Development of Secure IoT Ecosystems for Healthcare

Pranav Shirgur
B. Tech Computer Science and Engineering,
Manipal University Jaipur, India
Email: pranav.shirgur@gmail.com

***Abstract*—The Internet of Things (IoT) is a rapidly developing field of technology which entails a network of smart devices connected to each other and the internet. The IoT industry is anticipated to increase by 18% to 14.4 billion active connections in 2022. There will likely be about 27 billion linked IoT devices by 2025 as supply limitations, brought about by the current global semiconductor and chip shortage - loosen and demand quickens. IoT has quickly penetrated the healthcare industry, this paper defines a framework that enables the development of secure and scalable IoT healthcare platforms/applications. These platforms will also allow for secure cloud storage and analysis of patient data, helping professionals recognize latent parameters such as patient behavioral patterns that contribute to an ailment. This ultimately will enable the study of social and economic impact of a particular disease. This will greatly cull the survivorship bias in the healthcare industry – especially in testing times like a pandemic.**

***Index Terms*—IoT, Microcomputers, MQTT, personalized healthcare, disease identification, secure healthcare platform.**

I. Introduction

The Internet of Things (IoT) is a cutting-edge technology that connects even commonplace devices and objects to the Internet. It is a tool for devising communication with or without human interactions. [12] This technology has penetrated the healthcare sector especially as wearable technologies for smart personalized healthcare. [4] The United Nations *Department of Economic and Social Affairs* confirmed that the world population has reached 8 billion as of November 2022 [13]. Data from *World Population Prospects: the 2019 Revision* [14] reflects that one in six people in the world will be over age 65 (16%) by 2050. According the National Council on Ageing 80% of people aged over 65 suffer from at least one chronic ailment. [15]

Most chronic diseases such as Alzhimers, Diabetes, Chronic-Arthritis need constant monitoring of a patient's body statistics and health. This paper aims to define a framework that would enable the development of IoT healthcare applications or platforms allowing stakeholders to adapt to smart healthcare.

II. Application Objectives

The effectiveness of incorporation of IoT in the field of healthcare has been exemplified by smart healthcare monitoring environments [1] scaled up to BSN (Body Sensor Networks) [2] as well as characteristic solutions such as OpenAPS, an open-source platform that provides Continuous Glucose Monitoring for diabetic patients [3]

A. Monitoring Patient Data

Wearable sensors [4] monitoring patient data will use cloud integration to study trends in these metrics, this data being digital will allow for automation of analysis by medical professionals and allow for agile responses by caregivers.

B. Cloud Integration

A considerable implementation of this application will provide a liberal dataset on the cloud to study the social and economic impact of a particular disease. This will help generalize factors of influence of a disease, facilitating better recognition of how it shapes individual elements contributing to it, promoting measures of prediction and prevention.

C. Unique Patient Profile

Each patient will have a UID (Universal Identity) within the environment which will track fluctuations and monitor health metrics secured to the individual.

D. False Positive Elimination

Rapid scanning of patient health metrics will aid in subjugating an ailment while expediting recognition and classification of a disease, helping eliminate false positives bound to its diagnosis.

E. Remote Patient Monitoring

Capturing of patient health data using wearable sensors can be streamed to healthcare professionals remotely [1]. This also enables preventive and short-term care (STC) [1] enabling identification of a disease in its rudimentary stages and overseeing a patient's health post treatment [5].

F. Critical Overhead Channel (COC)

The Critical Overhead Channel is a pipeline that is utilized when sensors reading healthcare metrics of a patient pick up data that needs physical attention of a healthcare professional such as extremely low sugar, erratic heartrate etc. The Critical Overhead Channel bypasses non-essential security protocols to preserve an equitable execution speed.

III. Risk Factors and Security Requirements

The described application is largely wireless in its nature with its implementation being predominantly in healthcare institutions such as hospitals making securing of aspects such as user/patient data confidentiality, authentication and other components related to its preservation of paramount importance.

Risks to the security and privacy of an IoT based healthcare environment [6] are defined below:

- Loss of data privacy [7]
- Hijacked and modified data [7]
- Location privacy [8]
- Genuine authentication on medical/clinical procedure [9] [10]
- Attack on server security [11]

Broad conceptual mitigations for these risks [2] are defined below:

A. Data Privacy

In an IoT based healthcare infrastructure vital and confidential patient data is sent to a server through a series of nodes. Privacy of data must be ensured to prevent MITM attacks and modification of data by a malicious actor at a compromised node.

B. Data Integrity

The networking infrastructure must be designed in such a way that it adheres to the principles of data integrity, preventing retransmission of maliciously altered data packets to successive nodes in the network.

C. Data Freshness

Data being transmitted through the network is chronologically relevant vital patient data and must not be accepted by the system if an old key is used for authentication to filter out possible redundant and compromised data.

D. Authentication

To enable smooth functioning of the environment and to prevent malicious impersonation of stakeholders within the environment itself, authentication practices must be put in place to distinguish layers of access.

E. Anonymity

Anonymity shields the origin of the data packet while making it indistinguishable from its successive counterparts preventing a malicious actor to exploit a specific user. In contrast, this follows the UID (Unique Identification) objective of the application.

IV. Infrastructure

To ensure development of a secure ecosystem, it is developed in adherence with OWASP Internet of Things Security Verification Standard (ISVS) which defines requirements and best practices to establish levels of confidence in IoT security.

ISVS describes three security verification levels, the complexity and efficiency of security increases with the level.

a. ISVS Level 1

Level 1 provides protection against software attacks and compromises; here physical compromise of the device and hardware do not result in high security hazards.

b. ISVS Level 2

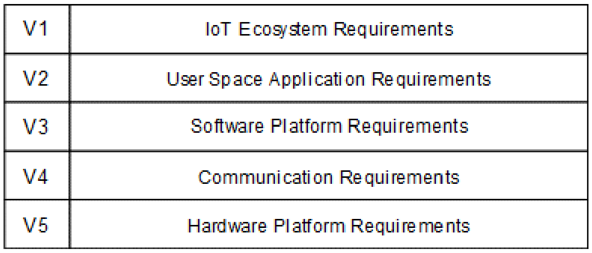
Level 2 aims to secure physical access to the device hardware; level 2 assumes that sensitive information is stored on the device hardware and any malicious access to it must be prevented.

c. ISVS Level 3

Level 3 includes requirements set by level 1 and level 2 with the addition of other defense-in-depth techniques to hinder malicious efforts. Level 3 assumes that the data stored on the device hardware and its successive elements is sensitive enough where its compromise could be treated as a fraud.

Adherence up to ISVS Level 2 is optimal for the development of this application.

The ISVS standard also describes 5 requirement classes:



V1	IoT Ecosystem Requirements
V2	User Space Application Requirements
V3	Software Platform Requirements
V4	Communication Requirements
V5	Hardware Platform Requirements

Elaboration of these classes and how they are integrated in the application are defined below:

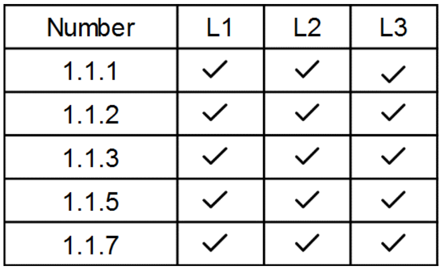
A. V1: IoT Ecosystem Requirements

IoT Ecosystem Requirements deal with system security design prior to development and practices during development that help create a secure ecosystem integrating all the necessary elements contributing to creating and maintaining a secure product architecture and implementing it. There are three generalized security requirements, listed below.

- Application and Ecosystem Design
- Supply chain
- Secure Development

The following requirements are included in the application along with its compliance to consecutive security levels.

1. Application and Ecosystem Design



Number	L1	L2	L3
1.1.1	✓	✓	✓
1.1.2	✓	✓	✓
1.1.3	✓	✓	✓
1.1.5	✓	✓	✓
1.1.7	✓	✓	✓

1.1.1
All applications developed within the ecosystem must be developed with a level of security in line with the criticality of the entire application.

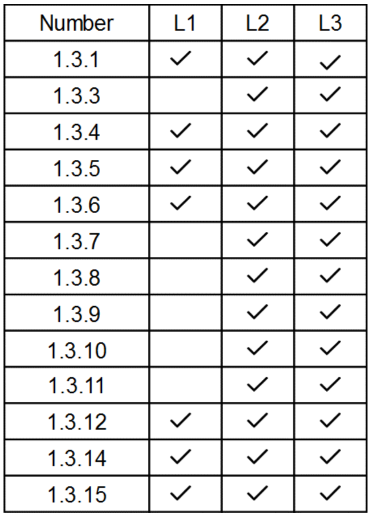
1.1.2
All communication components within the IoT ecosystem that are essential are identified, redundant and unnecessary elements are to be removed.

1.1.3
Use of threat modelling to simulate and identify potential threats as features are added to the application.

1.1.5
Verify that security controls are enforced server-side and that data and instructions are not blindly trusted by server-side components.

1.1.7
Appropriate stakeholders must be notified when a vulnerability is identified through established communication channels.

2. Secure Development



Number	L1	L2	L3
1.3.1	✓	✓	✓
1.3.3		✓	✓
1.3.4	✓	✓	✓
1.3.5	✓	✓	✓
1.3.6	✓	✓	✓
1.3.7		✓	✓
1.3.8		✓	✓
1.3.9		✓	✓
1.3.10		✓	✓
1.3.11		✓	✓
1.3.12	✓	✓	✓
1.3.14	✓	✓	✓
1.3.15	✓	✓	✓

1.3.1
Verify that the application and all its successive components are built in a secure and repeatable environment.

1.3.3
Verify that banned C/C++ functions are not used in development and that their safe counterparts are used instead.

1.3.4
Packages must be downloaded and built from trusted resources.

1.3.5
Build pipelines must only perform builds of the source code in adherence to version control systems in use.

1.3.6
Verify that compilers, SDKs, version control clients and other development tools are analyzed and monitored for tampering, trojans and other malicious code.

1.3.7
Packages must be compiled with Object Size Checking (OSC)

1.3.8
Packages must be compiled with No eXecute (NX) or Data Execution Protection (DEP)

1.3.9
Packages must be compiled with Position Independent Executable (PIE)

1.3.10
Packages must be compiled with Stack Smashing Protector (SSP)

1.3.11

Packages must be compiled with read only relocation (RELRO)

1.3.12
Verify that release builds do not have privileged diagnostic functionality and debug code.

1.3.14
Verify that debug information does not contain sensitive information and credentials.

1.3.15
Verify that embedded applications are not susceptible to OS command injection by performing input validation and escaping of parameters within firmware code, shell command wrappers, and scripts.

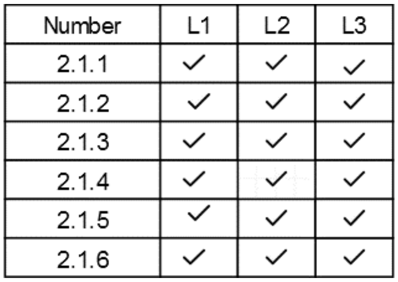
B. V2: User Space Requirements

The requirements listed below ensure secure access to an IoT system by appropriate stakeholders and that sensitive information is protected.

- Identification and Authentication
- Authorization
- Data Protection
- Cryptography

All the requirements listed above are paramount for secure functioning of the proposed application.

1. Identification and Authentication



Number	L1	L2	L3
2.1.1	✓	✓	✓
2.1.2	✓	✓	✓
2.1.3	✓	✓	✓
2.1.4	✓	✓	✓
2.1.5	✓	✓	✓
2.1.6	✓	✓	✓

2.1.1
All accounts and users within the ecosystem must be uniquely identified.

2.1.2
All sensors and connected devices must be uniquely identified within the ecosystem, their connection to other devices, hubs and other elements like the cloud must also be identified uniquely.

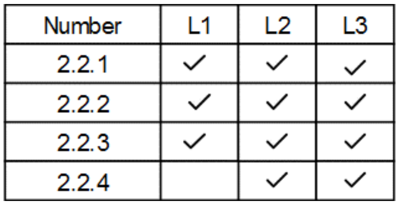
2.1.3
A robust user and device authentication system must be implemented.

2.1.4
Authentication schemes for connected devices, users and services must share a common centrally managed framework in the ecosystem.

2.1.5
Certificate based authentication must be preferred over password-based authentication.

2.1.6
Strong password policies must be implemented disallowing hardcoded passwords and duplicate identities or same password across multiple devices.

2. Authorization



Number	L1	L2	L3
2.2.1	✓	✓	✓
2.2.2	✓	✓	✓
2.2.3	✓	✓	✓
2.2.4		✓	✓

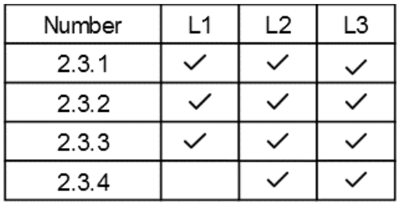
2.2.1
Users, services and devices across the ecosystem must share a common authorization framework.

2.2.2
Least privilege must be enforced, allowing only certain applications and services to be run as administrator or root.

2.2.3
Verify that ownership is validated upon registration and as part of decommissioning when devices move across accounts.

2.2.4
Verify device debug capabilities can only be accessed by approved staff.

3. Data Protection



Number	L1	L2	L3
2.3.1	✓	✓	✓
2.3.2	✓	✓	✓
2.3.3	✓	✓	✓
2.3.4		✓	✓

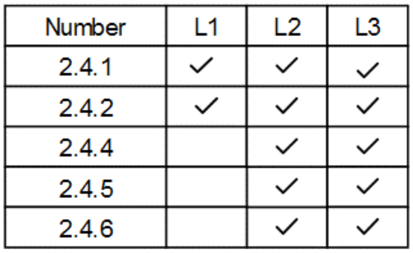
<u>2.3.1</u>
Sensitive information and user account credentials are stored securely using strong encryption to prevent data leakage while integrity checking to scan for tampering of data.

<u>2.3.2</u>
Verify that if any device is decommissioned all sensitive data from it is removed.

<u>2.3.3</u>
Manage a centrally managed database to keep record of devices which have been decommissioned or whose ownership has changed for auditing.

<u>2.3.4</u>
Sensitive information kept in memory must be overwritten with zeros when no longer required.

4. Cryptography



Number	L1	L2	L3
2.4.1	✓	✓	✓
2.4.2	✓	✓	✓
2.4.4		✓	✓
2.4.5		✓	✓
2.4.6		✓	✓

<u>2.4.1</u>
Cryptographic secrets and keys must be unique per device.

<u>2.4.2</u>
Only standard cryptographic algorithms must be used; the key size must be adequate, and all implementations must be secured.

<u>2.4.4</u>
Verify that cryptographic secrets used by the device are stored securely by leveraging functionality provided by dedicated security chips if any.

<u>2.4.5</u>
Verify that cryptographic primitives used by the device are provided by dedicated security chips if any.

<u>2.4.6</u>
Cryptographic libraries used must be certified to be compliant with a recognized cryptographic standard.

C. V3: Software Platform Requirements

Software platform requirements deal with securing the operating system during boot and the kernel, these requirements set a standard for configuring the firmware vendor to secure the bootloader.

Boot trustworthiness is ensured by verifying cryptographic signatures against loaded code, barring access to memory, shell and other debug access during boot time and disallowing images to load from external locations.

The operating system and kernel run in privileged execution modes, containing many security primitives.

This application does not instruct the use of any particular operating system, making use of these standards very modular. Developers are free integrate the operating system that they deem most appropriate into their application, regardless of the choice of operating system it is highly recommended that the appropriate stakeholders introduce the directives included in the *OWASP ISVS V3: Software Platform Requirements* into the core of their project.

Generalized requirements are listed below.

- Bootloader
- OS Configuration
- Linux
- Software Updates
- Security Chip Integrations
- Kernel Space Application Requirements

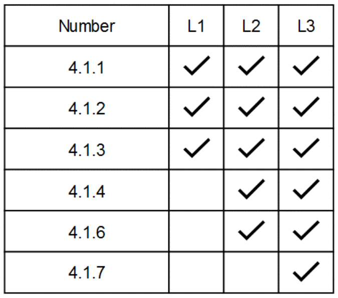
D. V4: Communication Requirements

Communication requirements define directives to secure the elements of communication within the ecosystem.
Different parties within the ecosystem need to trust the contents in the communication channels, these parties must be authenticated, malicious tampering of packets must be prevented while maintaining confidentiality against information leakage.
Generalized communication requirements are listed below.

- General Requirements
- Machine-to-Machine
- Bluetooth
- Wi-Fi

Requirements that this application will require are listed below.

1. General Requirements



Number	L1	L2	L3
4.1.1	✓	✓	✓
4.1.2	✓	✓	✓
4.1.3	✓	✓	✓
4.1.4		✓	✓
4.1.6		✓	✓
4.1.7			✓

4.1.1
All components in the IoT ecosystem must communicate over secure channels in which confidentiality and integrity of data is guaranteed and the communication protocol in use must house protection against replay attacks.

4.4.2
In case TLS is used, its configured to only use FIPS-compliant cipher suites, or other equivalents.

4.1.3
In case TLS is used, the device cryptographically verifies the X.509 certificate.

4.1.4
protection or detection of jamming must be provided for availability-critical applications.

4.1.6
Verify that the device's TLS implementation uses its own certificate store, pins to the endpoint's certificate or public key, and disallows connections to endpoints with different certificates or keys, even if signed by a trusted CA.

4.1.7
Inter-chip communication must be encrypted.

2. Machine-to-Machine

Number	L1	L2	L3
4.2.1	✓	✓	✓
4.2.2	✓	✓	✓
4.2.3	✓	✓	✓

4.2.1
Unencrypted data must only consist of data and instructions not of sensitive nature.

4.2.2
Verify MQTT brokers only allow authorized IoT devices to subscribe and publish message topics.

4.2.3
Verify certificates are favored over native username and passwords to authenticate MQTT transactions.

3. Wi-fi

Number	L1	L2	L3
4.4.1	✓	✓	✓
4.4.2	✓	✓	✓
4.4.3	✓	✓	✓
4.4.4	✓	✓	✓

4.4.1
Verify Wi-Fi connectivity is disabled unless required as part of device functionality. Devices with no need for network connectivity or which support other types of network connectivity, such as Ethernet, should have the Wi-Fi interface disabled.

4.4.2
Verify that WPA2 or higher is used to protect Wi-Fi communications.

4.4.3
Verify that in case WPA is used, it is used with AES encryption (CCMP mode).

4.4.4
Verify that Wi-Fi Protected Setup (WPS) is not used to establish Wi-Fi connections between devices.

E. V5: Hardware Platform Requirements

Compromising hardware is significantly more difficult due to its robust nature, however this does not make it immune to attacks by malicious actors. Backdoors, undocumented debug features can be leveraged to compromise the device and ultimately the ecosystems security.
Listed below are the Design based requirements to strengthen device security, this mainly concerns wearable devices and the other hardware nodes across the ecosystem.

1. Design

Number	L1	L2	L3
5.1.1		✓	✓
5.1.2		✓	✓
5.1.3		✓	✓
5.1.4		✓	✓
5.1.5		✓	✓
5.1.6		✓	✓
5.1.7		✓	✓
5.1.8		✓	✓
5.1.9			✓
5.1.10			✓

5.1.1
Platform must support protecting or disabling access to debug interfaces.

5.5.2
Platform must support validation of authenticating the first stage bootloader.

5.5.3
Cryptographic functions must be provided by the platform.

5.5.4
Sensitive data such as private keys and certificates must be stored securely by leveraging dedicated hardware security features, preferably.

5.5.5
Verify that the platform provides memory and I/O protection capabilities so that only privileged processes can access certain resources.

5.5.6
Verify that the security configuration of the platform can be locked.

5.5.7
Debugging headers must be removed from all successive PCBs.

5.5.8
Verify the chosen hardware has no unofficially documented debug features, such as special pin configurations that can enable or disable certain functionality.

5.5.9
Verify that the platform provides protection against physical decapsulation, side channel and glitching attacks.

5.5.10
Descriptive silkscreens must be removed from all successive PCBs.

V. Networking Ecosystem

This section of the document describes how different modules and hardware units that constitute towards building the application interact with each other using specified networking protocols and their security features.

A. *Message Queuing Telemetry Transport (MQTT)*

Our application will leverage MQTT as its foundational networking protocol within its ecosystem, due to its lightweight nature, low bandwidth consumption and utilization of a publish/subscribe communication pattern.

Advantages of MQTT –

- MQTT uses a publish subscribe model that fits well within the IoT ecosystem we have designed so far.

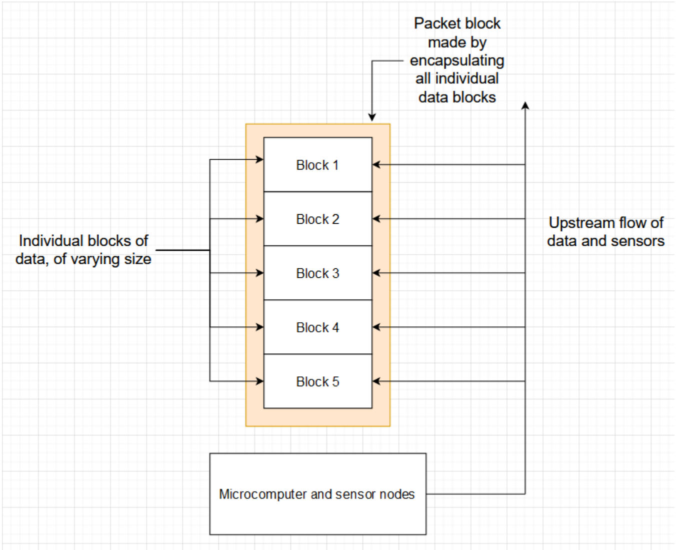
- MQTT has a secure extension called the *Secure Message Queue Telemetry Transport* (SMQTT) which uses lightweight attribute encryption for added security.

- MQTT allows for fast and efficient message delivery utilizing small amounts of power making it optimal for connected devices used for remote sensing and control.

- HTTP and MQTT both run over TCP connections and adhere to a client-server architecture.
  MQTT however needs to establish its security protocols only once within its ecosystem whereas HTTP needs to re-establish its security protocols every time a device submits data within a HTTP system.

  MQTT also allows messages to be passed to-and-fro between clients and servers whereas HTTP servers only pay respond to client requests.

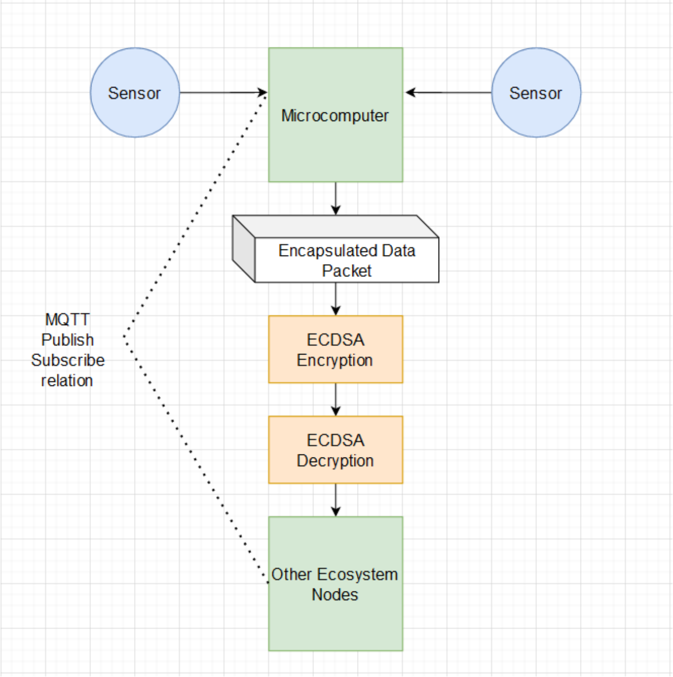
Data sensed by various sensors will be encapsulated and compressed into a single packet before the encryption and transfer to achieve synchronous packet transmission and prevent Mutual Exclusions which will hamper the interpretation of data.


Packet block made by encapsulating all individual data blocks
Individual blocks of data, of varying size
Block 1
Block 2
Block 3
Block 4
Block 5
Upstream flow of data and sensors
Microcomputer and sensor nodes

B. *Elliptical Curve Cryptography*

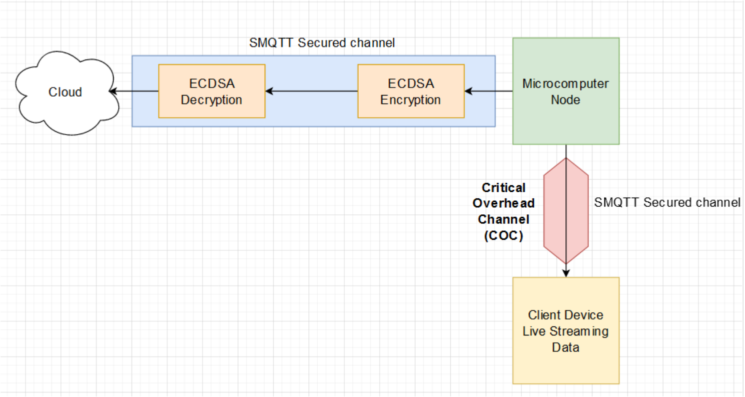
Using a lightweight encryption scheme for this setup is paramount considering the limited computing capabilities of microcomputers. Elliptic Curve Cryptography (ECC) is the most equitable implementation for resource constrained

devices [16], as opposed to AES or RSA. ECC is a public key-based cryptography solution based on elliptic curves over finite fields. For lower bit processors ECC's level of security is not compromised in comparison to RSA. [16] [17]
Each encapsulated packet will use Elliptical Curve Digital Signature Algorithm (ECDSA) an ECC-based cryptographic scheme based on the Digital Signature Algorithm (DSA) to verify appropriate stakeholder nodes and preserve the integrity of patient data. This relationship will hold between various nodes and the microcomputers in use.



C. *Secure Message Queuing Telemetry Transport (SMQTT)*

SMQTT is an extension of the MQTT protocol, it adds a lightweight attribute-based encryption over elliptical curves [18] and is a session layer protocol.
Here, the data is encrypted before publishing by the broker and the subscribers have to perform appropriate decryption. Here the key generation and encryption algorithms are not standardized they depend on developer implementation.
In our model SMQTT and ECDSA will be leveraged to store data in the cloud while the Critical Overhead Channel (COC) will only use SMQTT to reduce computational load in case of a medical emergency or a scenario that requires physical attendance of a medical professional. The COC livestreams patient data on a client device.



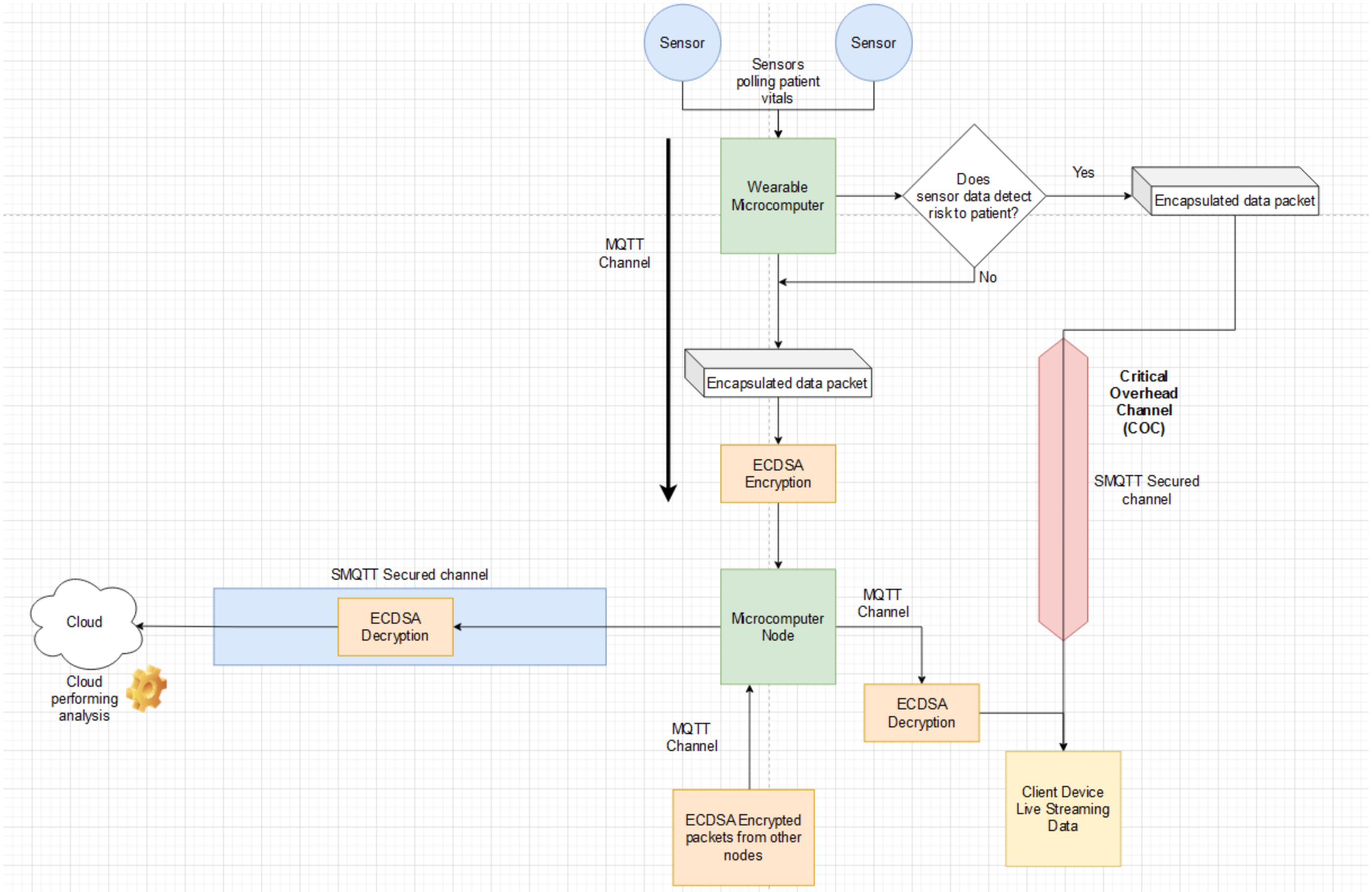
VI. Conclusion

We have now devised a secure framework that can be leveraged by corporations to develop their own scalable and secure IoT healthcare ecosystem that adheres to essential constraints defined by the Internet of Things Security Verification Standard (ISVS). We have also eliminated the need for obscure and complicated monitoring devices, all a medical professional needs in order to view a patient's vitals in real time is a web-page or a mobile application. Monitoring of patient vitals is enabled using wearable technology.

The integration of the cloud in this ecosystem enables medical professionals to analyze a large dataset of patient parameters, preferable with an AI integration. This dataset will enable the research of social and economic factors that accompany a disease. This will also help identify patient behavioral patterns that aggravate an ailment. Cloud integration here aims to minimize survivorship bias in the said ecosystem by providing objective diagnosis based on empirical data obtained by analysis of the dataset.

Each patient in the ecosystem has a Unique Identifier (UID) which enables Short Term Care (STC) and minimizes false positive distinguishing of symptoms with the help of could integration - that may lead to a potential misdiagnosis.

The Critical Overhead Channel (COC) alerts medical professionals when the patient is in need of their attention, and is designed to minimize computational load on the ecosystem in such an event.


Sensor
Sensor
Sensors polling patient vitals
Wearable Microcomputer
Does sensor data detect risk to patient?
Yes
No
Encapsulated data packet
MQTT Channel
Encapsulated data packet
ECDSA Encryption
Critical Overhead Channel (COC)
SMQTT Secured channel
SMQTT Secured channel
Cloud
ECDSA Decryption
Microcomputer Node
MQTT Channel
ECDSA Decryption
Cloud performing analysis
MQTT Channel
ECDSA Encrypted packets from other nodes
Client Device Live Streaming Data

VII. References

[1] Sarierao, Borade Samar, and Amara Prakasarao. "Smart healthcare monitoring system using mqtt protocol." In *2018 3rd international conference for convergence in technology (I2CT)*, pp. 1-5. IEEE, 2018.

[2] Gope, Prosanta, and Tzonelih Hwang. "BSN-Care: A secure IoT-based modern healthcare system using body sensor network." *IEEE sensors journal* 16, no. 5 (2015): 1368-1376.

[3] Chakrabarty, Ankush, Stamatina Zavitsanou, Tara Sowrirajan, Francis J. Doyle III, and Eyal Dassau. "Getting IoT-ready: The face of next generation artificial pancreas systems." In *The Artificial Pancreas*, pp. 29-57. Academic Press, 2019.

[4] Krey, Mike. "Wearable Technology in Health Care–Acceptance and Technical Requirements for Medical Information Systems." In *2020 6th International Conference on Information Management (ICIM)*, pp. 274-283. IEEE, 2020.

[5] Aburukba, Raafat, Assim Sagahyroon, and Mohammed Elnawawy. "Remote patient health monitoring cloud brokering services." In *2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pp. 1-6. IEEE, 2017.

[6] Zakaria, Huraizah, Nur Azaliah Abu Bakar, Noor Hafizah Hassan, and Suraya Yaacob. "IoT security risk management model for secured practice in healthcare environment." *Procedia Computer Science* 161 (2019): 1241-1248.

[7] Laplante, Phillip A., and Nancy Laplante. "The internet of things in healthcare: Potential applications and challenges." *It Professional* 18, no. 3 (2016): 2-4.

[8] Lingaraj, K., Rajashree V. Biradar, and V. C. Patil. "A survey on middleware challenges and approaches for wireless sensor networks." In *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*, pp. 56-60. IEEE, 2015.

[9] Rosenbaum, Benjamin P. "Radio frequency identification (RFID) in health care: privacy and security concerns limiting adoption." *Journal of medical systems* 38, no. 3 (2014): 1-6.

[10] Wu, Zhen-Yu, Lichin Chen, and Ju-Chuan Wu. "A reliable RFID mutual authentication scheme for healthcare environments." *Journal of medical systems* 37, no. 2 (2013): 1-9.

[11] Abie, Habtamu, and Ilangko Balasingham. "Risk-based adaptive security for smart IoT in eHealth." In *Proceedings of the 7th International Conference on Body Area Networks*, pp. 269-275. 2012.

[12] Sholla, Sahil, Roohie Naaz, and Mohammad Ahsan Chishti. "Incorporating ethics in Internet of Things (IoT) enabled connected smart healthcare." In *2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*, pp. 262-263. IEEE, 2017.

[13] [online] Available: https://www.un.org/en/desa/world-population-reach-8-billion-15-november-2022

[14] [online] Available: https://www.un.org/en/global-issues/ageing

Basic format for journals (when available online):

[15] [online] Available:https://www.ncoa.org/article/the-top-10-most-common-chronic-conditions-in-older-adults

[16] Rao, Vidya, and K. V. Prema. "Lightweight authentication and data encryption scheme for IoT applications." In *2020 IEEE International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)*, pp. 12-17. IEEE, 2020.

[17] Gura, Nils, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz. "Comparing elliptic curve cryptography and RSA on 8-bit CPUs." In *International workshop on cryptographic hardware and embedded systems*, pp. 119-132. Springer, Berlin, Heidelberg, 2004.

[18] Dhanshri Kolhe & Prof. Smita Kapse. “Design and Implementation of SMQTT for IoT Applications “International Journal of New Technologies in Science and Engineering Vol. 5, Issue. 3, 2018, ISSN 2349-0780.