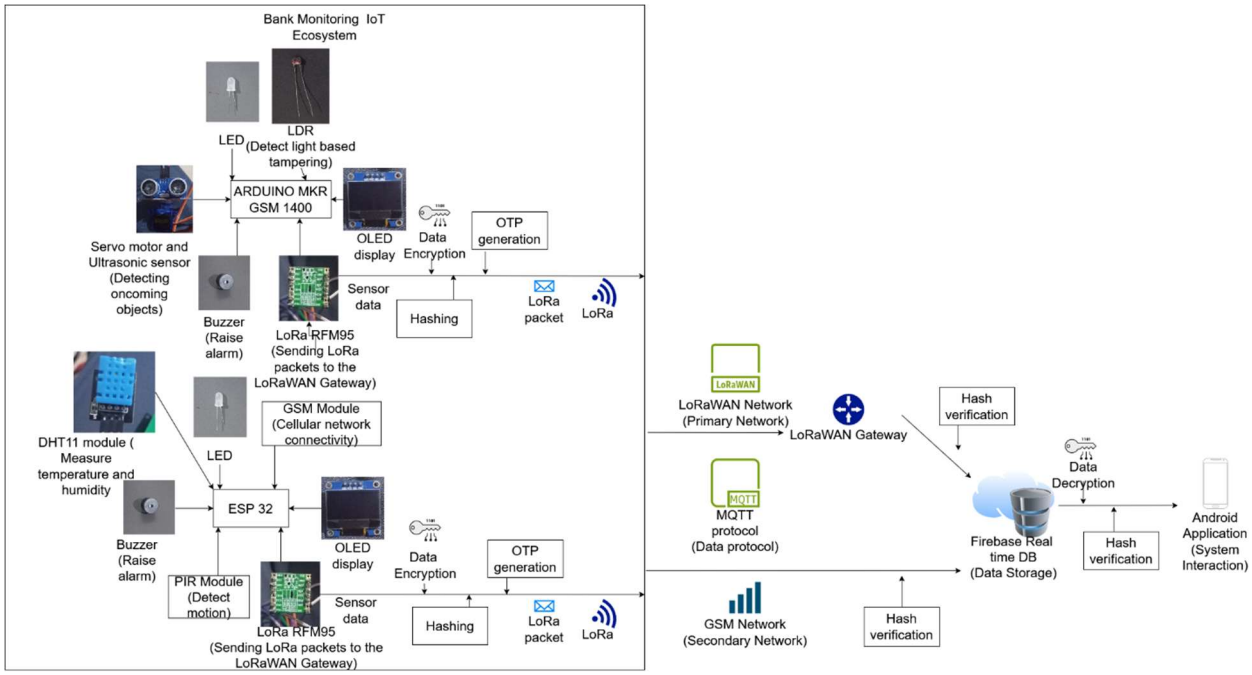



Figure 2.7: Conceptual Framework

# Chapter 3: Methodology

## 3.1 Introduction

This chapter presents the methodology adopted in the design and development of the proposed CIA-triad-oriented security architecture for IoT systems. It discusses the key design requirements, both functional and non-functional requirements, as well as the implementation requirements including hardware and software needs required to implement the proposed solution.

## 3.2 Project Methodology

This section highlights the systematic approach which will be embraced for the design, development, and implementation of the secure IoT-based bank monitoring system guided by the CIA triad principles. First, an in-depth evaluation of existing literature will be conducted on the relevant IoT technologies required for the project, security vulnerabilities and challenges in IoT as well as existing solutions to help in securing IoT deployments. The review will require delving into conference papers, journal articles, books and official websites to eke out required information. Next, an IoT-based bank monitoring system will be designed by drawing relevant system diagrams to analyze how the system developed using sensors and microcontrollers with network capabilities to actualize a mission-critical IoT deployment along with an Android application for interaction with the IoT system. To build the mobile application and the system, prototyping will be employed. It is defined as a system of creating software in which prior versions (prototype) are created, tested and refined as needed until a satisfactory version is realized and from it the whole system can be constructed (Lewis, 2023) Figure 3.1 indicates the various stages of the selected methodology.

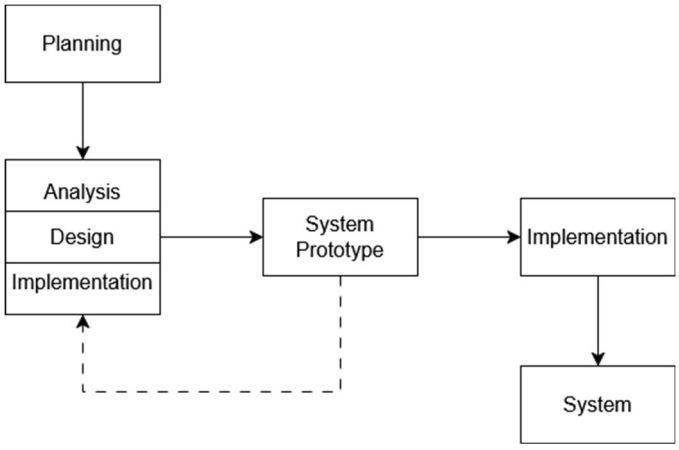



Figure 3.1: Prototyping Methodology diagram

### 3.2.1 Justification for the Methodology

Prototyping will be chosen since it effectively reduces the time spent in development by utilizing operational models of the final system early in the project. Furthermore, it allows for adaptability of changes early enough before the final system is actualized and prototypes can be used to track progress within the project development. It also aids in minimizing the chances of encountering failure in the project by shedding light on errors early enough in the process. In this methodology, evolutionary prototyping will be adopted where the prototypes will be reused and refined to reduce time spent in development (Prototyping Model - Software Engineering, 2025). It helps conserve time and effort spent in developing the system.

### 3.2.2 Applying Stages to the Project

#### A. Planning

Planning will involve critically calculating and outlining the required objectives for the project and the end goal to be achieved. This will include the preparation of the project proposal featuring the project's objectives and justification, literature review and research on existing solutions and highlighting the methodology adopted for the project as well as the feature and implementation requirements. The scope of the project was also identified as well as the problem and outlining the technologies to be used in implementing the proposed solution. The deliverables at this stage will be a concept not, the project proposal and list of hardware and software to be used in the project.

#### B. Analysis

Analysis will include specifying the requirements, both functional and non-functional, and the hardware and software needs required to achieve the system's full functionality. The assessments carried out will review the security needs of mission-critical IoT systems, potential vulnerabilities targeting IoT systems and existing solutions and their limitation in securing IoT deployments. The deliverables will include an in-depth literature review which will feature in the project proposal as the second chapter.

#### C. Design

At this stage, a preliminary design will be developed to give a quick overview of the system to aid in creating the prototype. It will inform the basic architecture of the system, from the device to the gateway using LoRa connectivity, to The Things Stack and then to Firebase and finally to the

mobile application. The deliverables at this stage will be system diagrams, circuit diagrams for the hardware connectivity and wireframes for the mobile application.

### D. System Prototype

This stage of the methodology will be where a functioning model of the system will be developed. Evolutionary prototyping will be employed to make use of existing prototypes and as such, each prototype will be created to target a single requirement of the final system. The prototypes will be built incrementally with increasing functionalities in order of the level of complication. The sensor subsystems will be built to collect relevant data, displays will be added, and networking features will complete this iterative process. The mobile application will also be developed at this stage to accommodate the access of sensor data once the IoT prototypes are complete. The deliverables will be working ESP32 and Arduino MKR GSM prototypes with sensor data transmission.

### E. Implementation

Implementation will then involve adequate testing and integration of the prototypes into the final working system. Each prototype will be subjected to modular testing with relevant test cases drawn, to test various aspects of the functionality desired at that stage. The testing types will be white box testing since the expected results will be known, and system integration testing to ensure the subsequent prototypes with increased functionalities of each implementation. The process will be iterated where after testing, the prototypes will be evaluated to investigate their strengths and weaknesses, and further refinement will be carried out till the prototype will meet the desired functionality and before realization as the complete and final system. Here, the deliverables will be a fully integrated working version of the IoT bank monitoring system with data storage and mobile application access of sensor data.

### F. System

This will be the final stage whereby the iterative stages of implementation, design and system prototype will be complete, and the system will be fully complete with reliable data flow from the nodes to the application, complete implementation of security protocols in line with the CIA-triad and proper documentation of the entire process. The deliverables at this final stage will include the complete final system and project report with all six chapters detailing the entire project.

## 3.3 Feature Requirements

This section details the fundamental features the proposed solution must fulfil to meet the set objectives. It includes the functional requirements and non-functional requirements, which detail the features and functions that are mandatory to be implemented for the required objectives to be met and complement the quality of the solution respectively.

### 3.3.1 Functional Requirements

The IoT system will collect relevant live data from the sensors such as temperature, resistance, humidity and distance data. Furthermore, it will allow authorized users to access the data through a custom-designed Android application. The data will then be encrypted before transmission over LoRaWAN or GSM and decrypted before being displayed in the mobile application to preserve confidentiality. To ensure data integrity, a hash value will be generated alongside the original data before transmission. It will be verified both before storage in the database to avoid storing bogus data and before presentation in the mobile application. Sensor data will be securely stored in Firebase for real-time monitoring, logging and analysis. In the event that the LoRaWAN network is unavailable, the system will use GSM as a backup communication network for redundant data transmission. The mobile application will also use OTP-based verification to limit access to sensitive data, making sure that only users who have been verified can access and interact with the system's data.

### 3.3.2 Non-functional Requirements

The mobile application will be designed with proper UI/UX principles to improve the overall user experience and ensure users will be able to easily and efficiently access sensor data from the IoT-based bank monitoring system. Furthermore, the application will employ secure Email and Password Based authentication to control access to the application due to its ease of integration with Firebase along with features such as email verification and password recovery. For the IoT bank monitoring application, the system will be designed to send SMS alerts when the sensors are triggered beyond the predefined thresholds, and it will be powered by lithium-ion batteries which will be recharged by solar panels to enhance the sustainability of the proposed system. To strengthen the overall maintainability of the system, the firmware and applications will be well-documented to facilitate future development, updates and debugging. A 3D housing structure will be designed and printed using a 3D printer to accommodate the IoT components including the microcontrollers and sensors for enhanced protection, organization and portability of the system.

The enclosure will aid in shielding the electronics from environmental damage and improve its aesthetic appearance.

## 3.4 Implementation Requirements

The hardware components, sensors, and microcontrollers needed to construct the suggested solution are highlighted in this section, together with the software tools and platforms needed to configure, implement, and oversee the whole IoT ecosystem.

### 3.4.1 Hardware Requirements

To undertake this project, a set of hardware components will be required to support sensor integration, data transmission and secure operation. The microcontrollers of choice will be Arduino MKR GSM 1400 selected for its inbuilt GSM capabilities thus requiring no additional hardware for GSM connectivity and the ESP32 38-pin version because it has relatively more GPIO pins to interface other sensors. The sensors and components to be used include LEDs to function as status indicators, a PIR sensor which detects infrared radiation emitted or reflected by objects to detect motion, a light dependent resistor to detect light based tampering, an ultrasonic sensor placed on a servo motor to detect oncoming objects by shifting the sensor's position across varying degrees, RFM95 LoRa modules for LoRa and LoRaWAN capabilities since they offer wide spread spectrum, robust resistance to interference and reduced power usage, omnidirectional antennas for transmitting and receiving signals to and from the LoRaWAN gateway, an A9G module to provide the ESP32 with GSM connectivity as the module offers both GSM and location capabilities, 128x64 OLED screens to display sensor data, buzzers to function as alarms when the sensors are triggered and a DHT11 module to detect temperature and humidity because it saves on the pin allocation while capturing both temperature and humidity and resistors to regulate the flow of current in the circuits. To upload the code for the firmware to the microcontrollers, a micro-USB cable will be used due to its compatibility with both microcontrollers. Jumper wires will be used to connect the sensors and components to the microcontrollers, with the help of a breadboard and an ESP32 expansion breakout board since they allow the components to be interfaced with the microcontrollers without the need for soldering. To power the system, external lithium-ion batteries will be used and recharged via solar panels, supported by HW-107 micro-USB battery charging board charger modules. A 3D model of a housing structure will be printed to house the IoT equipment using an Ender Neo Max 3D printer and a filament. When it comes to network connectivity, IoT SIM cards will be utilized for GSM coverage as a backup network and a

LoRaWAN gateway on campus for LoRaWAN as the main network for transmission. Lastly, a mobile phone running on Android will be utilized to communicate with the system to provide access to the monitoring data.

### 3.4.2 Software Requirements

The development and operation of the proposed CIA-triad-oriented bank monitoring system will rely on several software tools and platforms for programming, handling data, developing a mobile application and securing data. Firstly, Arduino IDE will be used for writing, compiling and uploading firmware developed in C++ to the ESP32 and Arduino MKR GSM 1400 microcontrollers and it supports relevant board packages and libraries for sensor integration and communication protocols. To develop the Android application, Android Studio will be used to program the application using Kotlin as well as SDKs e.g. Firebase to integrate real-time data integration. The application will function as the user interface for monitoring real time data, triggering system actions and implementing OTP-based authentication for secure access. For the backend database, Firebase will be used to store and manage sensor data because it offers cloud-based real time data synchronization, built in authentication and secure data handling features. The Things Stack (TTS) will be used to register and manage LoRaWAN nodes, decode payload data and route sensor data from the LoRa enabled IoT ecosystem to Firebase storage. Its selection is due to its ease of use as the Network Server in the LoRaWAN architecture. To test the A9G module's GSM connectivity, Aithinker Serial Tool will be used by sending the appropriate AT commands. Fritzing will be used to draw circuit diagrams since it has a wide range of components and a large community of developers who make and avail custom parts for use and Draw,io will be used to draw the necessary diagrams required for the design of the proposed solution. To design the housing structure for the IoT components, UltiMaker Cura will be used since its 3D files are compatible with the Ender Max Neo 3D printer. For version control, Git will be used since it is open source, efficient in working with projects of varying sizes, it has multiple workflows and supports local branching and not only is it well documented but also has large community support (Git, 2025). This will be used in tandem with GitHub, a web-based platform which allows the creation, modification, storage and collaboration of code among developers (What Is GitHub and Why Should You Use It?, 2025). Finally, Microsoft word will be used to document the proposal and final report of the project since it is easy to use and has extensive features that ensure proper

formatting and editing in a professional and admissible manner and it will be complemented with Mendeley Reference Manager to handle citations of literature used in preparation of the document.

## References


(2025). Retrieved from Git: https://git-scm.com/

*Advanced Encryption Standard (AES)*. (2023). https://doi.org/10.6028/NIST.FIPS.197-upd1

Almuhaya, M. A. M., Jabbar, W. A., Sulaiman, N., & Abdulmalek, S. (2022). A Survey on LoRaWAN Technology: Recent Trends, Opportunities, Simulation Tools and Future Directions. *Electronics*, *11*(1), 164. https://doi.org/10.3390/electronics11010164

Alqarawi, G., Alkhalifah, B., Alharbi, N., & El Khediri, S. (2023). Internet-of-Things Security and Vulnerabilities: Case Study. *Journal of Applied Security Research*, *18*(3), 559–575. https://doi.org/10.1080/19361610.2022.2031841

Annamalai, C. (2024). *TCP/IP and Cellular Networks of GSM*. https://doi.org/10.33774/coe-2024-n0l8r

Awati, R., Ndungu, S., & Mixon, E. (2025, May 27). What is GSM (Global System for Mobile Communications)? Retrieved from TechTarget: https://www.techtarget.com/searchmobilecomputing/definition/GSM

Azrour, M., Mabrouki, J., Guezzaz, A., & Kanwal, A. (2021). Internet of Things Security: Challenges and Key Issues. *Security and Communication Networks*, *2021*, 1–11. https://doi.org/10.1155/2021/5533843

Birlog, I.-A., Borcan, D.-M., & Covrig, G.-M. (2020). Internet of Things Hardware and Software. *Informatica Economica*, *24*(2/2020), 54–65. https://doi.org/10.24818/issn14531305/24.2.2020.05

Das, A. (2023, April 7). Arduino vs MicroPython vs CircuitPython: Which One Will You Choose? Retrieved from Electrocredible: https://electrocredible.com/arduino-vs-micropython-vs- circuitpython/

Farooq, J., & Zhu, Q. (2021). *Resource Management For On-Demand Mission-Critical Internet of Things Applications* (1st ed.). Wiley-IEEE Press.

Firebase Realtime Database. (2025, July 4). Retrieved from Firebase: https://firebase.google.com/docs/database

Hong, S. (2020). AN EFFICIENT IOT APPLICATION DEVELOPMENT BASED ON IOT KNOWLEDGE MODULES. *Issues In Information Systems*. https://doi.org/10.48009/3_iis_2020_72-82

Jouhari, M., Saeed, N., Alouini, M.-S., & Amhoud, E. M. (2023). A Survey on Scalable LoRaWAN for Massive IoT: Recent Advances, Potentials, and Challenges. *IEEE Communications Surveys & Tutorials*, *25*(3), 1841–1876. https://doi.org/10.1109/COMST.2023.3274934

Karie, N. M., Sahri, N. M., Yang, W., Valli, C., & Kebande, V. R. (2021). A Review of Security Standards and Frameworks for IoT-Based Smart Environments. *IEEE Access*, *9*, 121975–121995. https://doi.org/10.1109/ACCESS.2021.3109886

Labhade-Kumar, N., Kumar, N., Kamje, R., Landge, P., Shitole, S., & Takale, B. (2024). *OTP-Based Authentication System*. https://www.researchgate.net/publication/387079266

Laghari, A. A., Li, H., Khan, A. A., Shoulin, Y., Karim, S., & Khani, M. A. K. (2024). Internet of Things (IoT) applications security trends and challenges. *Discover Internet of Things*, *4*(1), 36. https://doi.org/10.1007/s43926-024-00090-5

Lewis, S. (2023, June 9). Prototyping Model. Retrieved from TechTarget: https://www.techtarget.com/searchcio/definition/Prototyping-Model

Litoussi, M., Kannouf, N., El Makkaoui, K., Ezzati, A., & Fartitchou, M. (2020). IoT security: challenges and countermeasures. *Procedia Computer Science*, *177*, 503–508. https://doi.org/10.1016/j.procs.2020.10.069

LoRaWAN. (2025). Retrieved from The Things Network: https://www.thethingsnetwork.org/docs/lorawan/security/

LoRaWAN. (2025). Retrieved from The Things Network: https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan

Madaminov, U. A., & Allaberganova, M. R. (2023). Firebase Database Usage and Application Technology in Modern Mobile Applications. *2023 IEEE XVI International Scientific and Technical Conference Actual Problems of Electronic Instrument Engineering (APEIE)*, 1690–1694. https://doi.org/10.1109/APEIE59731.2023.10347828

Madiiarbekova, A. (2025). *Cybersecurity in IoT Devices: Vulnerabilities, Risks, and Mitigation Strategies*. https://doi.org/10.2139/ssrn.5075016

Márquez-Vera, M. A., Martínez-Quezada, M., Calderón-Suárez, R., Rodríguez, A., & Ortega-Mendoza, R. M. (2023). Microcontrollers programming for control and automation in undergraduate biotechnology engineering education. *Digital Chemical Engineering*, *9*, 100122. https://doi.org/10.1016/j.dche.2023.100122

Martinez, M., & Gois Mateus, B. (2022). Why Did Developers Migrate Android Applications From Java to Kotlin? *IEEE Transactions on Software Engineering*, *48*(11), 4521–4534. https://doi.org/10.1109/TSE.2021.3120367

Meet Android Studio. (2025, 5 20). Retrieved from Android.com: https://developer.android.com/studio/intro

Nižetić, S., Šolić, P., López-de-Ipiña González-de-Artaza, D., & Patrono, L. (2020). Internet of Things (IoT): Opportunities, issues and challenges towards a smart and sustainable future. *Journal of Cleaner Production*, *274*, 122877. https://doi.org/10.1016/j.jclepro.2020.122877

Oliveira, V., Teixeira, L., & Ebert, F. (2020). On the Adoption of Kotlin on Android Development: A Triangulation Study. *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 206–216. https://doi.org/10.1109/SANER48275.2020.9054859

Prototyping Model - Software Engineering. (2025, April 12). Retrieved from Geeksforgeeks: https://www.geeksforgeeks.org/software-engineering-prototyping-model/

Sharma, S. (2024, June). *A Comprehensive Study of Cryptographic Hash Functions*. https://www.researchgate.net/publication/381639930

Siwakoti, Y. R., Bhurtel, M., Rawat, D. B., Oest, A., & Johnson, R. C. (2023). Advances in IoT Security: Vulnerabilities, Enabled Criminal Services, Attacks, and Countermeasures. *IEEE Internet of Things Journal*, *10*(13), 11224–11239. https://doi.org/10.1109/JIOT.2023.3252594

The Things Stack. (2025). Retrieved from The Things Industries: https://www.thethingsindustries.com/stack/

The Things Stack Architecture. (2025). Retrieved from The Things Industries: https://www.thethingsindustries.com/docs/concepts/architecture/

The Things Stack Features. (2025). Retrieved from The Things Stack: https://www.thethingsindustries.com/stack/features/

Tournier, J., Lesueur, F., Mouël, F. Le, Guyon, L., & Ben-Hassine, H. (2021). A survey of IoT protocols and their security issues through the lens of a generic IoT stack. *Internet of Things*, *16*, 100264. https://doi.org/10.1016/j.iot.2020.100264

Usmani, M. F. (2021). *MQTT Protocol for the IoT-Review Paper MQTT Protocol for the IoT*. https://doi.org/10.13140/RG.2.2.26065.10088

Vimalkumar, J., Babu, H. R., & M, B. (2023). FPGA Implementation of Modified Lightweight 128-Bit AES Algorithm for IoT Applications. *2023 IEEE International Symposium on Smart Electronic Systems (ISES)*, 306–309. https://doi.org/10.1109/iSES58672.2023.00069

W. N. Abidde, N. Eyidia, & E. S. J. Eme. (2025). CIA Triad: A Review of the Confidentiality, Integrity and Availability of Data in a System of Connected Networks. *International Research Journal of Modernization in Engineering, Technology and Science*, *7*(3).

Wang, W., Zhao, Y., Liu, Y., Liu, G., Zheng, F., & Sun, C. (2024). MQTT Protocol and Implementation of Equipment Management System for Industrial Internet of Things. *2024 43rd Chinese Control Conference (CCC)*, 6139–6144. https://doi.org/10.23919/CCC63176.2024.10662474

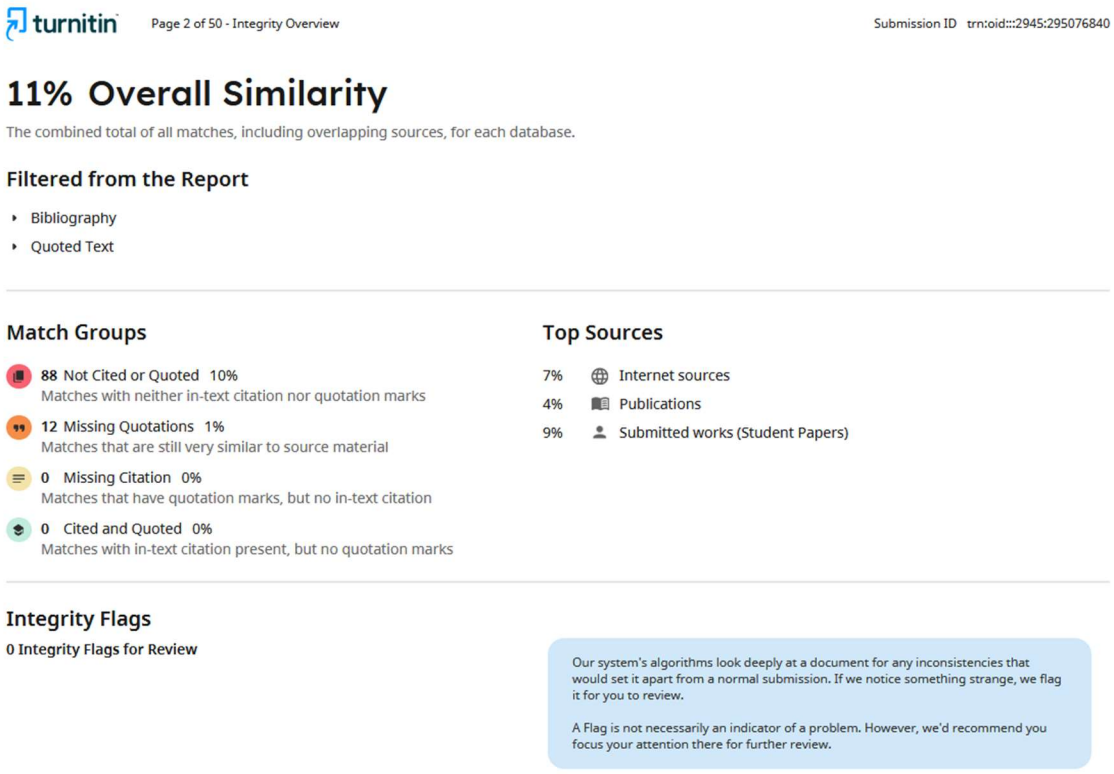
What Is GitHub and Why Should You Use It? (2025, March 9). Retrieved from Coursera: https://www.coursera.org/articles/what-is-git

Williams, P., Dutta, I. K., Daoud, H., & Bayoumi, M. (2022). A survey on security in internet of things with a focus on the impact of emerging technologies. *Internet of Things*, *19*, 100564. https://doi.org/10.1016/j.iot.2022.100564

Windarta, S., Suryadi, S., Ramli, K., Pranggono, B., & Gunawan, T. S. (2022). Lightweight Cryptographic Hash Functions: Design Trends, Comparative Study, and Future Directions. *IEEE Access*, *10*, 82272–82294. https://doi.org/10.1109/ACCESS.2022.3195572

# Appendices

## Appendix 1: Turnitin Digital Report



turnitin Page 2 of 50 - Integrity Overview Submission ID trn:oid:::2945:295076840

**11% Overall Similarity**

The combined total of all matches, including overlapping sources, for each database.

**Filtered from the Report**

- Bibliography
- Quoted Text

**Match Groups**

- **88** Not Cited or Quoted 10%
  Matches with neither in-text citation nor quotation marks
- **12** Missing Quotations 1%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

**Top Sources**

7% Internet sources
4% Publications
9% Submitted works (Student Papers)

**Integrity Flags**

**0 Integrity Flags for Review**

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.