

HARDWARE AUTHENTICATION AND ENCRYPTION MODULE

(HEAM)



By

Khalid Khushal
Ahmed Salar
Shahid Naqash
Tahir Rahim

Supervised by:

Project Supervisor: Asst. Prof. Dr Fawad Khan

Co-Supervisor: Asst. Prof. Shahzaib Tahir

Submitted to the faculty of Department of Electrical Engineering,
Military College of Signals, National University of Sciences and Technology, Islamabad,
in partial fulfillment for the requirements of B.E Degree in Electrical (Telecom) Engineering.

May 2022

In the name of ALLAH, the Most benevolent, the Most Courteous

CERTIFICATE OF CORRECTNESS AND APPROVAL

This is to officially state that the thesis work contained in this report

“Hardware Authentication and Encryption Module”

is carried out by

Khalid Khushal

Ahmed Salar

Shahid Naqash

Tahir Rahim

under my supervision and that in my judgement, it is fully ample, in scope and excellence, for the degree of Bachelor of Electrical (Telecom.) Engineering in Military College of Signals, National University of Sciences and Technology (NUST), Islamabad.

Approved by



Supervisor

Project Supervisor: Asst. Prof. Fawad Khan

Department of IS, MCS

Date: May 23, 2022

DECLARATION OF ORIGINALITY

We hereby declare that no portion of work presented in this thesis has been submitted in support of another award or qualification in either this institute or anywhere else.

ACKNOWLEDGEMENTS

Allah Subhan'Wa'Tala is the sole guidance in all domains.

Our parents, colleagues, and most of all supervisor Asst. Prof. Fawad Khan
, who remained our pillar of support despite the obstacles. We owe him our gratitude for
instilling in us his vision, knowledge, and mentality, and we feel privileged and proud to have
benefited from his mentoring and guidance.

The group members, who through all adversities worked steadfastly.


Plagiarism Certificate (Turnitin Report)

This thesis has 6% similarity index. Turnitin report endorsed by Supervisor is attached.



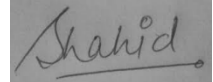
Khalid Khushal

NUST Serial no 00000256577



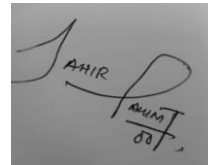
Ahmed Salar

NUST Serial no 00000278782



Shahid Naqash

NUST Serial no 00000258975



Tahir Rahim

NUST Serial no 00000278798



Signature of Supervisor

ABSTRACT

The field of technology and worldwide data exchange is constantly growing. One of its most promising outcomes has been the introduction of IOT and its uses. In the field of smart ecosystems IOT sensors are deployed and the corresponding data is used for specific purposes. The IOT sensors are resource and power constrained meaning they don't provide any reasonable authentication and encryption of the data which results in the traversal of data in unencrypted form making it vulnerable and easy to alter. The software-based encryption is also done on the edge layer meaning there is no way of encrypting the data prior to it being shared, which in case of insecure gateways can be very harmful. HEAM is a black box easy to use device which is made up by x2 ESP32 (Sender and receiver) devices. The device acts as a gateway for the IOT sensors where the data is encrypted at run time prior to being shared and then the receiver HEAM device decrypts the corresponding Cipher for its use case. The project aims to provide safe traversal of data where sensitive data is being shared and no secure mechanisms are in place. When implemented, this project can be beneficial in providing encryption and authentication to all kind of IOT sensory communication making it a vital part of everyone's life in the smart eco system.

Key Words: *ESP32, Encryption, Decryption, IOT, Real time sensory data*

Contents

List of Figures	10
Abbreviations	11
1 Introduction	12
1.1 Overview	12
1.2 Problem Statement.....	13
1.3 Proposed Solution.....	15
1.4 Working Principle.....	16
1.4.1 Sensor Actuation.....	16
1.4.2 ADC Conversion	17
1.4.3 Encryption of Data.....	17
1.4.4 Communication and Transfer of Data.....	17
1.4.5 Decryption	17
1.4.6 Presentation of Digital Data.....	18
1.5 Objectives	18
1.6 Scope	18
1.7 Deliverables	19
1.8 Relevant Sustainable Development Goals	19
1.9 Structure of Thesis.....	19
2 Literature Review	21
2.1 Introduction	21
2.1.1 How does cryptography work?	21
2.1.2 Cryptography vulnerabilities	22
2.2 Cryptography in IoT	22
2.2.1 Challenges with IoT security	23
2.2.2 IoT Security	24
2.2.3 Market Alternatives:	25
3 Design and Development	29
3.1 Network Diagram	29
3.2 Technical Specifications.....	31
3.2.1 ESP 32	31
3.2.2 IoT Sensors	32
3.2.3 Arduino I2C LCD	34
3.2.4 Power Source	35
3.2.5 IOT and real time Encryption/Decryption	36
3.2.6 Tools for Authentication and Encryption.....	36
3.3 Methodology.....	37
3.4 Stage 1: Actuating the Sensors with the ESP 32.....	37

3.5 Stage 2: ADC Conversion	38
3.6 Stage 3: Encryption.....	39
3.6 Stage 4: Decryption	40
3.7 Stage 5: Display and Use of Data	40
4 Code Analysis and Evaluation	41
4.1 Code for Sensor Actuation.....	41
4.1.1 ESP 32 Libraries	41
4.2 Receiver HEAM Device Code:.....	42
4.2.1 Connecting to Sender’s HAEM	42
4.2.2 Decryption of Received Data (AES 128-bit).....	43
4.2.3 Making HTTP Requests	44
4.2.4 Initializing HTTP Client	45
4.2.5 HTTP Response	46
4.2.6 Interfacing LCD.....	47
4.2.7 Output	48
4.3 Sender’s Side:.....	49
4.3.1 Encryption Algorithm (AES 128-bit)	49
4.3.2 Reading from Sensors.....	50
4.3.3 Communication	51
4.3.4 Data Transform for Encryption.....	52
4.3.5 Encrypting Sensory Data	53
4.3.6 Response to HTTP Requests.....	54
4.3.7 Output	55
5 Conclusion	56
6 Future Work	57
7 Personal Reflections.....	58
8 Appendix	59
8.1 Appendix 1: Synopsis	59
9 Bibliography.....	62
Plagiarisim Report.....	68

List of Figures

Figure 1.1 IOT in Smart Agriculture	13
Figure 1.2 Data Breach Statistics.....	14
Figure 1.3 Cost of Data breach	15
Figure 2.1 Basics of Cryptography	25
Figure 2.2 Tozny Platform.....	26
Figure 2.3 Hardware Security Module	27
Figure 2.4 E4 Model.....	28
Figure 3.1 Block Diagram	29
Figure 3.2 ESP 32 WROOM	31
Figure 3.3 DHT Sensor.....	34
Figure 3.4 I2C 16X2 LCD	35
Figure 3.5 Battery	35
Figure 3.6 DHT Sensor interfaced with ESP32	38
Figure 3.7 ADC Conversion	38
Figure 3.8 AES Implementation	39
Figure 3.9 Output Display	40

Abbreviations

Abbreviation	Meaning
IOT	Internet of things
AES	Advance Encryption Standard
HTTP	Hyper Text Transfer Protocol
Ack	Acknowledge
CIO	Chief Information Officer
MITM	Man in the Middle
ADC	Analog to digital conversion
BLE	Bluetooth low energy
DES	Data Encryption Standard
IDE	Integrated development environment
IDC	International Data Corporation
HSM	Hardware Secure Module

1 Introduction

This chapter discusses briefly, the problem statement, approach, and the scope of the project.

1.1 Overview

As organizations and security professionals adapt with the advancements in the IOT sector they face threats in securing their private and confidential data in short, their security is at risk. To cater these problems, cryptography comes into play [9][11][21]. Cryptography uses Ciphers to protect information and data, enabling only the authorized personnel to get the vital information. Security experts are convinced by the use of cryptography in IoT environments, saying it's an efficient way to secure data at rest and in motion, make the gateways safer that move the data and even provide authentication of the devices within the IoT environment, thereby providing utmost protection against hackers. The risk of unauthorized access to devices and the data they carry booms as soon as those devices connect to the internet. In IOT ecosystem al of the devices are connected with internet which makes it a major target. According to IoT Analytics, the number of active endpoints in the world in 2021 at 12.33 billion; more than 27.5 billion IoT connections are predicted to add up by 2025. Meanwhile, according to IDC researchers there will be 55.78 billion connected devices in the world by 2025, with around 77% of them connected to an IoT platform [27]. They further estimate that by 2025 those IoT devices will generate 73.1 zettabytes of data, up from 18.3 zettabytes in 2019. These Staggering numbers aren't the only security challenge. IoT established databases also increase penetration risks because their data exists in different places: on gateways, in centralized servers or in endpoint devices as well as in motion among all those endpoints. To tackle these calamities, we use cryptography [16]. We must ensure the security of the IOT data by merging cryptography in the IOT environment.

1.2 Problem Statement

Nowadays IOT (Internet of Things) is a major part of our lives. In the fields of smart agriculture, secure military setups and close spaces IOT sensors are used like temperature, humidity detectors etc. These sensors record real time data recordings and send them to the cloud/datacenter [2][4]. If data here is shared without encrypting, it this data can be intercepted by the middlemen and our security will be compromised. IOT sensors are resource and power constraint meaning low processing power and no suitable authentication/encryption mechanisms.

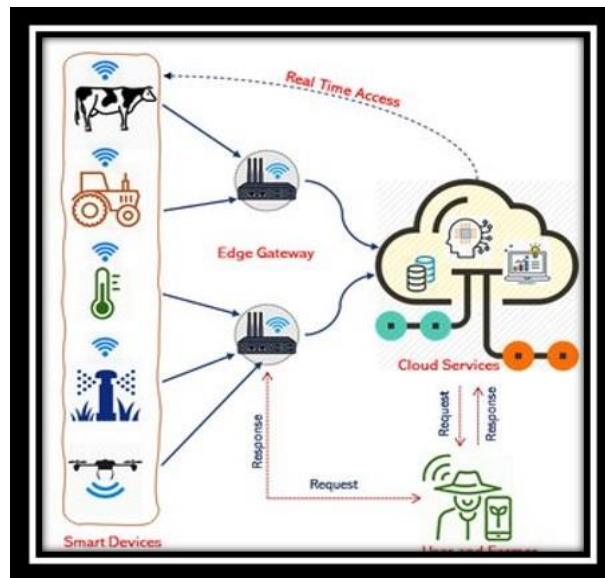


Figure 1.1 IOT in Smart Agriculture

Figure 1.1 depicts the implementation of IOT environment in the agriculture showing the importance of IOT in today's world where live sensory data from the farm fields is stored over the cloud. According to the report

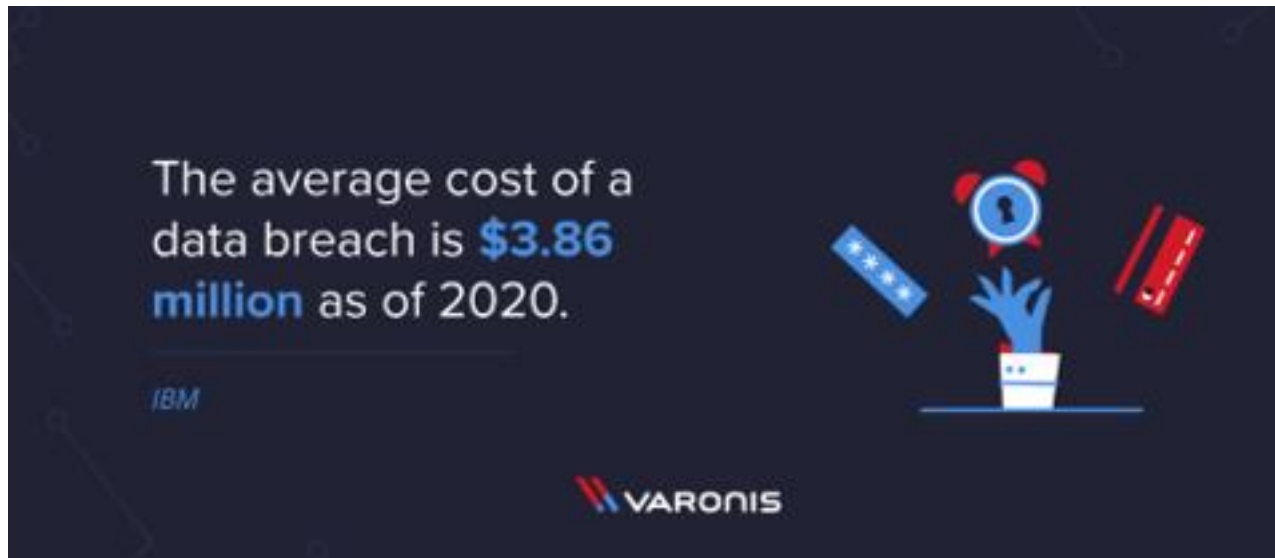


Figure 1.2 Data Breach Statistics

published by Palo Alto in the field of IOT 98 % of the data is unencrypted. By tapping into that unencrypted stream of data, attackers get in between devices or a device and the larger network which obviously means that attackers can perform Man in The Middle (MiTM) attacks and steal or alter the data. And yet whatever the state of its security the IoT is growing voraciously. McKinsey estimates that there will be 58.1 billion IoT devices connected to the internet by 2023. If current trends continue and 98 percent of IoT traffic is left unencrypted, it will be a feeding frenzy for cyber-criminals. The other alternatives in the market are mostly software based however when we consider real time sensory data these alternatives mostly encrypt data at the edge layer which means the data originating from the sensors is in unencrypted form and can be intercepted in the case of vulnerable gateways. Figures 1.2 and 1.3 show the common types of data breaches and their economic significance.

With the authentic encryption and secure mechanisms in our device, the Realtime data will be encrypted and sent at real time to the receiver device in this way the data will be encrypted prior to being shared at real time

Data breach incident	Reference
Unencrypted tapes containing the PHI of more than 19,000 patients were lost	(HHS, 2011)
Desktop computer with unencrypted data was stolen from Sutter Medical Foundations, Sacramento, CA.	(PRCH, 2012; Baker et al., 2011)
Unencrypted backup tapes were stolen from a car, exposing the health and personal information of more than 4.9 million patients.	(PRCH, 2012)
On August 25, 2009, a Blue Cross and Blue Shield laptop containing unencrypted ePHI of more than 800,000 physicians was stolen from an employee at the company's headquarter in Chicago	(OSF, 2012)
In 2003, a psychiatric Hospital in Aarhus (Denmark) sent an unencrypted email to a doctor, containing PHI. A Virus on the doctor's computer caused the email to be forwarded to unauthorized recipients.	(Kierkegaard, 2012)

Figure 1.3 Cost of Data breach

allowing safe traversal of data [13][19].

1.3 Proposed Solution

The following are the key goals that this project aims to achieve:

- Actuating IOT sensors with our HEAM device so we can detect real time data from the sensors.
- Relevant ADC conversion and suitable measures to make sensory data suitable for secure algorithms.

- Suitable encryption algorithms to cater data (Audio/Text/Video) and create cipher of data. The data will be sent securely receiver device where it will be decrypted for purpose use.
- To reduce data breach risks in the field of IOT at runtime where no authentication/encryption algorithms are present. Providing an easy to use and very cheap option.

1.4 Working Principle

The project mainly works on the principles of Cryptography i.e., Encryption/Decryption in securing the unencrypted data in IOT ecosystem. The project is divided into different modulus and every module is interwoven with the next module. The list of modules is as under:

- Sensor Actuation
- ADC Conversion of Sensory data
- Encryption of data (AES/SALSA)
- Communication and transfer of data to the receiver authenticated device (Real Time)
- Decryption at the receiver side
- Presentation of Digital data

1.4.1 Sensor Actuation

In this process IOT sensors for example temperature, humidity, DHT Sensor etc. are made usable by providing necessary power source and connection with the ESP32 device at the sender side. This allows the Sender device to get live readings from the sensors at real time.

1.4.2 ADC Conversion

The data from the IOT sensors is in analog form to make it usable in the encryption process we perform Analog to digital conversion of the sensory data through inbuilt mechanisms of the ESP32 Sender device. ESP32 module supports ADC with 18 channels of 12-bit SAR ADC and 2 channels of 8-bit DAC.

1.4.3 Encryption of Data

Appropriate encryption algorithms like AES/SALSA are used to encrypt the sensory data at run time allowing safe traversal of data attaining confidentiality.

1.4.4 Communication and Transfer of Data

The receiver HEAM device is authenticated by the sender device through HTTP request protocol and the data is sent to it at real time, the data is transferred in encrypted form. The receiver side sends HTTP requests to the server at equal time instances and the client side sends the IOT sensory data at real time. Http.h and Asyncwif.h libraries are used in Arduino IDE to follow this process.

1.4.5 Decryption

The receiver HEAM device decrypts the sensory data at real time so it can be used accordingly. The data is converted back from generated cipher to the plain text and then is presented for its use case.

1.4.6 Presentation of Digital Data

The decrypted data which shows the IOT sensory readings is displayed on the I2C 16X2 LCD connected to the receiver HEAM device. This completes the process with successful transfer of IOT Sensory data from sender to receiver side in Encrypted form.

1.5 Objectives

- To fetch data at real time from sensory devices.
- To perform fetched data encryption at runtime that is shared between the two devices connected through WIFI.
- To enable authentication mechanism capable of authorizing HAEM sending and receiving devices for performing encryption over the real-time data feeds.

1.6 Scope

This project might be used in various disciplines like smart agriculture, sensitive organizations, the healthcare sector, and the entire security sector of the country since it allows for the authentication and encryption/decryption of IOT Sensory data in real-time [23][17]. It will get the data from sensors at runtime, make the data suitable for secure mechanisms, encrypt the data with algorithms i.e., AES and share the Cipher with receiver HEAM device which in result decrypts the data and completes the process. Secure communication in military setup, patient live readings, real time crop levels in smart agriculture etc. all will be catered by our device. The project's primary purpose is to supplement and incorporate existing security tools while also providing Realtime operation. This project employs network security principles, including combining existing tools with our C code to authenticate and encrypt data. As a result, we hope to use this project to combine our theoretical expertise with practical experiences to improve our data security [28][32].

This project aims to create a complete security solution with its sensory data detection system and authentication/encryption mechanisms to securely communicate data at real time.

1.7 Deliverables

The following are the key goals that this project aims to achieve:

- Actuating IOT sensors with our HEAM device so we can detect real time data from the sensors.
- Relevant ADC conversion and suitable measures to make sensory data suitable for secure algorithms.
- Suitable encryption algorithms to cater data (Audio/Text/Video) and create cipher of data. The data will be sent securely receiver device where it will be decrypted for purpose use.
- To reduce data breach risks in the field of IOT at runtime where no authentication/encryption algorithms are present. Providing an easy to use and very cheap option.

1.8 Relevant Sustainable Development Goals

Our Project is relevant to the Sustainable Development Goal 9 Industry, Innovation, and infrastructure as it merges the use of Encryption/Decryption (Cryptography) and IOT industry to provide authentication and confidentiality in secure Communications departments and relevant Infrastructure.

1.9 Structure of Thesis

This project was created primarily to protect IOT sensory data against middlemen who can exploit the vulnerable gateways and alter the secure readings. Data is encrypted prior to it being shared as a result, safe traversal of data in the fields of IOT is made possible.

Chapter 2 contains the literature review and the background analysis of the project.

Chapter 3 contains the design and development of the project.

Chapter 4 contains detailed evaluation and analysis of the code.

Chapter 5 contains the conclusion of the project.

Chapter 6 Highlights the future work needed to be done for the commercialization of the project.

2 Literature Review

This chapter will go over the philosophy behind using Cryptography in IOT and some of its features and why it is used. The definition of a Cryptography will be compared to more official IT standards and some less formal ones to try to clarify what it is. Finally, the integration of the Cryptographic algorithms with already existing IOT smart sensors to effectively detect cyber threats and assist engineers in their efforts to protect personal privacy from being compromised and eroded.

2.1 Introduction

Cryptography is the study of utilizing math to encode and decode information or on the other hand it is the capacity to send data between members, in a damaged configuration, that keeps others from perusing it. it empowers you to store private data or communicate it across uncertain organizations (like the web) with the goal that it can't be used by anybody aside from the planned beneficiary.

2.1.1 How does cryptography work?

A cryptographic calculation, or code, is usually done for encrypting the message or any useful piece of information. A cryptographic calculation works by blending the plaintext with a key a word, number, or state to encode it. The equivalent plaintext encodes the various code text with various keys. the security of encrypted information depends completely on two things: the strength of the cryptographic calculation and the mystery of the key [14][1][25]. A cryptographic calculation, in addition to all possible keys and every one of the contract that make it work contain a cryptosystem.

2.1.2 Cryptography vulnerabilities

Currently cryptography exists at the combination of the sets of math, computer programming, electrical structure, resemblance in science, and material science. Applications of cryptography blend electronic exchange, chip-based portion cards, mechanized money related principles, personal computer passwords, and defence services correlation. There is this normal legend in the group of the web clients that cryptography is totally established. Revealing this truth, we express that it is not the situation [7][31]. There are sure number of risks in regard to quite possibly the most involved method for getting correspondences.

Vulnerabilities by which cryptographic systems get affected are: -

- Lifespan of keys
- Length of Public Key
- Length of Symmetric Key
- Secure storage of private keys
- Strength of the protocols of security
- Randomness of generated keys
- Strength of the security technology implementation
- Amount of plain text known to characters

2.2 Cryptography in IoT

Cryptography utilizes codes to safeguard data and interchanges, making it out of reach to everything except those approved to interpret the ciphers. Security pioneers advocate for its utilization in IoT conditions, claiming it's an ideal method for getting information very still and on the way, secure the channels that communicate

information and even confirm gadgets inside the IoT network, in this manner giving a sweeping of assurance against hacks [22][2][12].

2.2.1 Challenges with IoT security

Any electronic device that holds information can be compromised/penetrated, whether or not it's associated with the web. A troublemaker can take a PC and make use of the documents it holds, for instance. Be that as it may, the gamble of unapproved admittance to electronic gadgets and the information they hold peak when those gadgets interface with the web. IoT essentially grows that gamble of unapproved access basically because of the enormous number of gadgets being associated with the web.

That number is faltering. IoT Analytics, an IoT statistical surveying firm, determined the quantity of dynamic endpoints on the planet in 2021 at 12.38 billion; it predicts in excess of 27.1 billion IoT associations by 2025.

In the meantime, IDC analysts foresee that there will be 55.78 billion associated gadgets on the planet by 2025, with 76% of them associated with an IoT stage. They further gauge that those IoT gadgets will create 73.2 zettabytes of information by 2025, up from 18.2 zettabytes in 2019.

That huge volume isn't the main security challenge.

IoT organizations additionally increment hacking gambles in light of the fact that their information exists in better places: in endpoint gadgets, on entryways and in concentrated servers, as well as on the way among that multitude of focuses. Limiting those dangers is where cryptography comes in.

2.2.2 IoT Security

Encryption and mystery are outright prerequisites of IoT arrangements. They are utilized for getting correspondence, safeguarding firmware, and validation. As to, there are by and large three structures to consider:

Symmetric key encryption: Encryption and decoding keys are indistinguishable. RC5, DES, 3DES, and AES are types of symmetric key encryption. The phenomenon is depicted in figure 2.1.

Public Key encryption: The Encryption key is distributed openly for anybody to utilize and scramble information. Just the getting party has a confidential key used to decode the message. This is otherwise called uneven encryption. Uneven cryptography oversees information mystery, validates members, and powers non-renouncement. Understand web encryption and message conventions like Elliptic Curve, PGP, RSA, TLS, and S/MIME are viewed as open keys. Figure 2.1 shows the methodology and general concept.

Cryptographic hash: Maps information of an inconsistent size to a piece string (called the condensation). This hash work is intended to be "one way". Basically, the best way to reproduce the result hash is to compel each conceivable information blend (it can't be run backward). MD5, SHA1, SHA2, and SHA3 are types of one-way hashes. These are normally used to encode advanced marks, for example, marked firmware pictures, message validation codes (MAC), or confirmation. While encoding a short message like a secret word, the info might be excessively little to actually make a fair hash; all things considered, a salt or non-private string is affixed to the secret word to expand the entropy. A salt is a type of key determination work (KDF).

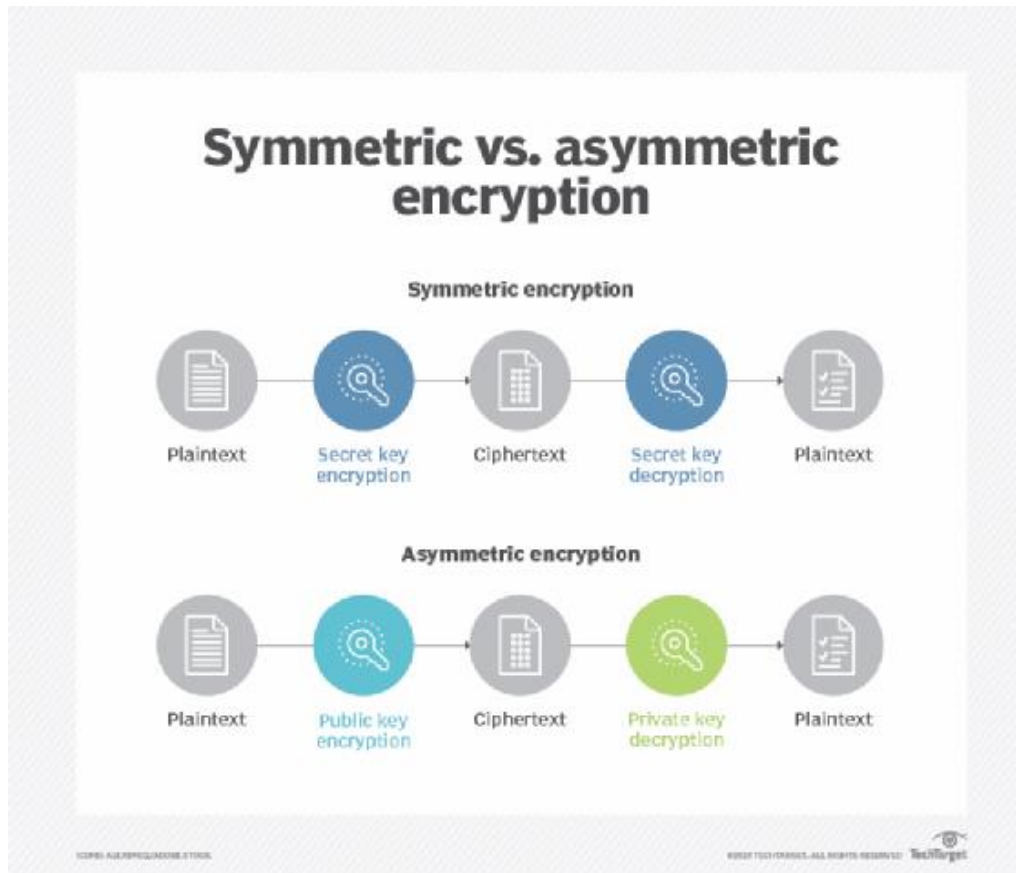


Figure 2.1 Basics of Cryptography

2.2.3 Market Alternatives:

There are a few software/hardware alternatives in the market that help in securing our precious data in the IOT ecosystems the details of these products are briefly summarized in the table below:

2.2.3.1 End to End Encryption Using Raspberry Pi (Tozny Platform)

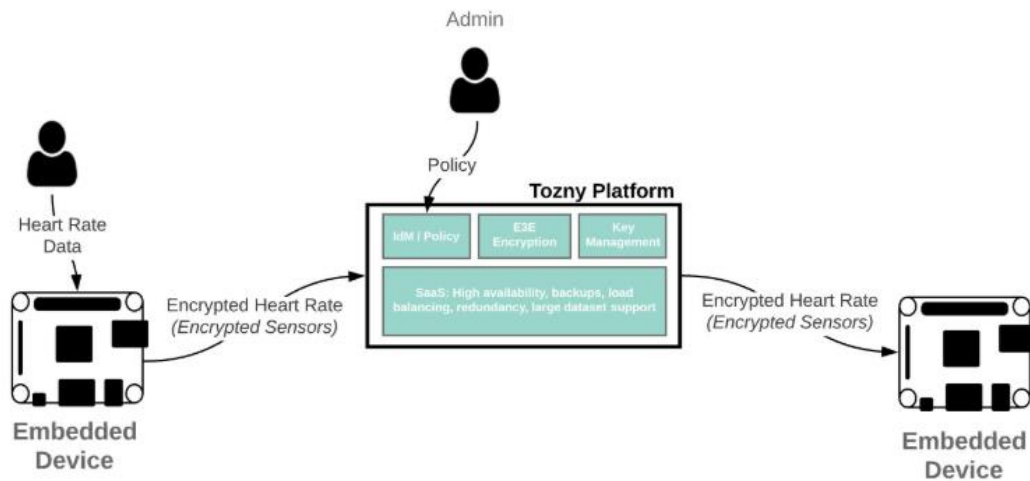


Figure 2.2 Tozny Platform

The figure 2.2 depicts end to end encryption and key establishment using raspberry pi. In the middle there is a Central identity manager which provides public/private (Asymmetric) key pairs to all of the sensors interfaced to the embedded device, the sensory data is end to end encrypted and the receiver device is authorized to display the encrypted sensory data [34].

2.2.3.2 Hardware Security Modules (HSM)

The figure 2.3 depicts the HSM products in the market which use the mechanisms of digital certificates for authentication with a signing request to the server and the use of asymmetric keys to get authorized for data

transfer and allow safe key sharing and data sharing in cases of digital transactions, bank transfers, confidential data sharing etc. [35].

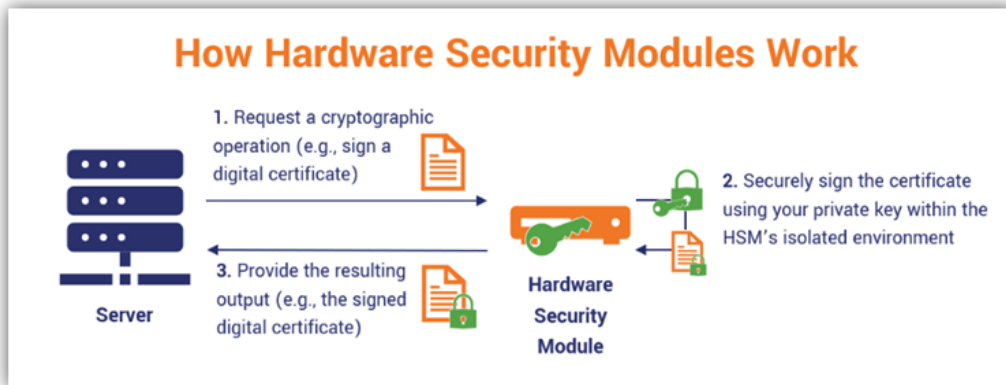


Figure 2.3 Hardware Security Module

2.2.3.3 E4

E4 is an open-source solution for IOT security. It has a dynamic key server to manage all the keys and for a specific use case the E4 solution uses a static key. As shown in the figure the E4 solution works in the background efficiently and is deployable over multiple architectures with functionality on most of the IOT sensors [36].

The E4 solution use case is shown in Figure 2.3.

Use Cases

E4 supports both "many-to-one" and "many-to-many" IoT architectures, and is suitable for various applications such as automotive, smart city/building, healthcare, or access control.

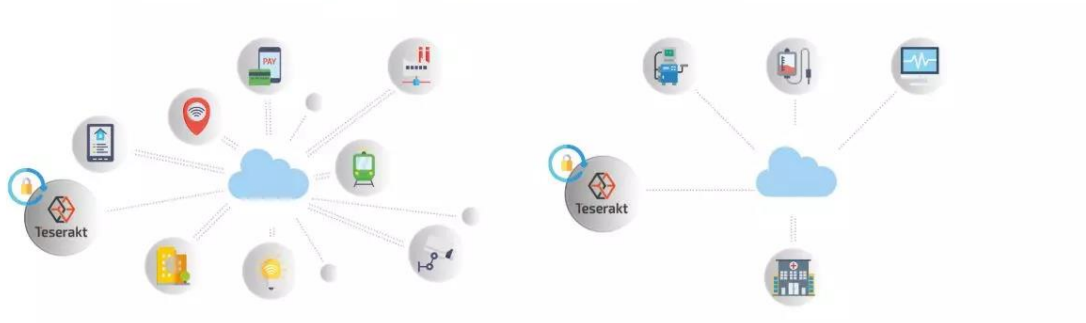


Figure 2.4 E4 Model

3 Design and Development

This chapter covers the project's technical requirements as well as the specifics of the project's key steps as it progresses.

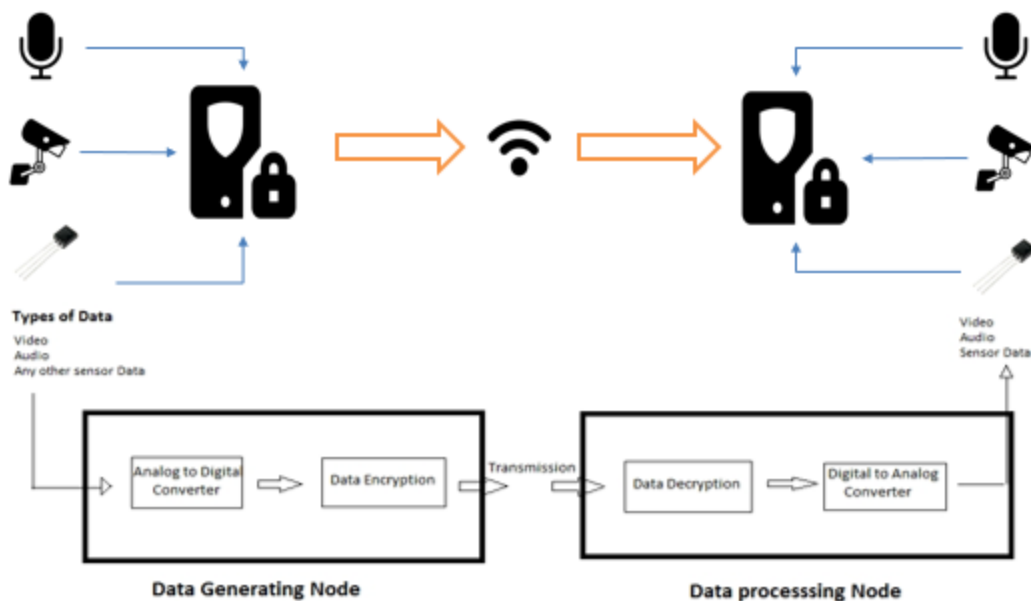


Figure 3.1 Block Diagram

3.1 Network Diagram

The network design is depicted in figure 3.1; In the model there will be a client side connected to the IOT sensors and will convert the data from analog to digital form to make it compatible for encryption and then suitable encryption algorithms are applied sending the data from client side to receiver side, which is also a HAEM device that performs the opposite function to the client side as it decrypts the data and displays the data in the appropriate way. This process ensures that the data is encrypted, and our device is used as a gateway

for secure transmission of data between the sender and receiver. In the figure 3.1 it is depicted clearly how the sender HEAM device carries with it all the sensors and with ADC conversion and encryption algorithms we achieve end to end encryption of IOT sensors in real time.

3.2 Technical Specifications

The below are the technical requirements for the hardware and software equipment used:

3.2.1 ESP 32

ESP32 WROOM is a variant of Arduino by Espressif with the Wi-Fi and Bluetooth technology in built in and used for many IOT applications. The module has the ESP32-D0WDQ6 chip which is both scalable and adaptive. It's user friendly, low cost and perfect for Internet of things projects. Use of Arduino IDE to code the device in C++ Programming language.

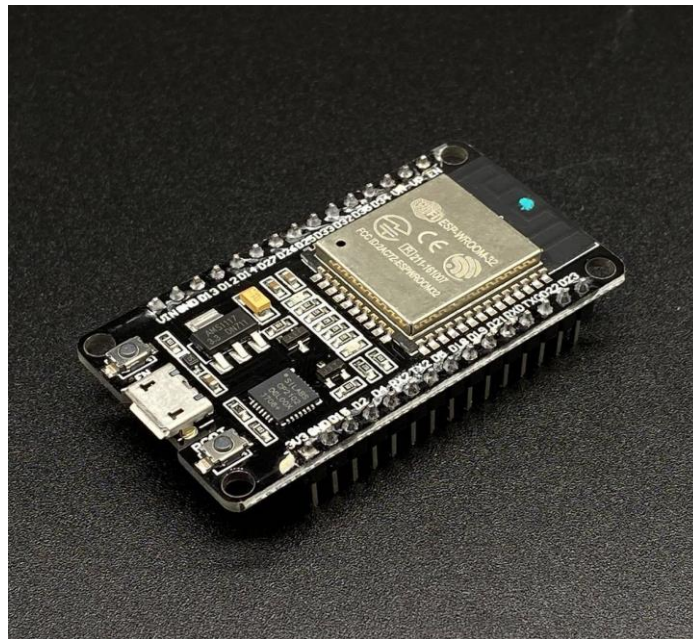


Figure 3.2 ESP 32 WROOM

Features:

- The ESP32 Wroom has a 32-bit Single or Dual core LX6 Microprocessor with clock frequency < 240 MHz
- The module has 448 KB of ROM, 16 KB of RTC SRAM. and 520 KB of SRAM

- Supports Wi-Fi
- Supports Bluetooth
- 34 Programmable General-purpose inputs/outputs
- The module supports ADC with 18 channels of 12-bit SAR ADC and 2 channels of 8-bit DAC
- Serial Connectivity included
- Ethernet MAC for physical LAN Comms
- Safe Boot and Flash Encryption
- Cryptographic Hardware Support for AES, Hash (SHA-2), RSA, ECC and RNG.

3.2.2 IoT Sensors

There are various varieties of IoT Sensors within the IOT industry consistent with the employment case. The IoT and its counterpart, the commercial Internet of Things (IIoT), are bringing the sensor tech to the following level.

Sensors are devices that detect and answer changes in an environment. Changes can come from a range of sources like temperature, motion, light, and pressure. Sensors are connected to displays or databases where they exhibit the live information regarding the environmental variable data. Sensors are available all shapes and sizes. Some are purpose-specific containing many built-in mini/individual sensors, enabling you to live and monitor many sources of information [33]. It's vital for sensors to incorporate digital and analog inputs in order that they will read data from legacy sensors in brownfield environments.

Various IoT sensors in the market:

- Temperature
- Humidity
- Pressure
- Level
- Accelerometers
- Gyroscope
- Gas
- Infrared
- Optical

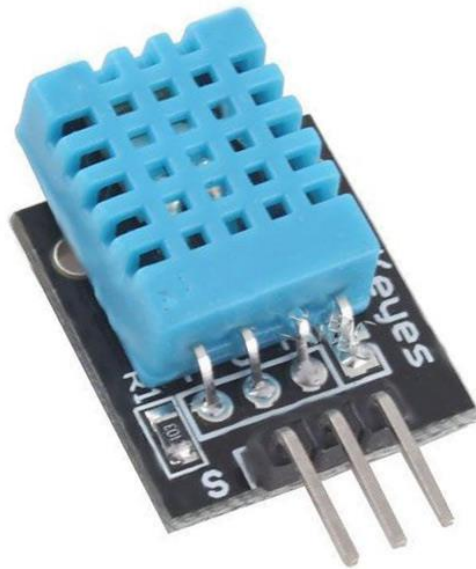


Figure 3.3 DHT Sensor

3.2.3 Arduino I2C LCD

Features:

- In the I2C LCD we have 16x2 display
- High contrast
- Wide viewing angle
- The LCD needs 5V DC voltage
- Display type: Characters
- Dimensions: 80mm x 35mm x 11mm
- Commonly used in Networking equipment, IOT projects, fax machines.



Figure 3.4 I2C 16X2 LCD

3.2.4 Power Source



Figure 3.5 Battery

Mini 3V 3.3V 3.7V 5V 9V to 12V Step Up Boost Converter Module Voltage Regulator

3.2.5 IOT and real time Encryption/Decryption

The data from these IOT sensors has to be encrypted and traversed in a secure form also at real time which requires proper analog to digital conversion and authentic encryption algorithms. All these issues will be catered by our project.

3.2.6 Tools for Authentication and Encryption

Different tools which were used:

- ESP 32 WROOM Hardware
- IOT Sensors
- Arduino IDE
- LCD and display mechanisms

3.3 Methodology

Our Solution is a black box easy to operate device which is simply an ESP 32 board at the Sender side and on the corresponding receiver side. The IOT Sensors are actuated with the Sender side ESP 32 which senses the data in analog form and converts it to the digital form at run time. This digital data which is in the text form is encrypted using suitable cryptographic algorithms in our case AES. Now this data is sent at runtime to the receiving HEAM device which decrypts the data and makes it usable and shows it on the output display i.e., LCD. This process ensures the secure traversal of data at run time from sender to receiver side accommodating the IOT data which is usually not encrypted at runtime and resource constrained specifications of the IOT sensors don't allow them to incorporate any algorithms within their functionality. The use of C language coding in Arduino IDE and inbuilt libraries will allow us to complete this process.

3.4 Stage 1: Actuating the Sensors with the ESP 32

The first and the most important step is to actuate /attach the sensors with our sender side ESP32. The process is simply done by integrating the sensors with the ESP 32 board and run the usual library and c code for it which will allow us to get the IOT sensory data at real time in analog form.

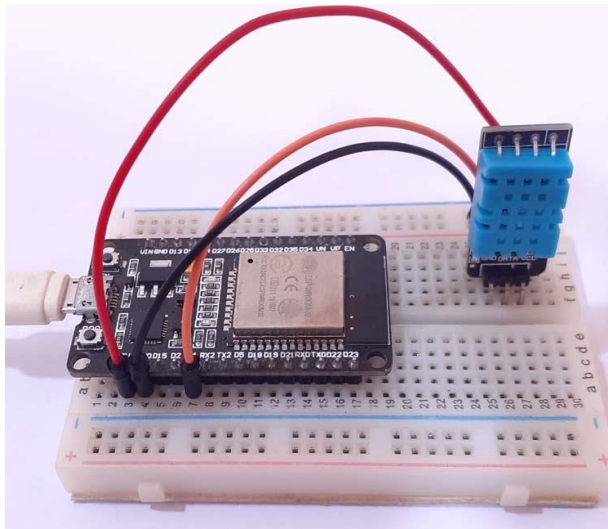


Figure 3.6 DHT Sensor interfaced with ESP32

3.5 Stage 2: ADC Conversion

The corresponding data picked up from the IOT Sensors is converted to digital form by the ESP 32 board. The ESP 32 has its's inbuilt ADC converter mechanism which allows us to translate data from analog to digital

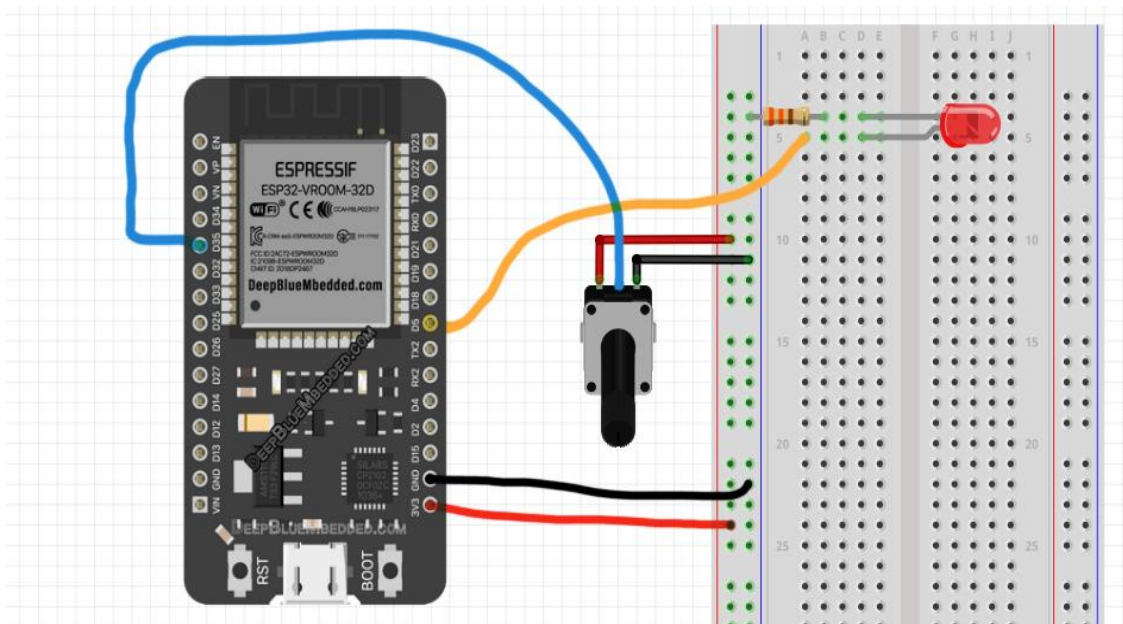


Figure 3.7 ADC Conversion

form in real time making it usable for encryption and data traversal. ESP32 module supports ADC with 18 channels of 12-bit SAR ADC and 2 channels of 8-bit DAC as shown in the figure 3.7.

3.6 Stage 3: Encryption

The data is encrypted by AES (Advanced Encryption Standard) EBC mode, with the relevant coding done to attain encryption through AES generating a cipher from the standard textual data from the IOT Sensors attached with the ESP 32 Board. This Cipher is generated and then sent over to the client side where the plain

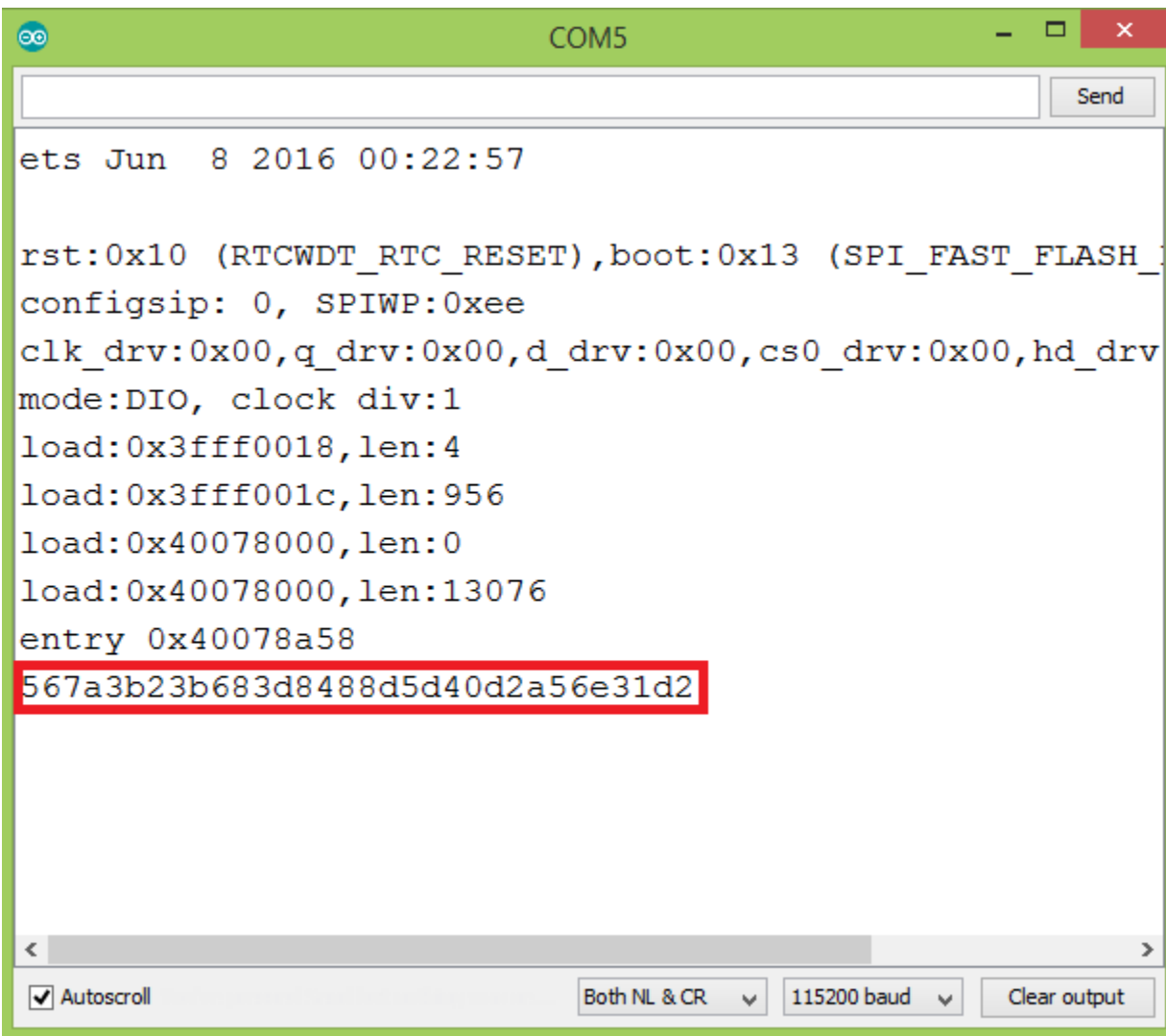


Figure 3.8 AES Implementation

text is regenerated using decryption. Figure 3.8 depicts the encryption process.

3.6 Stage 4: Decryption

The cipher text coming through from the sender ESP 32 board authenticates the Receiver ESP 32 device and heads over to the module for decryption. The reverse process of Encryption takes place with relevant coding and the corresponding plain text from the cipher text is generated.

3.7 Stage 5: Display and Use of Data

The plain text is then used by the receiver ESP 32 device to display the results on the I2C LCD. Which gives the relevant results. In our case the DHT Sensor temperature and humidity live readings are incorporated generating the readings cipher and safe transfer of that information which is then decrypted and shown on the LCD Display. As shown in figure 3.9.



Figure 3.9 Output Display

4 Code Analysis and Evaluation

4.1 Code for Sensor Actuation

4.1.1 ESP 32 Libraries

- **Adafruit_BMP085.h**

This library is used to interface the pressure sensor (Barometric Sensor BMP180) with ESP32. We can get Live readings of temperature, pressure, altitude, and sea level pressure.

- **DHT.h**

This library is used to interface the DHT Sensor (Digital Humidity and Temperature Sensor) with ESP 32. We can get live readings of temperature and humidity.

- **Wifi.h**

This library is used to communicate two ESP32 devices.

- **AsyncWebServer.h**

This library is used to initialize the server we are using at the sender's side of our channel.

- **HTTPClient.h**

This library is used on the receiver HEAM device to make HTTP requests on the server to get live digital data from the sender device.

- **LiquidCrystal.h**

This library is used to interface the I2C 16X2 LCD with ESP32 receiver device to display the digital data we are getting from IOT sensors (Sender HEAM Device).

4.2 Receiver HEAM Device Code:

4.2.1 Connecting to Sender's HAEM

This HAEM device is at the receiver end. It uses the code listed below to send HTTP requests to the sender side to get data of digital IoT sensors. After every defined instance in our case 1 second for digital data.

- We are connecting to the sender side by using the credentials given by the sender side.
- With the data received it is decrypted accordingly using AES in case of digital data.
- LCD is interfaced with this device and data is displayed on the 16X2 LCD.

4.2.2 Decryption of Received Data (AES 128-bit)

In this part of code decryption is implemented over the data this receiver HAEM is getting from the sender's side.

- We are using AES-128-bit encryption algorithm for digital data using the symmetric key cryptography.
- Key is static and already shared on both HAEM devices but for further security if we change the key to be not static, we can implement key sharing mechanisms over this device.

As shown in below screenshot of code.

```
client_dht
|
#include <WiFi.h>
#include <HTTPClient.h>
#include "mbedtls/aes.h"

const char* ssid = "ESPserver-access-point";
const char* password = "123456789";

//Your IP address or domain name with URL path
const char* serverNameTemp = "http://192.168.4.1/temperature";
const char* serverNameHumi = "http://192.168.4.1/humidity";

#include <Wire.h>

//#include <LiquidCrystal_I2C.h>
//
//LiquidCrystal_I2C lcd(0x27, 16, 2);

String temperature;
String humidity;
unsigned long previous_millis = 0;
unsigned long interval = 10000;
//unsigned char cipher_buffer[16];
char * key = "abcdefghijklmnop";
unsigned char output_val[16];

// HTTP request function
void decrypt(unsigned char * cipherText, char * key, unsigned char * outputBuffer){
```

4.2.3 Making HTTP Requests

- In this part of code, we are making HTTP requests from the receiver's side HAEM to the sender's side's HAEM to get the data of sensors actuated with that device.
- In our case we are getting data every second. This Interval can be set to any value as required.

As shown in below screenshot of code.

```
client_uart
//// for(;;); // Don't proceed, loop forever
//// }
// lcd.clear();
//// lcd.textcolor(WHITE);

WiFi.begin(ssid, password);
Serial.println("Connecting");
while(WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
Serial.println("");
Serial.print("Connected to WiFi network with IP Address: ");
Serial.println(WiFi.localIP());
}

void loop() {

  unsigned long currentMillis = millis();

  if(currentMillis - previous_millis >= interval){
    // Check WiFi connection status

    if(WiFi.status() == WL_CONNECTED ){
      delay(1000);
      temperature = httpGETRequest(serverNameTemp);
      delay(1000);
      humidity = httpGETRequest(serverNameHumi);
```

4.2.4 Initializing HTTP Client

We have initialized HTTP client here to make HTTP requests on the HTTP server to get the data, as shown in below screenshot of code.

```
//      lcd.print(" "); //
//
//
//      lcd.display();

      // save the last HTTP GET Request
      previous_millis = currentMillis;
    }else{

      Serial.println("Wifi not Connected!");

    }
  }
}

String httpGETRequest(const char* serverName) {
  WiFiClient client;
  HTTPClient http;

  // Your Domain name with URL path or IP address with path
  http.begin(client, serverName);

  // Send HTTP POST request
  int httpResponseCode = http.GET();

  String payload = "";

  if (httpResponseCode>0) {
    //Serial.print("HTTP Response code: ");
    //Serial.println(httpResponseCode);
  }
}
```

4.2.5 HTTP Response

In below screenshot of code, we are getting HTTP response from sender's HAEM.

```
String payload = "";  
  
if (httpResponseCode>0) {  
    //Serial.print("HTTP Response code: ");  
    //Serial.println(httpResponseCode);  
    payload = http.getString();  
    Serial.println("\n\nCiphared text: \n");  
    Serial.println(payload);  
    delay(500);  
    unsigned char* cipher_buffer = (unsigned char*) payload.c_str();  
  
    //function call to decryption function  
  
    decrypt(cipher_buffer,key,output_val);  
  
    Serial.println("\n\nDeciphered text:");  
    String plain = "";  
    for (int i = 0; i < 16; i++) {  
        Serial.print((char)output_val[i]);  
        plain = plain + (char)output_val[i];  
    }  
  
    http.end();  
    return String(plain);  
}  
else {  
    Serial.print("Error code: ");  
    Serial.println(httpResponseCode);  
}
```

4.2.6 Interfacing LCD

```
        http.end();
        return String(plain);
    }
    else {
        Serial.print("Error code: ");
        Serial.println(httpResponseCode);
        delay(1000);
    }
}
```

In this part of code, we are interfacing LCD with our device to get decrypted data visible to the user for any use case.

```
    Serial.print
```

```
Serial.println("\nTemperature Plain: " + temperature + "*C");
Serial.println("\nHumidity Plain: " + humidity );

//    lcd.clear();
//
//    // lcd temperature
////    lcd.setTextSize(2);
////    lcd.setTextColor(WHITE);
//    lcd.setCursor(0,0);
//    lcd.print("T: ");
//    lcd.print(temperature);
//    lcd.print(" ");
//
//
//
//
//    lcd.print("C");
//
//    // lcd humidity
//
//    lcd.setCursor(0, 1);
//    lcd.print("H: ");
//    lcd.print(humidity);
//    lcd.print(" %");
//
//
//    lcd.display();
```

4.2.7 Output

The Display here shows the Live readings of temperature from the DHT Sensor actuated with the sender's HAEM Device. With the credentials the sender side data can be accessed on the fly. Here the digital data is decrypted and then shown. While we receive the cipher text from the sender side data is in encrypted form it is decrypted and used accordingly.

In below picture we are getting data from our sender's device about the current temperature.



4.3 Sender's Side:

4.3.1 Encryption Algorithm (AES 128-bit)

All the IoT sensors are interfaced with the HAEM Sender device. It follows the following code to get the data from the sensors and encrypt it through AES 128-bit EBC mode in case of digital data. It uses a 128-bit static symmetric key. It handles the HTTP requests from the Receiver side sending the appropriate data in response at real time.

```
server_dht
#include "WiFi.h"
#include <ESPAsyncWebServer.h>
#include <Wire.h>
#include <DHT.h>
#include <DHT_U.h>
#include "mbedtls/aes.h"

DHT dht2(2,DHT11);

const char* ssid ="ESPserver-access-point";
const char* password ="123456789";

AsyncWebServer server(80);

void encrypt(char * plainText, char * key, unsigned char * outputBuffer){

    mbedtls_aes_context aes;

    mbedtls_aes_init( &aes );
    mbedtls_aes_setkey_enc( &aes, (const unsigned char*) key, strlen(key) * 8 );
    mbedtls_aes_crypt_ecb( &aes, MBEDTLS_AES_ENCRYPT, (const unsigned char*)plainText, outputBuffer);
    mbedtls_aes_free( &aes );
}

String readTemp() {

    char * key = "abcdefghijklmnop";
    float temp = dht2.readTemperature();
```

4.3.2 Reading from Sensors

We are getting the data from sensors in this part of code and initializing the credentials for encryption algorithm we have just defined in this above code.

```
String readTemp() {

char * key = "abcdefghijklmnop";
float temp = dht2.readTemperature();
if(isnan(temp)){
temp = 27;
}
Serial.println(temp);
char tempch[16];
dtostrf(temp, 16, 13, tempch);
Serial.println(tempch);
unsigned char outputtemp[16];

//function call for encrypting data

encrypt(tempch,key,outputtemp);
Serial.println("\nTemperature plain text:");
Serial.println(temp);
String out = "";
Serial.println("\nTemperature Ciphared text:");
for (int i = 0; i < 16; i++) {

char str[3];

sprintf(str, "%02x", (int)outputtemp[i]);
Serial.print(str);
out = out+str;
}
}
```

4.3.3 Communication

- We are initializing our server in this part of code using our sender's side credentials.
- Our server is the responding to the HTTP requests made from the receiver's side device.

```
void setup()
{

  Serial.begin(115200);
  while (!Serial) {}
  Serial.println();

  Serial.print("Setting access point...");
  WiFi.softAP(ssid,password);
  IPAddress IP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(IP);
  Serial.begin(115200);
  Serial.println("Temperature Cipher: " + readTemp());
  Serial.println("Humidity Cipher: " + readHum());

  //http request for temperature responded here

//tempch,key,outputtemp
server.on("/temperature", HTTP_GET, [] (AsyncWebServerRequest *request){
  request->send_P(200, "text/plain", readTemp().c_str() );
});

  //http request for temperature responded here

//humich,key,outputhumi
```

4.3.4 Data Transform for Encryption

In below screenshot of code, we are transforming the data, we are getting from sensors, for our encryption purposes.

```
for (int i = 0; i < 16; i++) {  
  
    char str[3];  
  
    sprintf(str, "%02x", (int)outputtemp[i]);  
    Serial.print(str);  
    out = out+str;  
}  
return String(out);  
  
}  
  
String readHum() {  
    char * key = "abcdefghijklmnop";  
    float humi = dht2.readHumidity();  
    if(isnan(humi)){  
        humi = 56;  
    }  
    // Serial.println(humi);  
    char humich[16];  
    dtostrf(humi, 16, 13, humich);  
    // Serial.println(humich);  
    unsigned char outputhumi[16];  
    delay(1000);  
    //function call for encrypting data
```

4.3.5 Encrypting Sensory Data

In this below screenshot of code, we are getting data transformed and then we are calling our encryption function for encrypting the data.

```
    }  
    // Serial.println(humi);  
    char humich[16];  
    dtostrf(humi, 16, 13, humich);  
    // Serial.println(humich);  
    unsigned char outputhumi[16];  
    delay(1000);  
    //function call for encrypting data  
  
    Serial.println("\nHumidity plain text:");  
    Serial.println(humich);  
    encrypt(humich, key, outputhumi);  
    String out = "";  
    Serial.println("\nCiphered text:");  
    for (int i = 0; i < 16; i++) {  
  
        char str[3];  
  
        sprintf(str, "%02x", (int)outputhumi[i]);  
        Serial.print(str);  
        out = out+str;  
    }  
    return String(out);  
    delay(2000);  
}  
  
    //function definition for encryption function
```

4.3.6 Response to HTTP Requests

In below picture of our code

- Our server is the responding to the HTTP requests made from the receiver's side device.
- We are getting values from sensors by calling the functions from above part of code and sending that data in response to the HTTP requests.
- In void loop () we are printing that data on our local machine for testing purpose.

```
        //http request for temperature responded here

//tempch,key,outputtemp
server.on("/temperature", HTTP_GET, [] (AsyncWebServerRequest *request){
    request->send_P(200, "text/plain", readTemp().c_str() );
});

        //http request for temperature responded here

//humich,key,outputhumi
server.on("/humidity", HTTP_GET, [] (AsyncWebServerRequest *request){
    request->send(200, "text/plain", readHum().c_str());
});

        // Start server

server.begin();

}

void loop()
{
    delay(2000);
    Serial.println(readTemp());
    delay(1000);
    Serial.println(readHum());
    Serial.flush();
}
```

4.3.7 Output

- We are getting data from sensors on internet and on our local machine as well.
- We can access that data through any device if we know the credentials of sender's device.
- We can use our browser too to get that data if we know the credentials of network built by sender's side HAEM.
- This is best for indoor communication like for home automation and security purposes.
- In future we can scale it to large distances as well.

5 Conclusion

As the world gets smarter and smarter with every passing day it will not be long until most of our routine and living is virtualized with the addition of Internet of things in all aspects of human life. With IOT sensors and devices dominating the technology around us it's important to secure our data and keep it unrecognizable for people who may mean any harm to us. The project serves the IOT devices that are resource constrained and don't provide any viable encryption and authentication mechanism to secure the data. The sender Device senses data at real time encrypting and sending the data in encrypted form to the receiver side where it is decrypted and used accordingly. The Service is very important in Military organizations where secure comms are important or at hospitals where patient details must be kept private and live readings have to be monitored, also in the domains of smart agriculture and departments where important decisions have to be taken according to the IOT data so we need to Secure this data at real time to serve our purpose. The software alternatives in the market usually encrypt at the edge layer which can be dangerous if our gateways are vulnerable, so we have to encrypt the data prior to it being shared which we have achieved by our project.

We have discussed a solution for dealing with IOT data encryption at real time. As a result, the initiative is a massive support to researchers and network security experts. This information may strengthen the security of IOT devices and identify vulnerabilities within these Smart devices. We must continue our studies to protect ourselves from current and new future threats as we evolve.

6 Future Work

The project can be modified and incorporated further in the future for better outcomes. Our Project currently incorporates text data at run time from the IOT sensors. In future we'd like to incorporate further audio and video data, which will allow more flexible use cases and more security in smart homes and the smart environments.

The Different cryptographic algorithms can also be used for different purposes as it is suitable for heavier processing with audio and video data, we can use the device in purposes like secure Live footage and audio secure communications.

Another way of improving it could be to train it also with different authentication mechanisms like EAP flavors. Authentication on a larger scale can allow the HEAM device to locate and interact with its respective receiver device and transfer the data in secure form. For now we are using EBC mode of AES to attain encryption because it is efficient in regards of probabilistic encryption however in future CBC mode can also be incorporated. Improvement in processing speed, memory data types all can be benefited from in the future.

7 Personal Reflections

The project was chosen in the hope of pursuing career development in networking and network security after the completion of our degree. The field has remained our area of interest for a long time. Although dealing with the implementation of the Encryption Algorithms has been confusing and overwhelming at times, Encrypting the data at real time would help make better career decisions in the future. Moreover, we believe our choice of undertaking the project was the correct one and shortly may even revolutionize network security not only in Pakistan but all over the globe.

Initially, the project seemed like something that would be highly interesting, but after commencing our work on the project, it did take us to the realization that it was a lot more complicated than initially thought. Even though it was difficult, but just like any beautiful ending leaves behind a trail of difficulties, the work carried on the project was as enjoyable as the project's results were, and that is what kept us going.

In terms of personal development, we have gained a better understanding of the process of completing a project of this magnitude and have a better understanding of the kind of work that goes into such a project, and we are fully prepared to do large-scale documentation and consulting work in the future that could require this magnitude of data.

8 Appendix

8.1 Appendix 1: Synopsis

Extended Title: Hardware Authentication And Encryption Module (HEAM)

Brief Description of The Project / Thesis with Salient Specifications:

IOT sensors are resource and power constraint. They do not have encryption mechanisms on them and encryption is usually done on the edge layer, so if the gateways are vulnerable data can be breached/tampered etc. Our project's goal is to encrypt the data prior to it being shared.

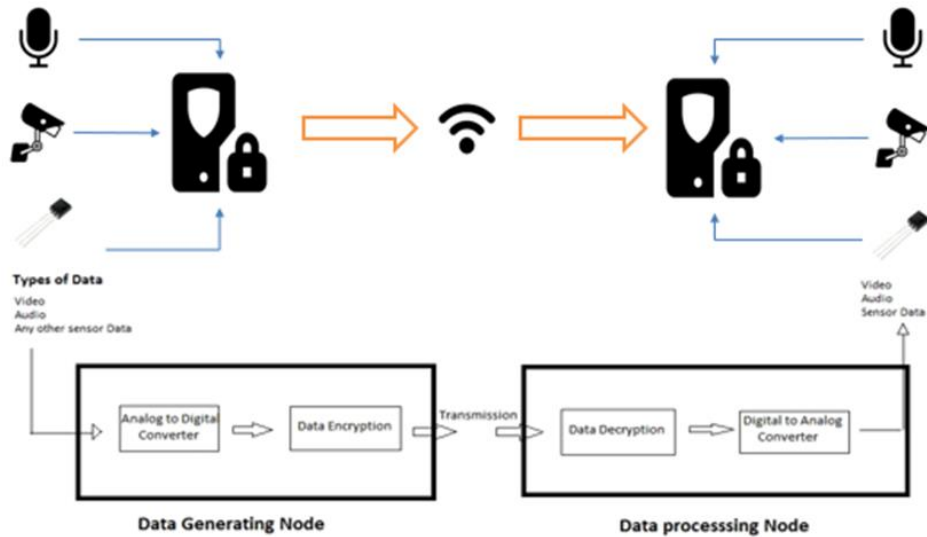
In this project, we will build a hardware based authentication and encryption module, which will take live data from sensors and will encrypt that data on the real-time. We will design a hardware module that will authenticate the other hardware module and allow the encrypted data transmission. Input data can be audio/video/real time sensor data and will be encrypted prior to transmission. Only the authenticated HAEM device will be able to decrypt data using pre-shared secret key.

We will have two hardware modules one will be at data generating node and the other at data processing node the module that encrypt the data send the data to the receiving end where it is decrypted by the other authenticated module.

The project is further subdivided into three distinct phases:

1. **Actuation of IOT Sensors:** In this phase the IOT Sensors are attached with the sending HEAM device (ESP32). Specific measures are taken to make the IOT sensors operable and their Data usable.
2. **Encryption of data:** Appropriate Encryption Algorithms are used to Encrypt the IOT Sensory Digital Data i.e., AES/SALSA etc.
3. **Decryption of data:** In this Phase the authenticated receiver HEAM device gets the digital data decrypts it accordingly and displays it on the 16x2 LCD or Speaker for Audio data.

Diagram



Scope of Work:

For development of HEAM we will be requiring sound knowledge of:

- Light Cryptographic Algorithms (AES/SALSA)
- Arduino IDE
- IOT Sensors Interfacing
- Wi-Fi/HTTP Libraries

We will be requiring hands-on expertise of:

- Arduino Coding
- Implementation of Cryptographic Algorithms
- Sensor Actuation / Libraries

Academic Objectives:

- Collection of detailed data about Cryptography and IOT in real-time
- Providing authentication and confidentiality in IOT Ecosystems.

Application / End Goal Objectives:

To Facilitate authentication and confidentiality of IOT sensory data at real time in closed Secure setups i.e. Military organizations, hospitals and Smart agriculture arenas etc.

Previous Work Done on The Subject:**Hardware Security module:**

HSM devices use latest technology of cryptography and digital signatures to provide relevant/efficient confidentiality and authentication in use cases such as passwords, credentials etc. Usable when connected to OS.

Attribute-Based Encryption on IoT Devices

The creation of ABE cipher is slow when executed on resource constraint devices, such as IoT sensors but for sharing the cipher text with a group of users it is suitable.

ATECC608A Software Alternative

The implementation (Software) of the efficient algorithm for IoT devices which are resource constraint shows up double efficiency in the encryption of (16 B) of plain text, in comparison to the currently used standard AES-128.

Material Resources Required: ESP32 Arduino Wroom, Battery/Power bank, IOT Sensors, LCD, Speaker

Special Skills Required:

- **Development:** Arduino Coding, IDE
- **OS:** Windows
- **Domains:** IOT, Cryptography, Network Security

9 Bibliography

- [1] H. Garg and M. Dave, "Securing IoT Devices and Securely Connecting the Dots Using REST API and Middleware," 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), 2019, pp. 1-6, Doi: 10.1109/IoT-SIU.2019.8777334.
- [2] A. K. Gupta and R. Johari, "IOT based Electrical Device Surveillance and Control System," 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), 2019, pp. 1-5, doi: 10.1109/IoT-SIU.2019.8777342.
- [3] E. P. Yadav, E. A. Mittal and H. Yadav, "IoT: Challenges and Issues in Indian Perspective," 2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU), 2018, pp. 1-5, doi: 10.1109/IoT-SIU.2018.8519869.
- [4] V. S. A. A. A. Don, S. W. Loke and A. Zaslavsky, "IoT-Aided Charity: An Excess Food Redistribution Framework," 2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU), 2018, pp. 1-6, doi: 10.1109/IoT-SIU.2018.8519856.
- [5] D. K. Aagri and A. Bisht, "Export and Import of Renewable energy by Hybrid MicroGrid via IoT," 2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU), 2018, pp. 1-4, doi: 10.1109/IoT-SIU.2018.8519873.
- [6] E. R. Naru, H. Saini and M. Sharma, "A recent review on lightweight cryptography in IoT," 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2017, pp. 887-890, doi: 10.1109/I-SMAC.2017.8058307.

- [7] P. Saraswat, K. Garg, R. Tripathi and A. Agarwal, "Encryption Algorithm Based on Neural Network," 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), 2019, pp. 1-5, doi: 10.1109/IoT-SIU.2019.8777637.
- [8] S. Ganiev and Z. Khudoykulov, "Lightweight Cryptography Algorithms for IoT Devices: Open issues and challenges," 2021 International Conference on Information Science and Communications Technologies (ICISCT), 2021, pp. 01-04, doi: 10.1109/ICISCT52966.2021.9670281.
- [9] N. A. Gunathilake, A. Al-Dubai and W. J. Buchana, "Recent Advances and Trends in Lightweight Cryptography for IoT Security," 2020 16th International Conference on Network and Service Management (CNSM), 2020, pp. 1-5, doi: 10.23919/CNSM50824.2020.9269083.
- [10] C. G. C. Carducci, A. Monti, M. H. Schraven, M. Schumacher and D. Mueller, "Enabling ESP32-based IoT Applications in Building Automation Systems," 2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0&IoT), 2019, pp. 306-311, doi: 10.1109/METROI4.2019.8792852.
- [11] N. Nikolov and O. Nakov, "Research of Secure Communication of Esp32 IoT Embedded System to.NET Core Cloud Structure using MQTTS SSL/TLS," 2019 IEEE XXVIII International Scientific Conference Electronics (ET), 2019, pp. 1-4, doi: 10.1109/ET.2019.8878636.

- [12] R. Podder and R. K. Barai, "Hybrid Encryption Algorithm for the Data Security of ESP32 based IoT-enabled Robots," 2021 Innovations in Energy Management and Renewable Resources(52042), 2021, pp. 1-5, doi: 10.1109/IEMRE52042.2021.9386824.
- [13] O. Barybin, E. Zaitseva and V. Brazhnyi, "Testing the Security ESP32 Internet of Things Devices," 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), 2019, pp. 143-146, doi: 10.1109/PICST47496.2019.9061269.
- [14] F. B. Setiawan and Magfirawaty, "Securing Data Communication Through MQTT Protocol with AES-256 Encryption Algorithm CBC Mode on ESP32-Based Smart Homes," 2021 International Conference on Computer System, Information Technology, and Electrical Engineering (COSITE), 2021, pp. 166-170, doi: 10.1109/COSITE52651.2021.9649577.
- [15] S. Sridhar and S. Smys, "Intelligent security framework for iot devices cryptography based end-to-end security architecture," 2017 International Conference on Inventive Systems and Control (ICISC), 2017, pp. 1-5, doi: 10.1109/ICISC.2017.8068718.
- [16] A. Abdaoui, A. Erbad, A. Al-Ali, A. Mohamed and M. Guizani, "A Robust Protocol for Smart eHealthcare based on Elliptic Curve Cryptography and Fuzzy logic in IoT," 2021 IEEE Globecom Workshops (GC Wkshps), 2021, pp. 1-6, doi: 10.1109/GCWkshps52748.2021.9682030.
- [17] F. K. Santoso and N. C. H. Vun, "Securing IoT for smart home system," 2015 International Symposium on Consumer Electronics (ISCE), 2015, pp. 1-2, doi: 10.1109/ISCE.2015.7177843.

- [18] W. Lardier, Q. Varo and J. Yan, "Dynamic Reduced-Round Cryptography for Energy-Efficient Wireless Communication of Smart IoT Devices," ICC 2020 - 2020 IEEE International Conference on Communications (ICC), 2020, pp. 1-7, doi: 10.1109/ICC40277.2020.9149305.
- [19] A. V. Jerald and S. Albert Rabara, "Secured Architecture for Internet of Things (IoT) Based Smart Healthcare," 2020 International Conference on Inventive Computation Technologies (ICICT), 2020, pp. 828-833, doi: 10.1109/ICICT48043.2020.9112586.
- [20] M. G. Z. Fernando, A. M. Sison and R. P. Medina, "Securing Private Key using New Transposition Cipher Technique," 2019 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE), 2019, pp. 490-493, doi: 10.1109/ECICE47484.2019.8942798.
- [21] H. T. T. Truong, M. Almeida, G. Karame and C. Soriente, "Towards Secure and Decentralized Sharing of IoT Data," 2019 IEEE International Conference on Blockchain (Blockchain), 2019, pp. 176-183, doi: 10.1109/Blockchain.2019.00031.
- [22] S. Mohammed and M. H. Al-Jammas, "Data Security System for IoT Applications," 2020 International Conference on Advanced Science and Engineering (ICOASE), 2020, pp. 1-6, doi: 10.1109/ICOASE51841.2020.9436579.
- [23] S. Kaushik and A. Patel, "Secure Cloud Data Using Hybrid Cryptographic Scheme," 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), 2019, pp. 1-6, doi: 10.1109/IoT-SIU.2019.8777592.
- [24] Z. Malik, A. Saxena and K. Singh, "Designing a Secure IOT data Encryption algorithm for Smart Environmental Monitoring System," 2021 International Conference on Advances in

Technology, Management & Education (ICATME), 2021, pp. 106-111, doi: 10.1109/ICATME50232.2021.9732764.

[25] J. -H. Han and J. Kim, "A lightweight authentication mechanism between IoT devices," 2017 International Conference on Information and Communication Technology Convergence (ICTC), 2017, pp. 1153-1155, doi: 10.1109/ICTC.2017.8190883.

[26] S. Chaudhry, "An Encryption-based Secure Framework for Data Transmission in IoT," 2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2018, pp. 743-747, doi: 10.1109/ICRITO.2018.8748523.

[27] K. D. Muthavhine and M. Sumbwanyambe, "An analysis and a comparative study of cryptographic algorithms used on the Internet of Things (IoT) based on avalanche effect," 2018 International Conference on Information and Communications Technology (ICOIACT), 2018, pp. 114-119, doi: 10.1109/ICOIACT.2018.8350759.

[28] J. R. Naif, G. H. Abdul-Majeed and A. K. Farhan, "Secure IOT System Based on Chaos-Modified Lightweight AES," 2019 International Conference on Advanced Science and Engineering (ICOASE), 2019, pp. 1-6, doi: 10.1109/ICOASE.2019.8723807.

[29] N. M. Ansari, R. Hussain, S. S. Hussain and S. Arif, "Invariant of AES algorithm implementations against Attacks in IoT Devices," 2021 International Conference on Computer & Information Sciences (ICCOINS), 2021, pp. 84-89, doi: 10.1109/ICCOINS49721.2021.9497174.

[30] H. Rady, H. Hossam, M. S. Saied and H. Mostafa, "Memristor-Based AES Key Generation for Low Power IoT Hardware Security Modules," 2019 IEEE 62nd International Midwest

Symposium on Circuits and Systems (MWSCAS), 2019, pp. 231-234, doi: 10.1109/MWSCAS.2019.8885031.

[31] V. Arun, D. L. Reddy and S. Srinivas, "Encryption standards for security system in energy harvesting for IoT requirements — Review," 2017 International Conference on Intelligent Sustainable Systems (ICISS), 2017, pp. 1224-1227, doi: 10.1109/ISS1.2017.8389380.

[32] G. Sravya, M. O. V. P. Kumar, Y. Sudarsana Reddy, K. Jamal and K. Mannem, "The Ideal Block Ciphers - Correlation of AES and PRESENT in Cryptography," 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS), 2020, pp. 1107-1113, doi: 10.1109/ICISS49785.2020.9315883.

[33] F. B. Setiawan and Magfirawaty, "Securing Data Communication Through MQTT Protocol with AES-256 Encryption Algorithm CBC Mode on ESP32-Based Smart Homes," 2021 International Conference on Computer System, Information Technology, and Electrical Engineering (COSITE), 2021, pp. 166-170, doi: 10.1109/COSITE52651.2021.9649577.

[34] <https://tozny.com/blog/iot-device-and-key-management-with-end-to-end-encryption/>

[35] <https://cpl.thalesgroup.com/encryption/hardware-security-modules>

[36] <https://itigic.com/end-to-end-encryption-in-iot/>

Plagiarism Report

HARDWARE AUTHENTICATION AND ENCRYPTION MODULE

ORIGINALITY REPORT

6% SIMILARITY INDEX	5% INTERNET SOURCES	0% PUBLICATIONS	4% STUDENT PAPERS
-------------------------------	-------------------------------	---------------------------	-----------------------------

PRIMARY SOURCES

1	www.iot-now.com Internet Source	1%
2	Submitted to Good Shepherd International School Student Paper	1%
3	ikee.lib.auth.gr Internet Source	1%
4	www.privacyend.com Internet Source	1%
5	Www.Appknox.Com Internet Source	<1%
6	rosadesantjordi.hostings.tecnocampus.cat Internet Source	<1%
7	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	<1%
8	Submitted to De Montfort University Student Paper	<1%
9	Submitted to Sreenidhi International School	

Student Paper

<1%

Exclude quotes On

Exclude matches Off

Exclude bibliography On