Master of Science in Electrical Engineering with Emphasis On Telecommunication Systems
May 2023

The Deeper Investigation of Smart Healthcare Systems using 5G Security

Bindu Ananthula
Niharika Budde

Faculty of Faculty of Telecommunication Systems, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden

Contact Information:
Author(s):
Bindu Ananthula
E-mail: bian21@student.bth.se

Niharika Budde
E-mail: nibu21@student.bth.se

University advisor:
Oleksii Baranovskyii
Department of Department of Computer Science

Faculty of Faculty of Telecommunication Systems	Internet	:	www.bth.se
Blekinge Institute of Technology	Phone	:	+46 455 38 50 00
SE–371 79 Karlskrona, Sweden	Fax	:	+46 455 38 50 57

Abstract

A promising approach to raising the caliber and accessibility of healthcare services is the development of Smart Healthcare Systems. However, the union of wireless networks and smart medical devices has created additional security issues, such as the possibility of identity theft, data breaches, and denial-of-service assaults. These flaws emphasize the significance of creating a safe and dependable smart healthcare system that can safeguard patient data and guarantee the confidentiality of private medical information.

This study suggests adopting 5G security standards to address the security issues with smart healthcare systems. The STRIDE threat modeling approach, which includes six threat categories (spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege), is used in this study to investigate potential threats in smart healthcare systems. The report suggests using strong encryption protocols, such as AES-CCM and ECDH, between smart healthcare equipment and 5G-AKA to reduce these potential threats.

The proposed approach showed appreciable advancements in data security and privacy. According to the findings, 5G security standards can be used to efficiently reduce security risks in smart healthcare systems and establish a trustworthy and secure platform for delivering medical services. The study emphasizes the significance of including strong security controls in Smart Healthcare Systems to secure patient information and raise the standard of treatment generally.

Keywords: Smart Healthcare, 5G, IoT, Security, Risk Management

Acknowledgments

We are sincerely grateful to Mr. Oleksii Baranvskyii, who is in supervising our thesis, for allowing us to undertake this research and for his insightful suggestions. Throughout the entire educational process, he has been our source of support. Working with him in the present is a privilege and an honor for us. With his polite and adaptable suggestions and criticism, we were able to remove obstacles from the thesis work. We also appreciate the help and inspiration our family and friends have given us. In the context of this, we would want to draw emphasize BTH's superior instruction and dedication to encouraging students to succeed.

Contents

List of Figures

List of Tables

List of Abbreviations

1G-5G	First Generation - Fifth Generation
3gpp	Third Generation Partnership Project
AES-CCM	Advanced Encryption Standard with Counter with CBC-MAC
AI	Artificial Intelligence
AKA	Authentication and Key Agreement
AMF	Access and Mobility Management Function
API	Application Program Interface
ARPF	Authentication Credential Repository Processing Function
AUSF	Authentication Server Function
AUTH	Authentication token
AV	Authentication Vector
BAN	Body Area Networks
BS	Base Station
CCA	Chosen Cipher-text Attack
CMD	Configuration Management Database
CNN	Convolutional Neutral Network
CPS	Child Protective Service
CT	Computed Tomography
D2D	Device to Device
DHE	Diffie–Hellman Encryption
DoS	Denial of Service
ECC	Elliptic Curve Cryptography

ECDH	Elliptic-curve Diffie–Hellman
EEC	Europe Economic Community
EHR	Electronic Health Records
EMT	Emergency Medical Technicians
EPS	Evolved Packet System
ETSI	European Telecommunication Standard Institution
EU	European Union
GUTI	Globally Unique Temporary Identify
HIPAA	Health Insurance Portability and Accountability Act
HN	Home Network
HSS	Home Subscriber Server
HTTP	Hyper Text Transfer Protocol
ICT	Information and Communication Technology
iGW	Industry gateway
IK	Integration Key
IMSI	International Mobile Subscriber Identity
IoMT	Internet of Medical Things
IoT	Internet of Things
Ipv6	Internet Protocol Version 6
KAMF	Key for Access Mobility Management Function
KASME	Key for Access Security Management Entity
MAC	Media Access Control
MEC	Multi-access Edge Computing
MITM	Man in the Middle
MME	Mobile Management Entities
RES token	Response Token
RFID	Radio Frequency Identification
RRC	Radio Resource Control

RSA	Rivest-Shamir-Adleman encryption
SEAF	Security Anchor Function
SHS	Smart Healthcare Systems
SIDF	Subscription Identifier De-concealing Function
SN	Serving Network
SS	Subscribe Station
SUCI	Subscription Concealed Identifier
SUPI	Subscription Permanent Identifier
TLS	Transport Layer Security
UDM	Unified Data Management
UE	User Equipment
USIM	Universal Subscriber Identity Module
VPN	Virtual Private Network
XAUTH token	Expected Authentication Token
XRES token	Expected Response Token

Chapter 1

Introduction

1.1 Smart Healthcare Systems

A newly formed prototype is produced through technological advancement and adopted on a daily basis. Smart gadgets use Secure Internet protocols to transfer sophisticated data. These gadgets are used by many industries, including those in healthcare, business, administration, and education. The importance of Smart Healthcare Systems (SHS) is highlighted by the fact that some of them give users more control over their health data [1].

SHS uses networks and smart devices (such as smartphones, smartwatches, wireless smart glucometers, and wireless blood pressure monitors) to deliver healthcare services. (e.g., Body area network, wireless local area network, extensive area network). Smart devices examine health information gathered from many sources, such as sensors and biological systems. (i.e., the application has information about medical science such as diagnosis, treatment, and prevention of disease). In short, smart healthcare makes it possible for people from different backgrounds and professions (such as doctors, nurses, patient caregivers, family members, and patients) to access the right information and find the right solutions, which are primarily to reduce medical errors, increase efficiency, and cut costs when necessary. Improved patient care, lower healthcare costs, and greater provider efficiency are just a few advantages that SHS can give. These systems can aid medical professionals in optimizing their procedures as well as lowering mistakes, and improving patient outcomes. Furthermore, by giving patients real-time data and individualized treatment plans, smart healthcare systems can enable individuals to play a more active role in their healthcare. Overall, smart healthcare systems have the potential to change the healthcare sector by improving access to, affordability of, and effectiveness of healthcare for everybody.

SHS connects people, resources, and institutions involved in providing healthcare, allows for dynamic information access, and then actively monitors and responds to the needs of the medical ecosystem. These technologies include the wearable, the Internet of Things, and mobile Internet [2]. Smart healthcare may promote collaboration among all parties involved in the healthcare sector, ensure that participants receive the services they require, support decision-making, and promote resource efficiency. Simply said, smart healthcare is an improved state of information production in the medical field. With the aim to connect the people, organizations, and resources involved in providing healthcare, dynamically access information, and then actively manage and intelligently respond to the needs of the medical ecosystem, a smart healthcare system uses technology like wearable, the Internet of things, and mobile internet. Smart healthcare may promote communication among all healthcare sector participants, ensure that participants receive the services they require, enhance decision-making, and promote the effective use of resources. Simply said, smart healthcare represents a higher standard of information production within the medical sector.

1.2 5G

The most recent standard for cellular networks is 5G or fifth-generation wireless technology. It offers increased device connectivity, higher data rates, and reduced latency. High-frequency radio waves, commonly referred to as millimeter waves, are used by 5G networks to deliver data [3]. The desire for data-intensive applications like streaming video, online gaming, and the Internet of Things has fueled the development of 5G technology. (IoT). These applications require connections that are quicker and more dependable, and 5G networks are built to serve new technologies like driverless vehicles and smart cities. The installation of new infrastructure, such as tiny cells and antennas, is necessary to support the higher frequency bands required by 5G, making the deployment of 5G networks a challenging task. Concerns abound regarding the potential negative impacts of 5G on human health as well as the environment and current infrastructure. Despite these difficulties, many nations are already implementing 5G networks, with some predicting it would revolutionize sectors including healthcare, education, and industry. The way we live, work and communicate is projected to significantly change as 5G technology develops.

1.2.1 Use Of 5G in SHS

There are numerous 5G application cases in the healthcare industry. The ability to enable high-quality consultations and medical treatments for patients regardless of their locations is the biggest advantage. So, let's look at some of the top use cases [4].

- *5G and Telemedicine*: For 5G to deliver telemedicine, especially in remote areas, a network that can handle real-time, high-resolution calls without stops or jitters and high-speed downloads of enormous data are required [5]. The capacity of 5G to support remote operations and other complicated medical procedures is a significant benefit of telemedicine. Doctors can remotely manage surgical robots with more precision thanks to 5G's low latency and high bandwidth, lowering the possibility of mistakes and consequences. Higher bandwidths and lower latency provided by 5G networks can lead to better telehealth sessions, lower costs for healthcare facilities and overall better patient experiences. Because it cuts down on doctor visits and connects people in remote parts of the world with medical professionals, telemedicine is also the safest option for dealing with pandemics.

- *Augmented Reality In Healthcare*: From remote monitoring to surgery, augmented reality has numerous applications in healthcare. Using Augmented Reality in a 4G setting with lag durations ranging from 20 to 50 milliseconds.

Because there is a mismatch between visual identification and brain processing, these gaps cause motion sickness over time [6]. These constraints are overcome by 5G networks' increased performance, bandwidth, and dependability.

- *Clinical Collaboration and Communication*: The issue can be resolved by allowing employees in laboratories to send huge files in a matter of seconds without sacrificing quality via a hospital-wide 5G platform [7]. A platform like this connects all the hospital's technology, including the medical equipment and the staff's tablets and cellphones, allowing real-time data access. so that the doctors don't have to wait a long time to obtain medical imaging and other lab tests that are finished by other doctors in order to treat the patient at various times of emergency.

- *Computer-Aided Diagnostics - Medical Imaging*: Medical image data, such as that from a CT or MRI scan, is quite large and transfers slowly. With 5G, these transfers might be nearly instantaneous, allowing clinicians to identify patients more quickly and provide full therapy. 5G can also enable robot-assisted remote imaging, which has the potential to revolutionize the way patients with extremely contagious diseases like COVID-19 are treated. MRIs can also be converted into detailed holograms that doctors can show to patients while consulting with specialists and professionals like radiologists with the use of 5G. These methods allow clinicians to more accurately picture their patients' situations, eliminate guesswork, and arrive at faster more accurate diagnoses.

- *5G Connected Ambulance*: EMTs in an ambulance may communicate with hospital officials in real time about a patient's symptoms, vitals, and medical history thanks to 5G connectivity. Additionally, they can use 4K video conversations to alert the doctors to the scenario and assist them in properly preparing beforehand, particularly when dealing with unforeseen circumstances brought on by accidents, natural catastrophes, or pandemics. Doctors are even able to remotely monitor the entire situation and advise the ambulance team on the best course of action [8]. This real-time data interchange, along with the effective use of time and medical resources, shortens the time it takes for an ambulance to change hands, boosts the number of people using the emergency room, and improves the patient experience overall.

- *Remote Patient Monitoring*: Remote patient monitoring is made possible by advancements in wearable technology and medical equipment. The appropriate modern technology can be installed in patients' homes to monitor their vital signs, symptoms remotely, and healing over low-latency, high-bandwidth 5G networks [9]. Such networks enable everything from real-time vitals monitoring with medical devices to live-streaming patients and their illnesses. By swiftly releasing recovering patients and remotely monitoring their vitals, hospitals can lower their overall costs and accommodate larger patients.

- *5G Remote Surgery*: With the exception of the largest multispecialty hospitals in major cities, most hospitals lack experienced surgeons for every specialty. In these circumstances, doctors on the scene consult with the specialists via video and audio. However, 5G connectivity can go a step further by providing the specialists with real-time information from an ongoing surgery while they counsel the surgical team [10]. For instance, in addition to live-streaming the entire procedure with audio, surgical microscopes with 4K monitors can send consulting physicians high-quality pictures. With such information, consulting experts may offer greater monitoring and direction.

1.3 Basic Security controls in 5G

A security mechanism known as 5G-AKA (5G Authentication and Key Agreement) is used in 5G networks to authenticate devices and provide secure communication channels. With improvements to satisfy the more stringent security needs of 5G networks, it is an evolution of the AKA protocol used in earlier cellular networks. Network slicing capability, which enables the creation of many virtual networks inside a single physical network infrastructure, is one of the main aspects of 5G-AKA. The 5G-AKA protocol imposes its own set of access control rules and security parameters on each network slice. Additionally, 5G-AKA enables improved privacy features such the temporary identifiers that shield a device's identity and stop tracking. The protocol also has safeguards against attacks like eavesdropping, replay assaults, and man-in-the-middle attacks [11].
The process of converting information into a secret code that conceals its true meaning is known as encryption. In the world of computers, ciphertext refers to encrypted data, and plaintext to unencrypted data. Encryption algorithms, often known as ciphers, are the mathematical formulas used to encode and decode messages. A cipher is an algorithm or mathematical formula used to encrypt and decipher messages. The cipher's input parameter known as a variable or key is necessary for it to operate correctly. A variable is an essential component of the algorithm for a cipher to function. A cipher's output is made distinguishable by a variable referred to as a key. An intruder who intercepts an encrypted message must determine the cipher that the sender employed to encrypt it and the keys that were used as variables. Encryption is a very useful security tool because it takes time and is challenging to guess this information. Sensitive information has long been safeguarded via encryption. Historically, militaries and governments have employed it. Data stored on computers and other storage devices, as well as data being transmitted across networks, are all protected by encryption in the current era [12]. Although there are many uses for encryption, its primary goal is to protect the privacy and security of data. Data is encrypted by being converted into a code or cipher that is challenging to crack without a unique key or password. A few of the primary justifications for using encryption are listed below:

- *Confidentiality and privacy*: Encryption can aid in preventing unauthorized access to sensitive and private information. Only those with a password or key can decrypt data, preventing unauthorized access.

- *Security*: Protection from data breaches and hacker attempts is made possible through encryption. Without the key, an attacker would not be able to decrypt any encrypted data they might manage to intercept.

- *Compliance*: Certain sectors, like the financial and healthcare sectors, are mandated by law to use encryption to safeguard sensitive data.
- *Authentication*: A message's or sender's legitimacy can also be confirmed using encryption. To confirm that a message is authentic and from a reliable source, for instance, digital signatures are utilized.

The most appropriate encryption method to utilize will depend on the particular use case and the needed level of security. There are numerous encryption strategies available. Here are a few popular encryption methods and how they operate:

- *Symmetric-key encryption*: This method encrypts and decrypts data using the same secret key. Both the transmitter and the receiver encrypt and decrypt the data using the same key. The Advanced Encryption Standard (AES), Data Encryption Standard (DES), and Triple DES are symmetric-key encryption techniques.
- *Asymmetric-key encryption*: This method encrypts and decrypts data using a pair of keys—one public and one private. Data is encrypted using the public key and decrypted using the private key. Algorithms with asymmetric keys include RSA and Elliptic Curve Cryptography. (ECC).
- *Hash functions*: To create a fixed-length output from an input message, hash functions are utilized. A hash or message digest is the output that is specific to the input message and cannot be reversed to recover the original message. Hash functions are frequently used to produce digital signatures and to check the accuracy of data. The hash functions SHA-256 and MD5 are two examples.

1.4 Motivation

Global healthcare is significantly impacted by situations like COVID-19. Remote monitoring provided by smart healthcare technologies was very beneficial to both doctors and patients. Smart technologies help to keep the medical record and provide real-time notifications and emergency assistance, such as wearable health bands, fitness shoes, RFID-based smartwatches, and smartphone apps. Smart devices need a variety of sensors and actuators with information-perceiving capabilities in order to connect to the Internet and communicate with the outside world. The adaptability of the health apps to help with contact tracking, self-testing, quarantine monitoring, and social distance maintenance is easily adjustable, acceptable, and accessible.

Despite having a key role in patient diagnosis, SHS faces a number of security and privacy problems. Attackers or hackers might cut the connection and put the patient's life in peril voluntarily. Some issues related to modern technology are covered in SHS. So, we looked into potential threats that might exist in the SHS and 5G and how they might appear. These issues encouraged us to conduct research using the SHS network's integration of 5G security. So, we intended to use threat modeling to determine the threats. We came up with the notion to apply encryption techniques to increase the security of the data in SHS.

- What prerequisites must be satisfied before 5G may be used on the SHS network?
- To alleviate security issues in the SHS, we require a strong authentication system to verify each and every end user. How is 5G security used to SHS security, and how does it offer a strong network authentication?
- What limitations might be imposed on the roll-out of 5G?

These elements motivated us to carry out this investigation.

1.5 Aim and Objectives

1.5.1 Aim

The purpose of the thesis is to explore and assess the potential security risks in Smart Healthcare Systems (SHS) using Threat Modelling, and to suggest a new encryption strategy to improve the security of SHS employing the capabilities of 5G technology. By investigating the security risks already present in SHS, suggesting the addition of a new encryption method using AES-CCM with ECDH using 5G.

1.5.2 Objectives

- Explanation of the implementation of 5G in the context of a smart healthcare system.
- Assess the risk for integrity, confidentiality, and privacy.
- Attenuation plans for the problems identified by threat modeling.
- Propose improvement to the current security measures by introducing additional encryption between SHS and 5G.

1.5.3 Research Questions

- Which risk factors can be identified by the SHS while conducting the STRIDE threat analysis?
- Which security and privacy controls could be utilized to address the threats outlined in the STRIDE threat in 5G Smart Healthcare Systems?
- How we can enhance 5G AKA to offer optimal safety and privacy in smart healthcare systems?
- How can AES-CCM with ECDH be implemented in an SHS to ensure secure communication between healthcare providers and patients?

1.6 Research Mecthology

1.6.1 Literature Review

We conducted a a brief study to gather some basic background information regarding the Smart Healthcare system, 5G technologies, encryption methods, etc. We took into consideration a number of journal articles, conference papers, and 5G-AKA authentication mechanisms based on issues with smart healthcare systems, as well as encryption methods using AES-CCM and ECDH. Among the digital libraries where these publications are searched are ACM Digital Library, Scopus, IEEE Xplore, and Google Scholar. We subsequently developed an encryption framework to carry out the additional encryption utilizing AES-CCM and ECDH between the SHS and 5G network using the STRIDE modeling approach on SHS with 5G-AKA.

Inclusion criteria:

- Articles should be in English.
- Articles can be conference papers, journal articles, or books.
- Articles that are related to the current research are considered.

1.6.2 Threat Analysis

After collecting some background information from the literature review, we contemplated using the STRIDE approach for the Threat modeling. The Stride Threat modeling can be used to methodically iterate through the different threat scenarios for the security risk assessments. By defining the responses to the threats that the STRIDE modeling identifies and that the suggested authentication strategy can be used to address.

1.6.3 Case Study

The Threat analysis gives a basic understanding of the threats, which are held in the Smart Healthcare System. With the use of 5G networking in the SHS, there are certain limitations and to ensure these limitations strong encryption is needed. So, we are experimenting with encrypting SHS data using AES-CCM and ECDH over a 5G network. Observe and address the results, and how the threats are mitigated.

1.7 Outline

Chapter 1, is the introduction to the research, where the thesis is introduced. The Background, which provides the fundamental ideas needed to comprehend the research, the Aim and Objectives, and the Research questions chosen for the study are all included. Chapter 2, which is related work, where talks about the comprehensive overview of existing literature and research on the SHS. Chapter 3, refers to the Methodology and Technologies, To describe the workflow and all the methods which refer to analyzing the threats and mitigating solutions for the threats. Threat Analysis, where we describe about the Threat modeling by the STRIDE threat modeling.

Described the Lightweight Encryption and given the analysis of various encryption techniques and Key exchange protocols. Chapter 4, which refers to Implementation,

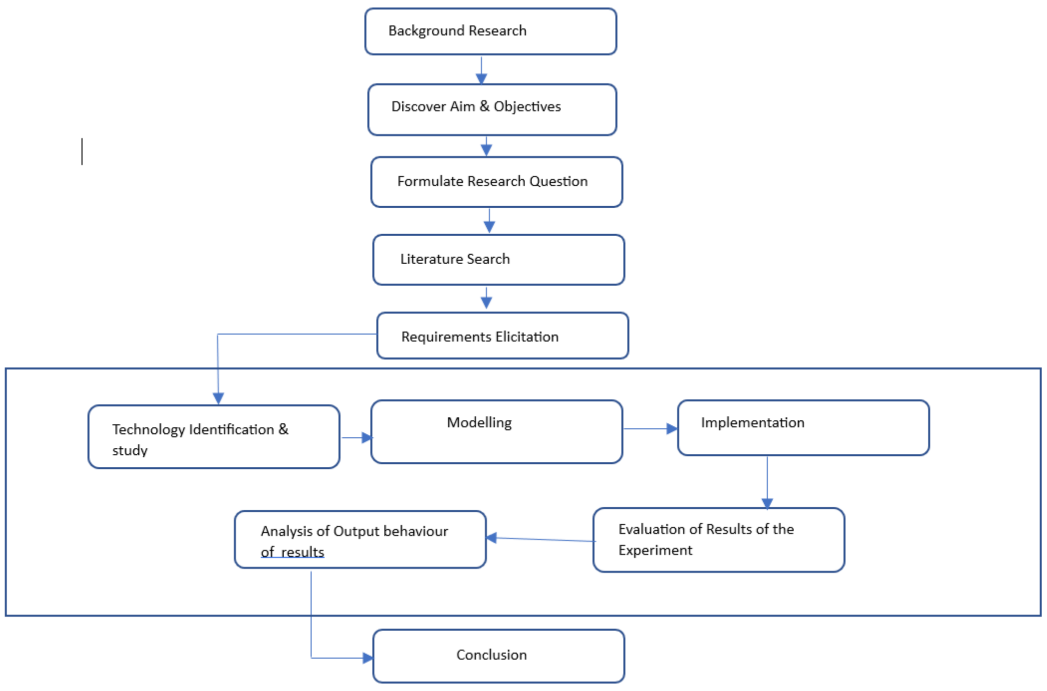

Background Research
Discover Aim & Objectives
Formulate Research Question
Literature Search
Requirements Elicitation
Technology Identification & study
Modelling
Implementation
Analysis of Output behaviour of results
Evaluation of Results of the Experiment
Conclusion

Figure 1.1: Research Flow.

where discusses the implementation of the experiment we conducted for the encryption of the Smart Healthcare Systems over the 5G network i.e., using AES-CCM and ECDH. Chapter 5, discuss the various experiments on different scenarios, and the outputs of the experiments are also discussed. Chapter 6, which refers to the Discussion, here we talk about the answers to the Research Questions for the thesis and the Contribution to the Thesis. Chapter 8, which refers to the Conclusion and Future Work, talks about the conclusion and discusses the future work of the thesis.

Chapter 2

Related Work

2.1 Research Overview

These authors Liu and Hong etc, [13], suggested a privacy protection system for smart healthcare systems. The innovative perturbed position generation technique, which guarantees privacy while keeping the necessary level of precision, forms the basis of this suggested system. The authors tried to construct their system using trusted third parties and location perturbation approaches because it is crucial to protect the patient's location. The major goal of his work is to make sure that no adversary may access and/or analyze a patient's location information, which is sent through Memory Protection Unit, to connect it with his sensitive places. The proposed method accomplishes this by adaptively altering the amount of location perturbation by taking into account how far away the actual location is from the sensitive Locations

The main processing unit attached to the patient's body generates the perturbed location by taking into account the distance between the patient's location and the previously identified (patient's sensitive locations). As opposed to other dummy locations, this method generates dummy locations that are farther from the patient's original location. Disturbed locations are to hospitals with a time delay to help support this improved protection mechanism and provide greater real-time location privacy protection. To retain privacy from others, the algorithm that is suggested is primarily utilized to construct fictitious locations solely instead of emergency circumstances (attackers).

The challenges with edge and cloud computing-based smart health were identified by the authors Natgunanathan and Iynkaran in [14], and a cooperative privacy preservation system was created for portable wearable technology. Authentication protocols are created to achieve data privacy preservation during authentication and access control while taking into account the specific security requirements in the edge and cloud computing modes. The suggested scheme relates to pursuing privacy for wearable devices during hybrid computing to maintain secure interactions. MinHash is applied to identify the similarity of different patient data fields without disclosing sensitive information by ciphertext policy attribute-based encryption is applied for remote data access control, and a bloom filter is applied to determine whether a value exists without secret exposure. The proposed scheme satisfies security properties, including data confidentiality and integrity, mutual authentication, forward security, and privacy preservation. But has some time taking process to get access to that wearable device.

These [15] authors Sun and Jianfei etc., proposed a Lightweight and privacy-aware fine-grained access control-based (LPAC-based) Ciphertext policy attribute-based encryption (CP-ABE) scheme designed for the IoT-oriented smart health system. LPAC-based CP-ABE scheme supports stronger attribute privacy protection, lightweight and fine-grained access policy, online/offline encryption, and efficient decryption. In the proposed mechanisms, the data users (doctors, nurses, etc) use a public key and security key over the trusted authority from the data owner (patients) and healthcare cloud server. The mechanisms which are based on their schemes seeking attribute revocation, traceability, and CCA-based mechanisms based on the proposed scheme are considered the future work of the researchers for better improvements. As this, the proposed scheme had issues related to the traceability problem regarding the data owners i.e., patients.

In this [16] article the author's Tang and Xiaoyong etc., suggested a 5G-based design for the healthcare dedicated network and other smart healthcare information infrastructure. The design uses the iGW as its central component and optimizes the most recent technical architecture regarding MEC 5G Integration that has been specified by 3GPP and ETSI for use in a smart healthcare scenario. Additionally, the network architecture supports intra-hospital, inter-hospital, and extra-hospital application scenarios for smart healthcare. Different apps can be launched and maintained in the 5G healthcare cloud by utilizing the new 5G-based medical information infrastructure that is made up of the dedicated network and cloud. where the 5G basic application clearly shows how 5G improves hospital production efficiency and 5G transmission capabilities. The goal of the medical applications is to create a full-scenario smart medical service ecosystem for telemedicine, emergency rescue, smart hospitals, and other applications in a variety of medical situations, including in-hospital, inter-hospital, and out-of-hospital.

In this [17] work, In order to accomplish the following objectives: rapid and accurate context-aware health situation identification; a secure data sharing mechanism based on blockchain; and responsive and low-latency services for urgent patients, authors have proposed a framework for 5G-secure-smart healthcare monitoring. Here are a few benefits of the suggested framework:

- A 5G-IPv6 network design can improve mobility support for wearable sensors by reducing latency.
- Based on similarity measurements in the edge cloud, the framework |immediately recognizes health problems and initiates advanced health services to prevent and manage cardiovascular disease. MLP + BN can produce the ideal weight values for similarity metrics.
- Patients and healthcare professionals can securely share health information via a blockchain-based secure data exchange method. The following stage is to better enhance the Quality of Service settings for identifying context-aware health situations, especially when there is extensive data gathering going on. The framework in the application can be strengthened for real-time edge situations.

The field of secure data transmission in smart healthcare systems has received significant attention from researchers in recent years. In this section, we review previous works related to the use of AES with ECDH in smart healthcare systems using 5G AKA with the Stride model.

The use of AES in healthcare systems to ensure the confidentiality of patient data has been explored by several researchers. For instance, Yang et al. [18] proposed a secure data-sharing scheme based on AES for cloud-based electronic health records. Their scheme allowed patients to share their records with healthcare providers while ensuring that the data remained confidential. Similarly, Zhang et al. [19] proposed an AES-based data encryption scheme for mobile health systems. Their scheme ensured that patient data remained confidential during transmission between the mobile device and the healthcare provider.

ECDH has also been used in healthcare systems to establish secure communication channels between patients and healthcare providers. For example, proposed a secure data transmission scheme based on ECDH for mobile health systems. Their scheme ensured that the communication channel between the mobile device and the healthcare provider remained secure.

The Stride model has been proposed as a way to ensure the confidentiality, integrity, and availability of data in healthcare systems. In the context of smart healthcare systems, the Stride model has been used to ensure that patient data is transmitted securely between the mobile device and the healthcare provider. For example, proposed a secure data transmission scheme based on the Stride model for mobile health systems. Their scheme ensured that patient data remained confidential, accurate, and available to authorized parties.

2.1.1 Comparison with Existing Work

A comparison of the encryption protocols used in smart healthcare systems reveals that DHE is vulnerable to man-in-the-middle attacks, which can compromise the confidentiality and integrity of the communication. In addition, DHE can be computationally expensive, particularly for resource-constrained devices used in smart healthcare systems. These factors make DHE less suitable for use in smart healthcare systems that require high security and efficient communication. [20] Previous research has identified the challenges associated with implementing DHE in smart healthcare systems. These challenges include the vulnerability to man-in-the-middle attacks, the computational overhead, and the complexity of managing and distributing keys. In contrast, research on implementing AES with ECDH in healthcare systems has shown that it is a more efficient and secure solution for key exchange. Case studies of implementing AES with ECDH in healthcare systems have demonstrated its effectiveness in securing data transmission. However, challenges still exist, such as key management and distribution, and there is a need for further research to address these challenges. [21]

Chapter 3

Methods and Technologies

3.1 Smart Healthcare Systems

This architecture of the Smart Healthcare system gives the main knowledge about the efficient, effective, and secure delivery of healthcare services made possible through SHS. A decision can be made by the SHS based on the patient's observed conditions and body temperature, heart rate, and pulse. Due to the fact that it does not constantly switch on all of the sensors, its architecture is also an energy-efficient solution. The system's algorithm will manage the utilization of the sensors and regulate their price and lifespan. [22]The study's suggested smart healthcare monitoring and patient management system includes communication channels, embedded internal and exterior sensors, an IoT server, cloud storage, and gateway support. Body temperature, pulse rate, and heartbeat sensors make up the system's three sensors. The Arduino board connects these three sensors in order to gather and classify medical data. Devices for networking and communication control data transport. The fuzzy logic system is employed in this arrangement to enable decision-making, with data analytics providing decision-making capabilities. The doctor's view gives hospital professionals the ability to observe and speak with the patient at a distance [23].

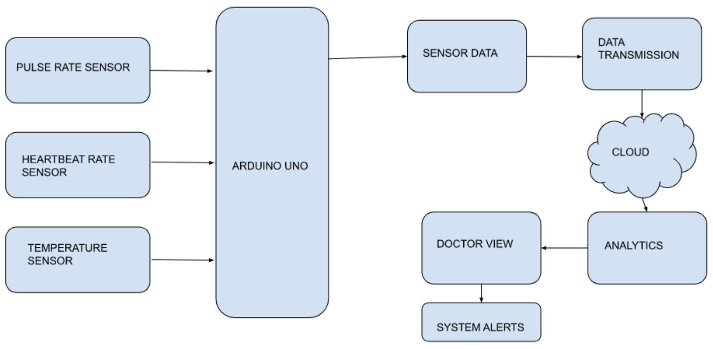


Figure 3.1: Architecture of Smart Healthcare Systems.

3.1.1 Security and Privacy in SHS

Numerous advantages, including better patient outcomes and more effective healthcare delivery, are possible with smart healthcare systems. They also present special privacy and security issues, which demand attention if patient data is to be kept secure [24] [25] [26]. In order to handle security and privacy in smart healthcare systems, consider the following:

- *Secure data storage*: Modern healthcare systems produce and save a lot of private information about patients, such as medical records and personal data. To stop theft, illegal access, and data breaches, this data must be stored safely. Using security to safeguard the data both in transit and at rest is one approach to accomplish this.

- *Access controls*: By ensuring that only authorized individuals have access to patient data, access controls can aid in preventing illegal access to it. This can include security precautions like secure passwords, and role-based access limits.

- *Education and training for users*: It's critical to inform healthcare professionals and personnel about security best practices and the dangers of using smart healthcare technologies. This can enhance the overall security posture and help avoid unintended data leaks.

- *Regulation compliance*: Regular security audits and testing can help find potential vulnerabilities in smart healthcare systems before they can be exploited. Penetration testing, vulnerability scanning, and other security evaluations fall under this category.

- *Regular security audits and testing*: In order to find potential vulnerabilities in smart healthcare systems before they can be exploited, regular security audits and testing are helpful. Penetration testing, vulnerability scanning, and other security evaluations fall under this category.

3.1.2 Comparison of SHS device and other Healthcare devices

Smart healthcare device offers convenient, ongoing monitoring and smoothly integrates into daily routines. They offer connectivity for remote access to health information and real-time data transmission. They enable people to enhance their well-being by providing individualized insights and recommendations. These reasonably priced tools assist in the early diagnosis of health problems, but they should be used in conjunction with expert medical guidance.

ATTRIBUTE	SMART HEALTHCARE DEVICE	OTHER HEALTHCARE DEVICE
Medical Monitoring	Designed for medical monitoring purposes	General-purpose devices
Connectivity	Equipped with wireless connectivity capabilities	May or may not have wireless connectivity
Data Accuracy and Reliability	Built with advanced sensors and algorithms for accurate and reliable data collection	Reliability may vary depending on the device
Health-focused Functionality	Designed with health-related features and functions	Designed for general purposes
Integration with Healthcare Systems	Can integrate with healthcare systems and electronic health records (EHRs)	Not specifically designed for healthcare integration
Health Data Privacy and Security	Prioritize privacy and security of personal health information	Privacy and security measures may vary
Health Insights and Analysis	Provide data analysis and insights for better understanding and management of health	Not specifically designed for health analysis
Remote Healthcare Support	Enable remote monitoring and virtual consultations with healthcare professionals	Not designed for remote healthcare support
User-Friendly Interfaces	Feature user-friendly interfaces for easy operation	User-friendliness may vary
Specific Medical Applications	Cater to specific medical applications and conditions	General-purpose devices without medical focus

3.1.3 Threats existing in SHS

Threats to Smart Healthcare Systems (SHS) can jeopardize both their performance and security. which may endanger the lives of patients and cost healthcare providers money in downtime. The widespread use of SHS might also raise challenges with data privacy and moral considerations with the gathering, storing, and utilization of personal health information.

- *User Impersonation Attack*: By making false message requests to the primary server, the attacker hopes to carry out an impersonal attack by taking the position of a trusted user. Because time stamps and identities are always verified at the SHS, the attacker can fabricate hidden identities and credentials.

- *Device Impersonation Attack*: Attempt to send a verification message if any other user wants to obtain the data from a device connected to the SHS network.

Additionally, the attacker creates the timestamps and attempts to submit the request to the user end using the SHS's name. The user checks the attackers' supplied authentication request. However, no actual keys are used in the computation of the secret key or any time stamp. However, the attackers can circumvent security.

- *Forward Secrecy Attack*: If another user from outside the SHS network tries to intercept communications in order to get the secret keys. To create the session key, the user needs smart card identification and random nonce.

- *Man-in-the-middle attack*: Using the secret key that was supplied to the attacker's home server, the request is delivered if the attempt to fabricate the information is successful. The SHS then checks the time-stamp and computes whether the keys are correct, continuing the session if it is, otherwise discarding. After receiving the request at SHS, it validates the session key id and verifies the session key, which is the SHS session key. If the adversary manages to evade the verification process, the actual user may experience message delays, informational changes, and a lack of information about user usage (such as logins by attackers using other people's data).

- *Untraceable Attack*: If the attacker keeps track of the communications between the trusted user and SHS. However, since all of the messages are computed utilizing pseudo-identity and random number creation, the attacker is unable to do eavesdropping. By including a timestamp with each packet that is requested, communication is protected from these types of intrusions while maintaining anonymity and the danger of untraceability.

- *Replay Attack*: The attacker can listen in on all conversations, copy the secret parameters, and then launch a replay attack by submitting all user IDs that are also saved on their smart cards at the same time to the SHS. The replay attack, however, cannot be made by confirming the message's timestamp and it cannot be transmitted further.

- *Password change attack*: A password and the user's and device's distinct identities are part of the user's smart card. The unique Identity code, the biometric identity, and the secret password are virtually impossible for an adversary to intercept and modify. The user's identification number, which is only known to the trusted user, is the source of the identity code. Another scenario is where the adversary unlawfully obtains the Identity code and needs the user's biometrics to proceed.

- *Eavesdropping*: The biggest security danger that is held in between the transmission medium is intercepted by attackers because the communication medium utilized by IoT devices is wireless. By superimposing jamming signals, the quality of data transferred between nodes in a wireless media may be diminished. Here, accurate data storage requires secure encryption and cryptographic techniques.

3.2 Threat Analysis

3.2.1 Stride model

To evaluate and resolve security threats in software systems, the STRIDE framework is frequently used in threat modeling. The letters in the acronym STRIDE stand for several threats that may jeopardize the confidentiality, integrity, or availability of data stored on a system or its operation. While tampering entails altering data or software to change its behavior or jeopardize its security, spoofing entails the impersonation of a user or system component to gain unauthorized access. While information disclosure refers to the unauthorized release of private information, repudiation describes situations in which a user disputes having carried out a specific action. Elevation of privilege refers to the process by which user permissions are increased so they can access resources or do activities that are outside the scope of their allowed access, whereas denial of service (DoS) attacks aim to prevent the availability of a system or network [27].
Developers can find possible vulnerabilities in software systems and take advanced steps to mitigate them by methodically applying the STRIDE methodology to the system. With this strategy, the risk of an attack is decreased by scrutinizing each system component, identifying any potential vulnerabilities, and putting the necessary security measures in place. Based on the seriousness and likelihood of potential threats, STRIDE can assist developers in prioritizing security requirements. Developers can create more secure systems and reduce the chance of security breaches, which can cause considerable monetary, legal, and reputational harm, by implementing this paradigm during the software development lifecycle.

In SHS, the threat of security breaches and attacks on patient data and healthcare infrastructure is a significant concern. To detect and prevent such threats, the Stride model can be used as a technique for threat analysis. By carefully evaluating the confidentiality, integrity, and availability of patient data and essential healthcare services, the STRIDE approach can help detect and mitigate potential security vulnerabilities in Smart Healthcare Systems (SHS). Spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege are the six categories into which potential security risks are divided in order to achieve this. Developers can assess the security risks in an SHS and take the necessary steps to mitigate them by taking into account each of these areas. For instance, if sensitive patient data is not adequately protected or if there are weaknesses in the system that might be exploited, a SHS that collects and keeps that data may be subject to information disclosure. Developers may thoroughly assess the security risks in a SHS and take appropriate action to solve them by taking into account each of the six threat categories in the STRIDE framework. Overall, developers can create more secure SHS that defend patient data and essential healthcare services from potential security threats by implementing the STRIDE framework throughout the software development lifecycle. [28]

The Stride model is a framework for identifying potential security threats by analyzing data and evaluating it across six dimensions:

- *Spoofing*: Spoofing occurs when an attacker poses as an authorized user or device to gain access to sensitive information or systems. This can be achieved through various means, such as by stealing passwords or credentials, using fake identities, or manipulating network traffic.

- *Tampering*: Tampering refers to the unauthorized modification of data, devices, or systems. This can include altering information, changing the functionality of a device, or disrupting communication between devices.

- *Repudiation*: Repudiation occurs when an attacker denies or rejects the authenticity or validity of a message or action, which can create ambiguity in data. This can make it difficult to determine what actually happened and who is responsible.

- *Information Disclosure*: Information disclosure refers to unauthorized access to or release of sensitive data or information. This can include personal health information, financial data, or other sensitive data that should be kept confidential.

- *Denial of service*: Denial of service (DoS) refers to a malicious attack that disrupts or prevents access to a system or network. This can be achieved through various means, such as flooding the network with traffic, exploiting vulnerabilities in software or hardware, or overloading a server.

- *Elevation of privilege*: Elevation of privilege refers to an attacker gaining unauthorized access to resources or privileges beyond their intended level of access. This can include accessing sensitive information, modifying critical system settings, or impersonating an authorized user.

In the context of SHS, wearable sensors and devices generate large amounts of data that need to be analyzed for threat detection. The Stride model can be used to analyze this data and identify potential threats by evaluating it across the six dimensions mentioned above. For example, abnormal data patterns or spikes in activity could indicate potential DoS or tampering attacks, while unauthorized access or modification of patient data could indicate spoofing or information disclosure threats.

By using the Stride model, SHS can improve its threat detection capabilities and take proactive measures to prevent security breaches and attacks. Additionally, the Stride model can be integrated with machine learning algorithms to automate the threat analysis process and improve the accuracy of threat detection. [29]

3.2.2 STRIDE Threat classification in SHS

There are so many methodologies that have been used while threat modeling (STRIDE, DRIDE, PASTA, VAST, CVSS, Attack Trees, etc). The right threat model for your needs depends on what types of threats you're trying to model and for your purpose. [30]Upon considering the model of Smart Healthcare systems architecture, STRIDE threat modeling is taken into view, which explains the workflow to analyze the threats and to find the mitigations for the threats.

Threat Classification	Cause	Desired Property	Affected Attribute of SHS
Spoofing	By looking at the authentication processes in place, such as username/password systems, to make sure that only authorized users are given access, spoofing threats can be found in SHS	Authenticity: Strong authentication measures can be used to apply authenticity to avoid spoofing in SHS. Implementing security measures like digital certificates and cryptographic keys can also assist in confirming the veracity of user identities and stop attackers from posing as authorized users.	Connectivity, Data privacy, and security, Integration with Healthcare systems, Health data privacy and security, Remote Healthcare support, User-friendly Interface.
Tampering	By checking the accuracy of the information and software used to manage patient data and healthcare services, tampering hazards can be found in SHS. Additionally, keeping an eye out for odd changes in data or system behavior can aid in the early detection of potential tampering threats.	Integrity: By using techniques like hash functions, and checksums to guarantee the integrity of the data and software within the system, integrity can be employed to avoid tampering in SHS.	Connectivity, Data accuracy and Reliability, Data security and privacy, Integration with The healthcare system, Health data privacy and security Health Insights and Analysis, Remote Healthcare support and User-friendly Interfaces.
Repudiation	By adding tools to detect and document user activities, like as audit trails and digital signatures, to make sure that actions taken by users within the system can be traced back to them, repudiation threats can be identified in SHS.	Non-Repudiation: By incorporating techniques like audit trails to offer proof of user behaviors within the system that cannot be repudiated, non-repudiability can be used to avoid repudiation in SHS. Repudiation attacks can also be avoided by putting in place safeguards such as access controls and permissions that restrict users' capacity to take actions that they can subsequently retract.	Data Accuracy and Reliability
Information Disclosure	By regularly doing vulnerability assessments and penetration tests to find any potential security holes that an attacker could exploit, Information Disclosure hazards can be found in SHS.	Confidentiality: By putting in place safeguards like encryption, access controls, and data masking strategies to make sure that sensitive patient data is safeguarded and only accessible by authorized users, confidentiality can be employed to prevent Information Disclosure in SHS.	Connectivity, Data privacy security, Integration with the Healthcare Systems, Health data privacy security, Health Insights and Analysis, Remote Healthcare and support, Specific medical applications.
Denial of Service	By monitoring network traffic and system performance for any unusual behavior or spikes in traffic that could signify a DoS attack, Denial of Service threats can be found in SHS.	Availability: By putting in place safeguards like redundancy and failover procedures to make sure that essential healthcare services are always accessible to patients and healthcare professionals, availability can be used to prevent Denial of Service (DoS) in SHS.	Connectivity and Remote Healthcare support.
Elevation of Previlage	Adopting stringent access controls and role-based permissions, SHS can detect threats involving the elevation of privilege by making sure that users only have access to the privileges required to carry out their job duties.	Authorization: By using stringent access controls and role-based permissions to make sure that users are only given the privileges necessary to carry out their job tasks, authorization can be used to prevent the EoP attacks.	Connectivity, Data privacy and security, Integration with Healthcare systems, Remote Healthcare support, User-friendly Interfaces

Table 3.1: STRIDE classification

3.3 5G-AKA Architecture

For the 5G core network, a service-based architecture (SBA) has been suggested. In light of this, 5G also defines new entities and service requests [31].

- During the authentication procedure between a UE and its home network, the Security Anchor Function (SEAF), which is in a serving network, acts as a "middleman". Although it has the ability to reject the UE's authentication, it depends on the UE's home network to do so.

- In a home network, the Authentication Server Function (AUSF) handles authentication with a UE. When 5G-AKA is utilized, it decides on UE authentication but depends on a backend service to compute the authentication data and keying materials.

- The Authentication Credential Repository and Processing Function (ARPF), which chooses an authentication method based on subscriber identity and configured policy and computes the authentication data and keying materials for the AUSF if necessary, is one of the data management-related functions hosted by the Unified Data Management (UDM) entity.

- The Subscription Identifier De-concealing Function (SIDF) decrypts a Subscription Concealed Identifier (SUCI) to reveal its Subscription Permanent Identifier (SUPI), such as the IMSI, which is its long-term identity. In 5G, a subscriber's long-term identity is always transferred in an encrypted format across the radio interfaces. More particular, the SUPI is shielded using public key-based encryption. Therefore, the private key linked to a public key issued to UEs for encrypting their SUPIs is only accessible by the SIDF.

- 5G defines new authentication-related services. For example, the AUSF provides authentication service through NausfUE Authentication, and UDM provides its authentication service through NudmUE Authentication (N is the vector). For simplicity, generic messages such as Authentication Request and Authentication Response are used without referring to the actual authentication service names.

- To begin the authentication process of 5G AKA, the SEAF must first receive any signaling messages from the UE. Note that if a 5G GUTI has not been assigned to the UE by the serving network, the UE shall give the SEAF an encrypted permanent identification (a SUCI) or a temporary identifier (5G GUTI). Using the public key of the home network, the SUCI is the encrypted version of the SUPI. As a result, in 5G, a UE's permanent identifier, such as the IMSI, is never sent via radio networks in clear text. Over earlier generations, like 4G, this capability is thought to represent a significant security enhancement.

- SEAF sends an authentication request to the AUSF, which then confirms that the serving network seeking the authentication service is approved before SEAF

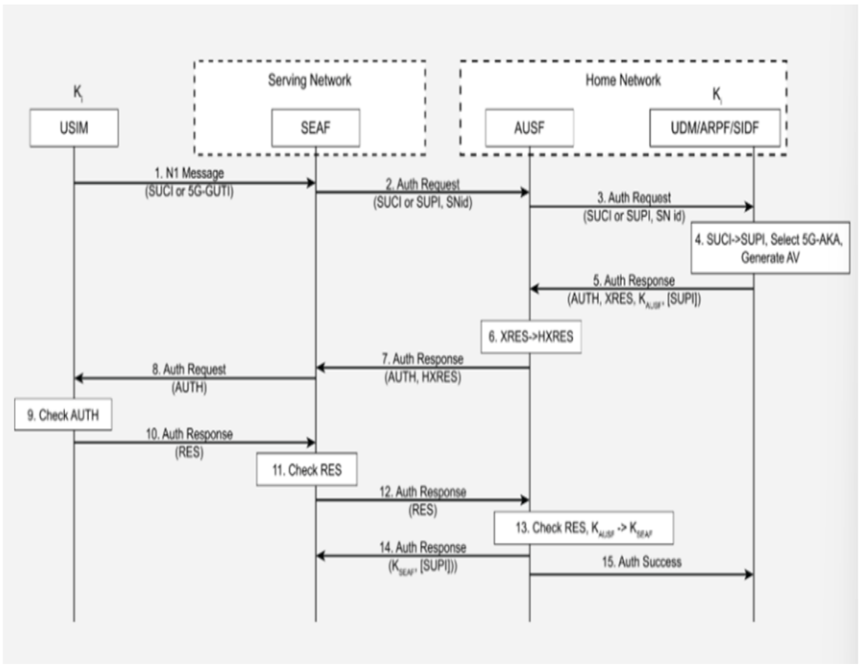

Serving Network
Home Network
K_i
USIM
SEAF
AUSF
K_i
UDM/ARPF/SIDF
1. N1 Message (SUCI or 5G-GUTI)
2. Auth Request (SUCI or SUPI, SNid)
3. Auth Request (SUCI or SUPI, SN id)
4. SUCI->SUPI, Select 5G-AKA, Generate AV
5. Auth Response (AUTH, XRES, K_AUSF, [SUPI])
6. XRES->HXRES
7. Auth Response (AUTH, HXRES)
8. Auth Request (AUTH)
9. Check AUTH
10. Auth Response (RES)
11. Check RES
12. Auth Response (RES)
13. Check RES, K_AUSF -> K_SEAF
14. Auth Response (K_SEAF, [SUPI]))
15. Auth Success

Figure 3.2: 5G-AKA Architecture.

may begin the authentication process. When successful, the AUSF notifies UDM or ARPF of the authentication request. If the AUSF provides a SUCI, the SIDF will be called upon to decrypt the SUCI in order to retrieve the SUPI, which is then utilized to choose the subscriber authentication method that has been configured. In this instance, 5G-AKA has been chosen and will be used.

- When 5G-AKA is initiated, UDM/ARPF sends the authentication response to the AUSF along with an authentication vector that includes, among other things, an AUTH token, an XRES token, the key KAUSF, and the SUPI if appropriate (for instance, when a SUCI is present in the corresponding authentication request).

- The authentication response is sent to the SEAF along with the AUTH token, the AUSF computes a hash of the expected response token (HXRES), saves the KAUSF, and transmits the authentication response. Notably, this authentication response does not send the SUPI to the SEAF. Only after successful UE authentication is it forwarded to the SEAF.

- The UE receives the AUTH token in an authentication request from the SEAF, which also holds the HXRES. Using the private key it shares with the home network, the UE verifies the AUTH token. The UE considers the network to be authenticated if validation is successful. The UE completes the authentication by creating and delivering a RES token to the SEAF, which the SEAF subsequently verifies. If all goes well, the SEAF will next send the RES token to the AUSF for verification. The authentication choice is ultimately made by AUSF, which is located on a home network. The AsUSF computes an anchor key (KSEAF) and delivers it to the SEAF, along with the SUPI if necessary, if the RES token from the UE is valid. The AUSF also informs UDM/ARPF of the authentication results so they can log the events, e.g., for auditing.

- When the SEAF receives the KSEAF, it generates the AMF key (KAMF), sends the KAMF to the nearby Access and Mobility Management Function, and promptly deletes the KSEAF (AMF). The Next Generation NodeB (gNB) base station will then use the key KgNB to derive the keys needed to protect subsequent communication between the UE and gNB. The AMF will then derive from the KAMF (a) the confidentiality and integrity keys needed to protect signaling messages between the UE and the AMF, and (b) another key, KgNB. The root of the key derivation hierarchy, the long-term key, is held by the UE. A shared set of keys between the UE and the network can be obtained by the UE by deriving all the preceding keys.

3.3.1 Working of schemas in 5G-AKA

The 5G AKA (Authentication and Key Agreement) protocol is used in 5G mobile communication networks for authentication and key agreement between a mobile device and the network [32] [33]. The protocol involves several steps and can be explained using the following schema:

- *Initialization*: The mobile device and the network exchange their identities and initial parameters to begin the authentication process. In 5G, this step involves the exchange of a "SUPI" (Subscription Permanent Identifier) and "Ks-NN" (a network-specific key) between the mobile device and the network. For example, Let's say you have a 5G-enabled smartphone and you want to connect to a 5G network. When you try to connect, the network will ask for your subscriber information, which includes a unique identifier called a "SUPI" and a network-specific key called "Ks-NN". These are exchanged in the Initialization step.

- *Random Challenge*: The network sends a random challenge, called "RAND", to the mobile device, which the device must use to generate a response. This challenge is used to prevent replay attacks. For example, Next, the network sends a random challenge, called "RAND", to your smartphone. This is a unique code that changes every time you connect to the network, and it's used to prevent someone from intercepting your response and replaying it later to gain access to the network.

- *Response*: The mobile device generates a response to the challenge using its secret key, called "Ks", and sends it back to the network. The response also includes the "SUPI", a temporary identifier called "SUCI" (Subscription Concealed Identifier), and a "MAC" (Message Authentication Code) to ensure the integrity of the response. For example, Your smartphone then generates a response to the challenge using a secret key called "Ks". This response includes your SUPI, a temporary identifier called "SUCI", and a "MAC" to ensure the integrity of the response. The response is sent back to the network.

- *Authentication*: The network authenticates the mobile device based on the response it received. If the authentication is successful, the network sends the mobile device a shared secret key, called "Kseaf" (Ks for the Encryption Algorithm with Freshness). For example, The network verifies your response and authenticates your smartphone based on the information provided. If the authentication is successful, the network sends your smartphone a shared secret key called "Kseaf". This key is used to generate a session key for secure communication between your smartphone and the network.

- *Key Agreement*: The mobile device and the network use the shared secret key "Kseaf" to generate a session key for secure communication. The session key is used to encrypt and decrypt all subsequent messages between the mobile device and the network.

- *Security Mode*: Once the session key is established, the mobile device and the network enter into a security mode where they use the session key to encrypt and decrypt all subsequent messages. For example: Once the session key is established, your smartphone and the network can communicate securely, using the session key to encrypt and decrypt all subsequent messages. This is the Security Mode.

The 5G AKA protocol is designed to be more secure than previous versions of the AKA protocol used in older mobile communication networks. It includes additional security features, such as the use of "SUCI" and "MAC", to ensure the privacy and integrity of the authentication process. Additionally, the use of "Kseaf" allows for more secure key agreement between the mobile device and the network.

3.3.2 5G AKA schema usage in SHS

The AKA protocol can be used in a healthcare system to provide secure authentication and key agreement between a patient's mobile device and the healthcare network [34] [35]. This can be useful in several scenarios, such as when a patient needs to securely access their medical records or when transmitting sensitive medical data.

- *Initialization*: When a patient first registers with the healthcare system, their mobile device, and the healthcare network exchange their identities and initial parameters to begin the authentication process. The patient may need to provide some identifying information, such as their name and date of birth, to verify their identity.

- *Random Challenge*: When the patient needs to access their medical records, the healthcare network sends a random challenge to the patient's mobile device. This challenge is unique to the current session and is used to prevent replay attacks.

- *Response*: The patient's mobile device generates a response to the challenge using their secret key and sends it back to the healthcare network. The network verifies the response and proceeds to the next step if it is correct.

- *Authentication*: The healthcare network authenticates the patient based on the response it received. If the authentication is successful, the network grants the patient access to their medical records.
- *Key Agreement*: The patient's mobile device and the healthcare network use the shared secret key to generate a session key for secure communication. The session key is used to encrypt and decrypt all subsequent messages between the patient's mobile device and the healthcare network.

Using the AKA protocol in this way can help to ensure that patient data is protected and secure. It can also help to prevent unauthorized access to sensitive medical information.

3.3.3 Limitations of usage for 5G-AKA in SHS

The 5G AKA protocol is a security protocol used in 5G networks to provide mutual authentication and establish a secure communication channel between the user equipment (UE) and the core network [36]. However, like any other security protocol, the 5G AKA protocol has some limitations, including:

- *Lack of forward secrecy*: The 5G AKA protocol does not provide forward secrecy, which means that if an attacker gains access to the long-term keys used for authentication, they can decrypt past and future communication.
- *Vulnerability to man-in-the-middle (MITM) attacks*: The 5G AKA protocol is susceptible to Man-in-the-Middle (MITM) attacks, wherein an attacker intercepts the communication between the User Equipment (UE) and the network. By modifying the messages, the attacker can gain unauthorized access to the system.
- *Potential for denial-of-service (DoS) attacks*: The protocol is susceptible to DoS attacks, where an attacker can flood the network with fake authentication requests, causing a denial of service to legitimate users.
- *Complexity*: The 5G AKA protocol is complex and requires a significant amount of computational resources, which can increase the processing time and delay the authentication process.

Overall, the 5G AKA protocol is a robust security protocol, but it has some limitations that need to be addressed to ensure the security and privacy of the 5G networks.

3.4 Limitations of Smart Healthcare Devices

The battery life and processing power of smart medical devices, such as wearable health monitors, are similarly constrained. Some of these restrictions consist :

- *Limited battery life*: Rechargeable batteries, which may have a finite battery life, are used in many smart healthcare equipments. This could be a problem if the battery needs to be replaced regularly or if it dies while the device is being used, which could result in data loss or inaccurate readings.

- *Computation power*: Because some smart healthcare devices may not have sufficient processing capacity, data processing may be sluggish or delayed. This can become an issue if the device needs to process data quickly or give the user feedback in real-time.

- *Technical difficulties*: Technical issues may arise because smart healthcare gadgets can be difficult to use. The data produced by these devices may be difficult to utilize and comprehend for non-technical people, which could result in mistakes in diagnosis or treatment.

- *Cost*: The expense of smart healthcare equipment can prevent those on a low budget from using them. Because individuals who cannot afford these devices have less access to accurate and timely health information, this can lead to health inequities.

- *Security and Privacy*: Sensitive health information is gathered by smart healthcare equipment, which makes them vulnerable to data breaches or cyberattacks. Information about patients' privacy is put at serious danger by this.

In summary, additional encryption is a critical component of securing medical data transmitted through smart healthcare devices and 5G networks. It protects sensitive data from unauthorized access, ensures confidentiality, integrity, and availability of the data, maintains patient privacy, and helps prevent cyber attacks and data breaches.

3.5 Comparison between Encryption techniques

It's critical to compare block cipher and lightweight encryption in order to comprehend the advantages and disadvantages of each algorithm and decide which one is better suited for a given use case or application. It enables the selection of the best cryptographic solution based on considerations such as security needs, resource limitations, and performance [37] [38].

Attribute	Block Cipher	Lightweight Encryption
Complexity	Usually, block ciphers are more complicated than lightweight encryption techniques.As fixed-size blocks of data are used by block ciphers, more complicated algorithms are needed to provide secure encryption.	On the other hand, lightweight encryption algorithms are created to be quick and effective, making them appropriate for contexts with limited resources.
Key Size	To attain the same level of security, block ciphers often need larger key sizes than simple encryption techniques. This is due to the fact that block ciphers work with fixed-size blocks of data, necessitating a larger key size in order to prevent key duplication.	On the other side, lightweight encryption algorithms are made to use smaller key sizes to minimize computational cost.
Key	Block ciphers take longer to encrypt and decrypt data than more straightforward encryption techniques.	Due to their simplicity, lightweight encryption algorithms are usually faster than block ciphers. This qualifies them for uses like low-power gadgets and wireless networks that need quick encryption and decryption.
Block Size	Block ciphers work with fixed-size data blocks that might be anything between 64 and 256 bits in size. The performance and security of the encryption are impacted by the block size.	In order to save time and energy, lightweight encryption methods often use lower block sizes.
Round Function	Block ciphers convert plaintext into ciphertext using a round function. Numerous operations, including substitution, permutation, and mixing, are included in the round function.	Less resource-intensive and memory-intensive round functions may be used in lightweight encryption techniques.
Key Schedule	The subkeys that are utilized in each round of encryption in block ciphers are created using a key schedule method. The security of the cipher depends on the key scheduling algorithm.	Lightweight encryption algorithms, may use simpler key schedule algorithms that can be implemented with fewer resources.
Use Cases	Applications requiring secure data storage and transfer, such electronic payment systems and file encryption, frequently use block ciphers.	Lightweight encryption techniques are appropriate for contexts with limited resources, such as wireless networks and low-power devices.

Table 3.2: Comparison for Block Cipher and Lightweight Encryption techniques

In conclusion, lightweight encryption is a viable option for securing medical data transmitted through smart healthcare systems due to its ability to provide strong security with minimal computational overhead. Block ciphers may provide greater security, but their high computational complexity can be a significant disadvantage in resource-constrained devices used in smart healthcare systems.

3.6 Lightweight Encryption

The term "lightweight encryption" refers to cryptographic algorithms and protocols that are specifically created to function effectively on devices with limited resources, such as memory, processing power, and energy. Low-power embedded systems, Internet of Things (IoT) gadgets, and wireless sensor networks are a few examples of these gadgets. Security, performance, and resource utilization are frequently tradedoff during the design of lightweight encryption methods. To protect sensitive data, they must nevertheless offer a high enough level of protection [39].

3.6.1 Benefits of Lightweight Encryption

Using Lightweight Encryption in SHS offers several benefits. They are:

- *Low computational cost*: In comparison to conventional encryption methods, lightweight encryption solutions are intended to be computationally efficient and demand fewer processing resources. This is crucial for resource-constrained smart healthcare devices since they frequently have low processing speeds and short battery lives.
- *Low memory usage*: When compared to conventional encryption methods, lightweight encryption solutions need less memory, which is crucial for small devices with limited memory.
- *Fast encryption and decryption*: For real-time applications like healthcare monitoring and emergency response, lightweight encryption solutions are made to be quick and effective.
- *Small code size*: Since they often require less code, lightweight encryption algorithms are simpler to install and implement on small devices.
- *Standardization*: International bodies like NIST frequently standardize weak encryption methods, which promotes security and interoperability.

3.6.2 Comparison between Lightweight Encryption

To assess the trade-offs between security, performance, and resource utilization of the two encryption algorithms, AES-CCM and SPECK must be compared. Using the unique needs and limitations of the system, this comparison aids in choosing the best encryption method for a certain application [40] [41].

In conclusion, AES-CCM is a better choice for lightweight encryption in smart healthcare systems due to its proven track record and widespread adoption in various applications, including the healthcare industry. While SPECK may have advantages in terms of performance and smaller code size, the security and reliability of AES-CCM make it a more reliable choice for securing medical data.

3.6.3 AES-CCM Encryption

The Advanced Encryption Standard (AES) block cipher's AES-CCM (Advanced Encryption Standard with Counter with CBC-MAC) mode of operation ensures data confidentiality and authenticity. It combines the usage of CBC-MAC for authentication with a counter mode (CTR) for encryption. The plaintext is initially encrypted in AES-CCM mode using the CTR mode, which creates a stream of key-dependent pseudo-random bits that are then XORed with the plain text. The CBC-MAC algorithm, which employs the same key as the encryption, is then used to create a message authentication code (MAC). To ensure validity and integrity, the MAC is

Attribute	AES-CCM	SPECK
Block Size	AES-CCM has a fixed block size of 128 bits and supports key sizes of 128, 192, or 256 bits.	SPECK, has variable block sizes of 32, 48, 64, 96, or 128 bits, and supports key sizes of 64, 96, or 128 bits.
Cryptanalysis	AES-CCM has been extensively studied and is considered to be a secure encryption algorithm.	SPECK, is a relatively new algorithm and has not undergone the same level of analysis and scrutiny.
Implementation	AES-CCM is a widely adopted encryption standard and is supported by many hardware and software implementations.	SPECK, is a newer algorithm and has fewer implementations available.
Key management	AES-CCM uses a key-wrapping technique that allows for the encryption and authentication of keys	SPECK does not provide this functionality.
Security	AES-CCM is a widely adopted encryption standard and has been subject to more scrutiny and analysi s than SPECK. As a result, AES-CCM may be considered a more secure encryption technique in some use cases.	SPECK is not widely adopted as AES-CCM.
Use cases	AES-CCM is commonly used in applications that require strong security and data integrity, such as secure storage and transmission of data.	SPECK, is more suitable for resource-constrained environments

Table 3.3: Comparison of Lightweight encryption between AES-CCM and SPECK

attached to the encrypted data [42]. For secure and authenticated data transfer, wireless communication protocols like IEEE 802.15.4, Zigbee, and Thread frequently use AES-CCM. Other uses include sensor networks, smart grid communication, RFID (Radio Frequency Identification) systems, and other situations where simple encryption and authentication are needed. In comparison to other modes of operation, AES-CCM is regarded as a secure and effective way to use AES. It offers both confidentiality and authenticity with a relatively low overhead. It is important to keep in mind that AES-CCM is not appropriate for all applications and that the security and performance of the system should be taken into account when evaluating AES-CCM [43].

3.6.3.1 Benefits of AES-CCM in SHS

Using AES-CCM encryption in SHS offers several benefits, They are:

- *Confidentiality*: Data is encrypted using the AES encryption method in CTR mode, which produces a stream of random bits that are XORed with the plaintext to produce the cipher text. This means the patient's data in SHS will be converted, this ensures confidentiality.
- *Authentication*: By creating a message authentication code (MAC) and appending it to the ciphertext, AES-CCM offers the authentication. By creating a MAC with the same key and comparing it to the one received, the recipient can confirm the message's legitimacy.
- *Efficiency*: It just takes one pass over the data to perform both encryption and authentication using the effective mode of operation known as AES-CCM. Because of this, it is especially well suited for low-power systems with constrained computing capabilities.
- *Simplicity*: AES-CCM is a straightforward and uncomplicated way of operation that uses a minimal amount of code.

- *Wide adoption*: AES-CCM is a well-known and well-trusted encryption method since it is widely utilized in many applications, including wireless sensor networks, Bluetooth Low Energy, and Internet of Things (IoT) devices.
- *Non-Repudiation*: AES-CCM provides non-repudiation, which means that the sender of the message cannot deny sending the message at a later time. This is achieved by including a timestamp and a unique sequence number in the message.
- *Flexibility*: AES-CCM can be used with a wide range of key sizes, from 128 bits to 256 bits. This allows for flexibility in choosing the appropriate key size based on the security requirements of the SHS.
- *Resistance to attacks*: AES-CCM is resistant to various attacks, such as brute force attacks and known plaintext attacks. This ensures that the data transmitted between healthcare providers and patients remain secure even in the face of an attack.
- *Low overhead*: AES-CCM has a low overhead, which means that it requires minimal additional bandwidth and processing power compared to other encryption algorithms. This makes it ideal for use in SHS, where resources may be limited.
- *compliance*: AES-CCM is compliant with various security standards and regulations, such as FIPS 140-2 and HIPAA. This ensures that SHS that use AES-CCM is in compliance with industry best practices and regulatory requirements.

AES-CCM is an encryption algorithm that provides robust confidentiality, message integrity, authentication, efficiency, and interoperability in SHS. These features make AES-CCM an optimal choice for securing communication between healthcare providers and patients in the SHS environment. The algorithm's secure encryption, message integrity, authentication, non-repudiation, and protection against attacks, along with its low overhead and compliance with industry standards, make it a suitable encryption algorithm for SHS applications.

3.6.4 Comparison between Key exchange protocol

Understanding the security and performance aspects of the two key exchange protocols requires a comparison of ECDH and DHE. Based on elements like security requirements, computational capabilities, and network conditions, this comparison aids in choosing the most suitable key exchange protocol for a given system [17] [44].

Attribute	**ECDH**	**DHE**
Computation Efficiency	ECDH is generally more computationally efficient, especially for key sizes of 256 bits or higher. This is because ECDH uses elliptic curve cryptography (ECC), which involves simpler mathematical operations.	DHE uses the key size of 2048 - 4096 bits. DHE doesn't uses elliptic curve cryptography (ECC) which involves modular arithmetic operations.
Key size	ECDH uses smaller key sizes than DHE for equivalent security levels. For example, a 256-bit ECDH key provides a similar level of security to a 3072-bit DHE key.	DHE typically uses 2048 - 4096 bit keys.
Security level	Both ECDH and DHE provide strong security guarantees, but the required key sizes for equivalent security levels differ. For example, a 256-bit ECDH key provides a similar level of security to a 3072-bit DHE key. However, the security of ECDH depends on the strength of the elliptic curve used, and there have been concerns raised about the security of some curves in the past.	DHE provide strong security using more keys.
Implementation complexity	Implementing ECDH can be more complex. This is because ECDH involves elliptic curve arithmetic, which can be more difficult to understand and implement correctly.	Implementing DHE is less complex because it involves in the modular arithmetic operations.
Hardware support	ECDH is supported by a wider range of hardware devices and platforms, This is because ECDH is a more recent cryptographic technique, and has gained wider adoption in recent years.	while DHE may be less widely supported.
Forward secrecy	ECDH is generally considered to provide stronger forward secrecy guarantees due to the use of elliptic curves. This is because elliptic curves are more efficient at generating new keys, which means that the key agreement can be refreshed more frequently, reducing the window of opportunity for an attacker to compromise the key.	DHE also provides forward seceracy, which means that a compromised private key in the future will not enable an attacker to decrypt past communications.

Table 3.4: Comparison between the key exchange protocol ECDH and DHE

In conclusion, ECDH is a more suitable choice for key exchange in smart healthcare systems due to its ability to provide strong security with shorter key lengths, which is essential for resource-constrained devices used in smart healthcare systems. While DHE may provide greater security, its higher computational complexity can significantly disadvantage these systems.

3.6.5 ECDH Key exchange protocol

Elliptic curve cryptography is used by ECDH (Elliptic Curve Diffie-Hellman), a key agreement protocol, to enable two parties to create a shared secret key across an insecure communication channel. (ECC). It is a modular arithmetic-based version of the original Diffie-Hellman key exchange technique. Each party in the ECDH protocol must create its own public and private keys. The shared secret key, which is used for symmetric encryption and decryption of messages sent back and forth between the parties, is created using the public key. Compared to prior key agreement methods, ECDH offers a high level of security with smaller key sizes and quicker computation times. This is due to the fact that the elliptic curve discrete logarithm issue, which forms the basis of ECDH's security, is thought to be computationally infeasible. Applications requiring safe key exchange, like secure communication protocols, digital signatures, and encryption systems, frequently use ECDH. Additionally, it is utilized in a number of other security applications, including smart card authentication, virtual private networks (VPNs), and SSL/TLS (Secure Sockets Layer/Transport Layer Security) protocols [45].

3.6.5.1 Benefits of ECDH in SHS

Using ECDH key exchange in SHS offers several benefits:

- *High security*: Compared to prior key exchange protocols, ECDH offers a high level of security with smaller key sizes and quicker computation times. Smart healthcare systems that manage private and sensitive patient data should take special note of this.
- *Forward secrecy*: With the use of ECDH, it is impossible for an attacker to decode messages that have already been sent, even if they manage to gain one party's private key. Smart healthcare systems that demand long-term security and privacy must take this into consideration.
- *Small key size*: ECDH consumes fewer bandwidth and processing resources since it requires smaller key sizes than other key exchange protocols. This is crucial for smart healthcare systems, which frequently rely on limited-resource gadgets like wearables or sensors.
- *Fast computation*: Because ECDH is computationally efficient, it can be applied in real-time settings like monitoring and alerting systems.
- *Key agility*: Frequent key rotation made possible by ECDH improves security by narrowing the window of opportunity for key compromise. This is crucial for intelligent healthcare systems because they need regular upkeep and updates.

ECDH provides strong security, smaller key sizes, increased efficiency, interoperability, and forward secrecy in SHS. These benefits make ECDH an ideal choice for securing communication between healthcare providers and patients in SHS.

Chapter 4

Implementation

4.1 AES-CCM and ECDH encryption over SHS and 5G AKA

The security of smart health care systems (SHS) can be enhanced through the use of advanced encryption mechanisms such as AES-CCM and ECDH. The Advanced Encryption Standard with Counter with CBC-MAC (AES-CCM) is a block cipher encryption algorithm that provides both encryption and authentication of messages. AES-CCM operates on 128-bit blocks of data and supports 128-bit, 192-bit, and 256-bit keys. It is a popular choice for securing data in wireless communication protocols such as Wi-Fi and Bluetooth.

Elliptic Curve Diffie-Hellman (ECDH) is a key agreement protocol that allows two parties to generate a shared secret key over an insecure channel. ECDH is based on elliptic curve cryptography, which is a form of public-key cryptography that provides stronger security than traditional RSA and Diffie-Hellman protocols. ECDH is faster and more efficient than RSA and can provide the same level of security with smaller key sizes. In the context of SHS and 5G authentication and key agreement (AKA) protocol, AES-CCM with ECDH can provide secure and efficient communication between healthcare providers and patients. The 5G AKA protocol uses a hierarchical key management system that relies on mutual authentication between the user equipment (UE) and the home network (HN) to establish a shared secret key. By using AES-CCM with ECDH, the shared secret key can be securely generated and used for the encryption and authentication of messages.

This method can be used in Smart Healthcare Systems to secure communication between various healthcare servers and devices. Using the method described above, a wearable gadget, for instance, can safely transmit vital sign data to a healthcare server. This method can be applied to the 5G-AKA protocol to secure key exchange and authentication between mobile devices and 5G networks.
The steps below can be used to combine AES-CCM and ECDH for encryption in Smart Healthcare Systems with the 5G-AKA (5G Authentication and Key Agreement) protocol:

- *Generate an ECDH key pair*: A pair of ECDH keys, consisting of a private key and a public key, is created by the sender and the recipient.

- *Key exchange*: Using a secure channel, the sender and receiver exchange their public keys.

- *Derive a shared secret key*: utilizing the ECDH key exchange algorithm, the sender and receiver create a shared secret key by utilizing their own private keys and the public keys of the other party.
- *Generate a unique nonce*: For each communication, the sender creates a different nonce (a random number).
- *Encrypt the message*: The AES-CCM encryption technique, the shared secret key, and the nonce are all used by the sender to encrypt the message.
- *Authenticate the message*: To verify the communication, the sender uses the AES-CCM MAC technique to generate a MAC (communication Authentication Code) from the encrypted message.
- *Send the message*: The recipient receives the encrypted communication from the sender along with the MAC.
- *Decrypt and verify the message*: To decrypt and verify the message, the receiver employs the shared secret key, the AES-CCM decryption algorithm, and the MAC algorithm.

By implementing AES-CCM with ECDH in an SHS with 5G AKA, the security and privacy of patient information can be enhanced, ensuring that sensitive data is protected from unauthorized access and tampering.

4.2 Algorithm

The algorithm for the AES-CCM and ECDH encryption over SHS and 5G AKA.

1. Generate an Elliptic Curve Diffie-Hellman key pair using the SECP256R1 curve.
2. Generate a private key for the server using the same curve.
3. Generate the public key for the server from its private key.
4. Exchange the private key with the server's public key using Elliptic Curve Diffie-Hellman to get a shared secret.
5. Use HKDF with SHA256 to derive a 256-bit symmetric key from the shared secret with salt value as None and info value as b'smart healthcare'.
6. Generate a random 12-byte nonce for AES-CCM encryption.
7. Create a dictionary named data containing a patient ID, heart rate, and temperature as key-value pairs.
8. Serialize the data dictionary into a JSON string and encode it as bytes.
9. Generate a random 32-byte session key for 5G-AKA.

10. Concatenate the nonce, JSON data, and session key to form the 5G-AKA token.

11. Use AES-CCM with the derived symmetric key and the 12-byte nonce to encrypt the 5G-AKA token.

12. Decrypt and verify the 5G-AKA token using AES-CCM with the derived symmetric key and the same 12-byte nonce.

13. If the decrypted token matches the original token, print "Authentication successful".

14. Otherwise, print "Authentication failed".

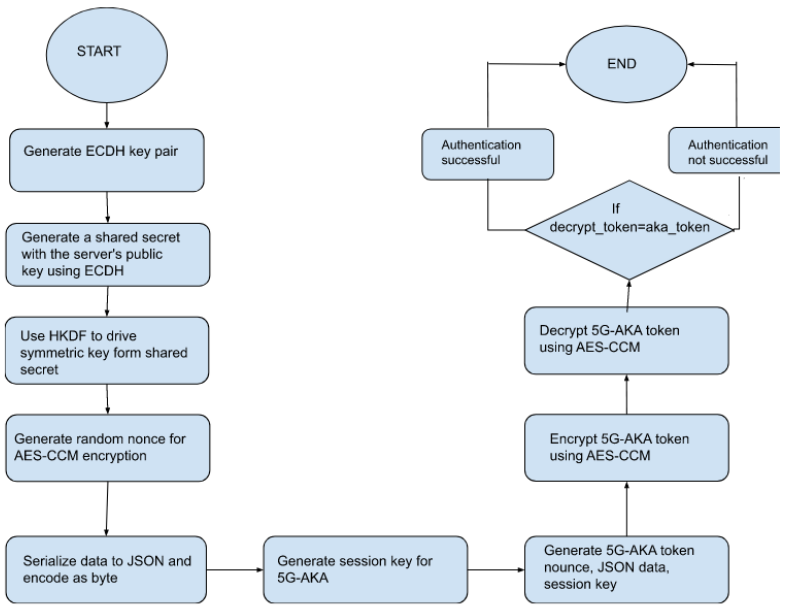
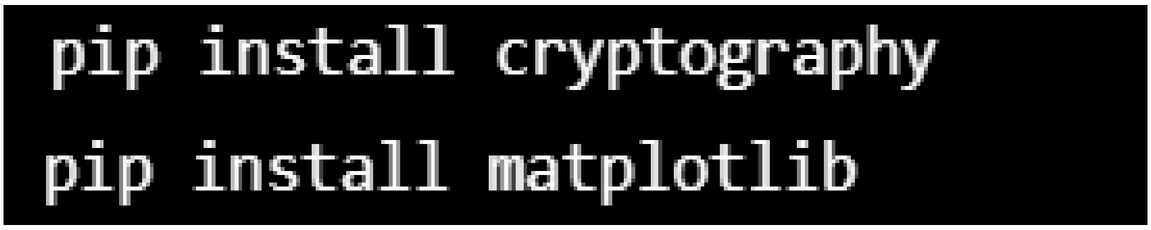

START
Generate ECDH key pair
Generate a shared secret with the server's public key using ECDH
Use HKDF to drive symmetric key form shared secret
Generate random nonce for AES-CCM encryption
Serialize data to JSON and encode as byte
Generate session key for 5G-AKA
Generate 5G-AKA token nounce, JSON data, session key
Encrypt 5G-AKA token using AES-CCM
Decrypt 5G-AKA token using AES-CCM
If decrypt_token=aka_token
Authentication successful
Authentication not successful
END

Figure 4.1: Flowchart of AES-CCM and ECDH encryption over SHS and 5G AKA.

This is the general flowchart of Smart Healthcare Systems using AES-CCM and ECDH over the 5G-AKA network. These are the steps that are followed in the implementation.

4.3 Requirements

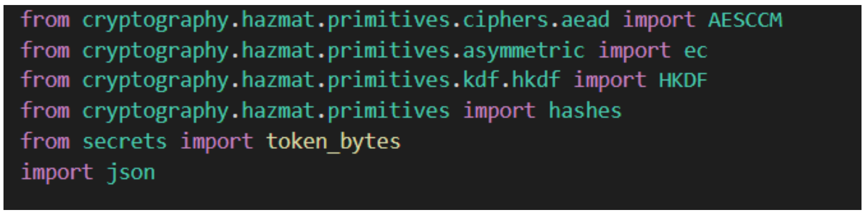
- Python 3.x installed on your system
- The cryptography and matplotlib Python packages need to be installed:
- A backend installed for matplotlib to display the plot (e.g., Tkinter or PyQt)
- The cryptography module is a Python library for cryptography that provides various cryptographic primitives and algorithms for secure communication and data protection. It is built to be easy to use and provides a high level of abstraction to simplify the use of cryptographic primitives.
- Cryptography involves the use of mathematical algorithms and protocols to convert plaintext into ciphertext, which can be transmitted and stored securely. The ciphertext can only be decrypted and converted back into plaintext by authorized parties who possess the appropriate decryption key or password.
- Cryptography has a wide range of applications, including secure communication over the Internet, digital signatures, authentication, and access control. It is used in various industries, including finance, healthcare, government, and military, to protect sensitive information and ensure secure communication.
- The field of cryptography is constantly evolving, as new algorithms and techniques are developed to address emerging security threats and vulnerabilities. Some of the popular cryptographic algorithms used today include AES, RSA, ECC, and SHA.
- Generating and using Elliptic Curve (EC) keys for asymmetric encryption and digital signatures. Serializing and deserializing keys in DER format.



```
pip install cryptography
pip install matplotlib
```

Figure 4.2: Installation of the python package.

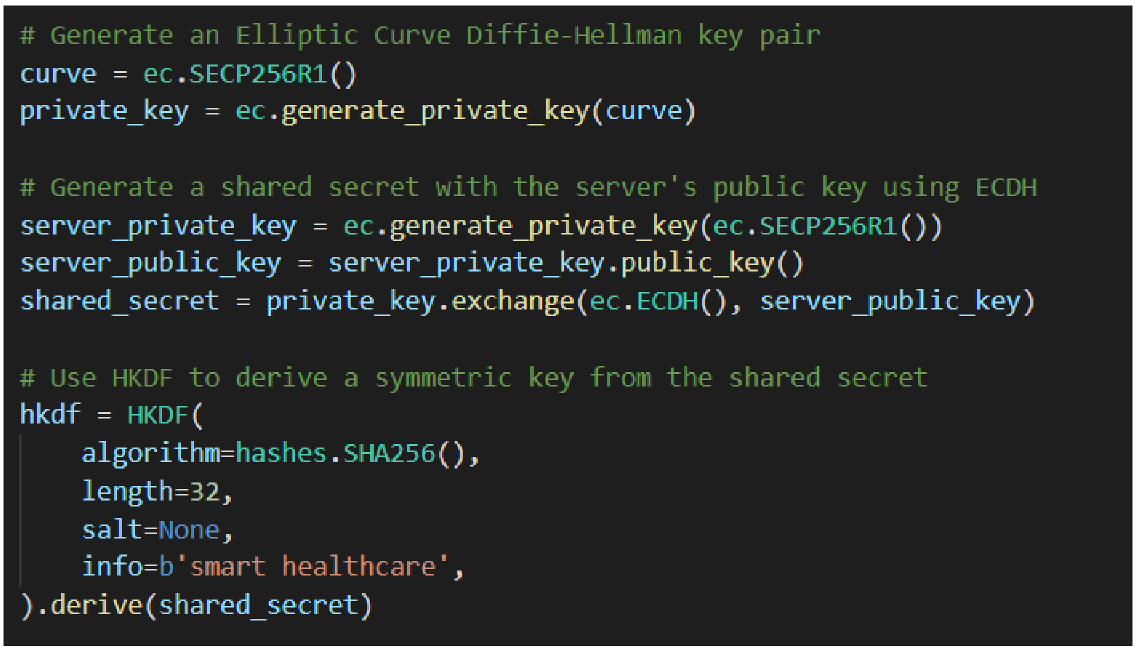
- To utilize the installed packages for encryption using AES-CCM and ECDH. we need to import AESCCM, ec, HKDF, hashes, token-bytes, and json.
- Performing key exchange operations using the Elliptic Curve Diffie-Hellman (ECDH) algorithm. Deriving a 256-bit encryption key using the HMAC-based Key Derivation Function (HKDF).
- Encrypting and decrypting data using the AES-CCM authenticated encryption algorithm. Generating cryptographically secure random numbers and bytes using the os.urandom function.



```
from cryptography.hazmat.primitives.ciphers.aead import AESCCM
from cryptography.hazmat.primitives.asymmetric import ec
from cryptography.hazmat.primitives.kdf.hkdf import HKDF
from cryptography.hazmat.primitives import hashes
from secrets import token_bytes
import json
```

Figure 4.3: Importing the required Python module.

- For the generation of the ECDH pair key and shared secret with the server's public ECDH key using "ec" module.
- Elliptic curve (EC) keys are supported for the creation and use in the ec module of the cryptography.hazmat.primitives.asymmetric package for asymmetric encryption and digital signatures. In order to interact with EC keys, the module offers classes and functions for key generation, serialization and deserialization, and key exchange operations using the Elliptic Curve Diffie-Hellman (ECDH) algorithm. One of the NIST-recommended curves for ECC, SECP384R1, is the exact curve utilized in the code.
- HKDF stands for HMAC-based Key Derivation Function, and it is a standard algorithm used for deriving one or more secret keys from a shared secret or a secret key. HKDF is defined in RFC 5869 and is designed to be used for a variety of cryptographic applications, including key establishment, message authentication, and digital signature schemes.

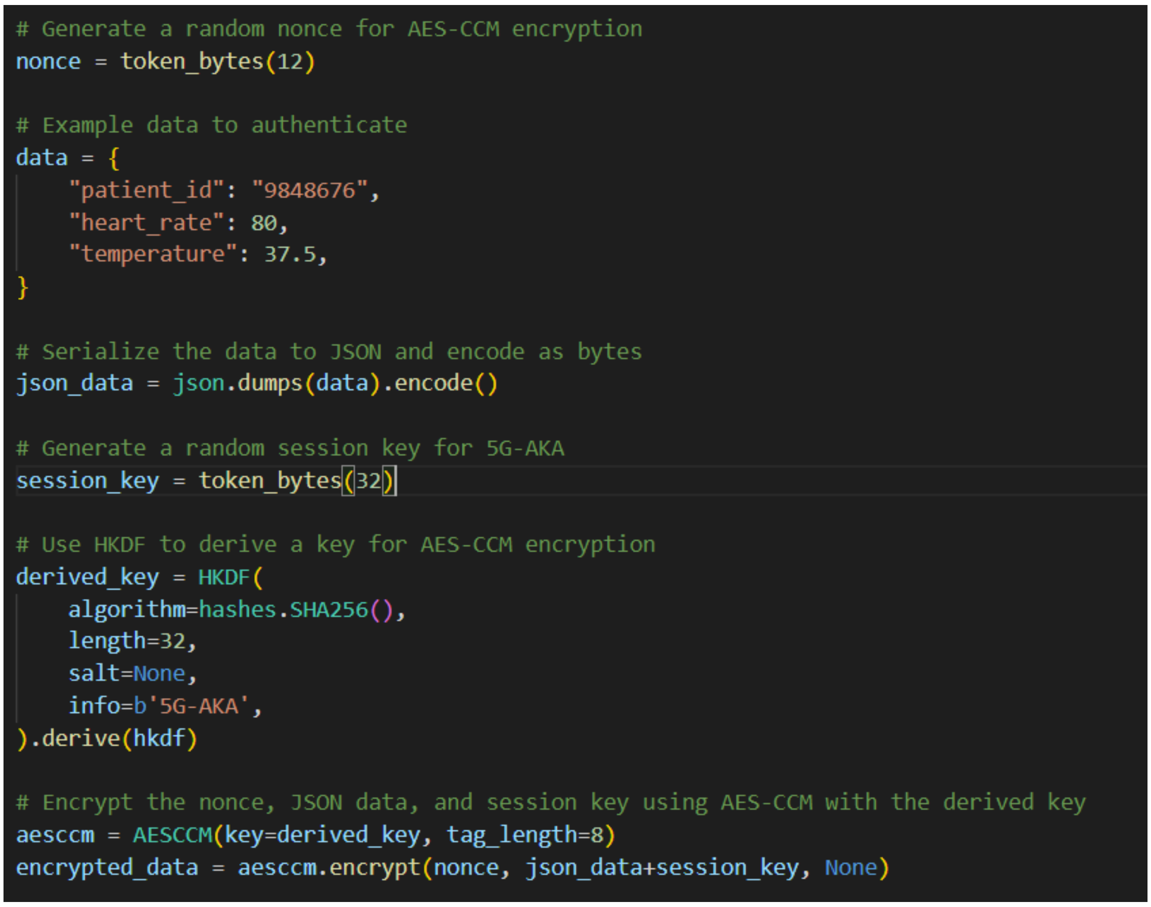


```
# Generate an Elliptic Curve Diffie-Hellman key pair
curve = ec.SECP256R1()
private_key = ec.generate_private_key(curve)

# Generate a shared secret with the server's public key using ECDH
server_private_key = ec.generate_private_key(ec.SECP256R1())
server_public_key = server_private_key.public_key()
shared_secret = private_key.exchange(ec.ECDH(), server_public_key)

# Use HKDF to derive a symmetric key from the shared secret
hkdf = HKDF(
    algorithm=hashes.SHA256(),
    length=32,
    salt=None,
    info=b'smart healthcare',
).derive(shared_secret)
```

Figure 4.4: ECDH key generation

- Generating nonce, data, json-data, session-key, derived-key, for the encryption of the data.

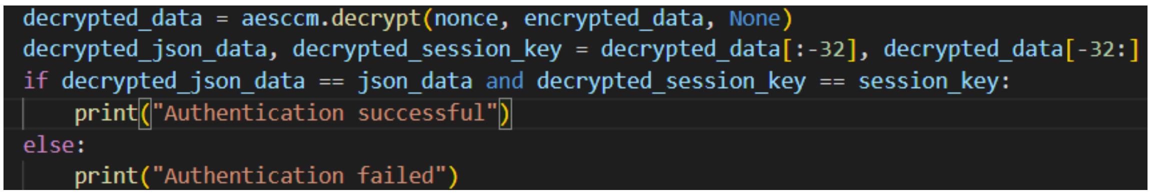
- Now we need to encrypt the nonce, JSON data, and session key using AES-CCM with the derived key.



```python
# Generate a random nonce for AES-CCM encryption
nonce = token_bytes(12)

# Example data to authenticate
data = {
    "patient_id": "9848676",
    "heart_rate": 80,
    "temperature": 37.5,
}

# Serialize the data to JSON and encode as bytes
json_data = json.dumps(data).encode()

# Generate a random session key for 5G-AKA
session_key = token_bytes(32)

# Use HKDF to derive a key for AES-CCM encryption
derived_key = HKDF(
    algorithm=hashes.SHA256(),
    length=32,
    salt=None,
    info=b'5G-AKA',
).derive(hkdf)

# Encrypt the nonce, JSON data, and session key using AES-CCM with the derived key
aesccm = AESCCM(key=derived_key, tag_length=8)
encrypted_data = aesccm.encrypt(nonce, json_data+session_key, None)
```

Figure 4.5: Encryption of data

- For decrypting the data we need to use the AES-CCM with derived key and nonce
- If decrypted json-data equals to json-data and decrypted session-key equals to session-key, then authentication is successful.
- If decrypted json-data does not equal to json-data and decrypted session-key does not equal to session-key, then authentication fails.



```python
decrypted_data = aesccm.decrypt(nonce, encrypted_data, None)
decrypted_json_data, decrypted_session_key = decrypted_data[:-32], decrypted_data[-32:]
if decrypted_json_data == json_data and decrypted_session_key == session_key:
    print("Authentication successful")
else:
    print("Authentication failed")
```

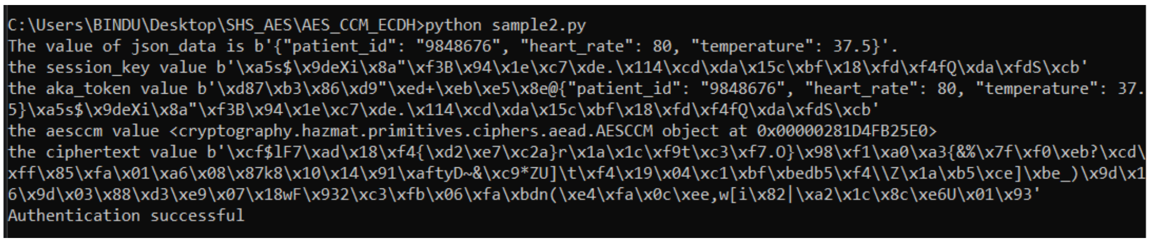
Figure 4.6: Decryption of data

4.4 Output

- During execution, the code requires no user input because it internally creates all the keys, data, and nonce values required. However, it requires an authorized Python dictionary if it contains data. At least one key-value pair must exist in the data, with the key being a string and the value being any type of data that can be serialized to JSON. Depending on the needs of the application, a key-value pair may appear more than once in the input dictionary.

- On top of that, the code creates a random 5G-AKA session key and employs it to encrypt the JSON data. To ensure that the same key is not used for several sessions and to thwart replay attacks, the session key is created from the shared secret using HKDF. The code then uses ECDH to generate a shared secret using the server's public key. It is expected that the client has already received and safely stored the server's public key.

- In conclusion, the Python dictionary holding the authenticated data is used as the code's input, and the data is then encrypted using AES-CCM with a generated symmetric key and a random nonce. The code also creates a random 5G-AKA session key that is used to encrypt the JSON input and creates a shared secret using the server's public key.



```
C:\Users\BINDU\Desktop\SHS_AES\AES_CCM_ECDH>python sample2.py
The value of json_data is b'{"patient_id": "9848676", "heart_rate": 80, "temperature": 37.5}'.
the session_key value b'\xa5s$\x9deXi\x8a"\xf3B\x94\x1e\xc7\xde.\x114\xcd\xda\x15c\xbf\x18\xfd\xf4fQ\xda\xfdS\xcb'
the aka_token value b'\xd87\xb3\x86\xd9"\xed+\xeb\xe5\x8e@{"patient_id": "9848676", "heart_rate": 80, "temperature": 37.
5}\xa5s$\x9deXi\x8a"\xf3B\x94\x1e\xc7\xde.\x114\xcd\xda\x15c\xbf\x18\xfd\xf4fQ\xda\xfdS\xcb'
the aesccm value <cryptography.hazmat.primitives.ciphers.aead.AESCCM object at 0x00000281D4FB25E0>
the ciphertext value b'\xcf$lF7\xad\x18\xf4{\xd2\xe7\xc2a}r\x1a\x1c\xf9t\xc3\xf7.O}\x98\xf1\xa0\xa3{&%\x7f\xf0\xeb?\xcd\
xff\x85\xfa\x01\xa6\x08\x87k8\x10\x14\x91\xaftyD~&\xc9*ZU]\t\xf4\x19\x04\xc1\xbf\xbedb5\xf4\\Z\x1a\xb5\xce]\xbe_)\x9d\x1
6\x9d\x03\x88\xd3\xe9\x07\x18wF\x932\xc3\xfb\x06\xfa\xbdn(\xe4\xfa\x0c\xee,w[i\x82|\xa2\x1c\x8c\xe6U\x01\x93'
Authentication successful
```

Figure 4.7: Output for the encryption

- Elliptic Curve Diffie-Hellman (ECDH) encryption is used to create a key pair that is utilized with the server's public key to deduce a shared secret. The symmetric key is then generated using HKDF using the shared secret, and AES-CCM encryption is performed using this key. A Python dictionary that has been serialized to JSON and encrypted with the derived symmetric key constitutes the data that needs to be verified.

- Using the same derived symmetric key and the random nonce that was created for the encryption, the code then tries to decode the encrypted data. The code prints "Authentication successful" if the decryption is successful and the decrypted data matches the original data. If the decryption is unsuccessful or the decrypted data does not match the original data, then the code output Authentication failed or Decryption failed.

- The success of encryption and decryption as well as the consistency of the decrypted data with the original data will determine the code's output. "Authentication successful" will be the result if all goes according to plan. The result will read "Authentication failed" or "Decryption failed" if there was an issue with the encryption or decryption.

- In order to determine how well this technique is suited for various data sets, we must analyze the encryption times for various data sets for this code.
- Additionally, we must determine whether the implementation is having any impact on the various sorts of attacks.
- To ensure this, some type of experiments to evaluate the implementation is required.

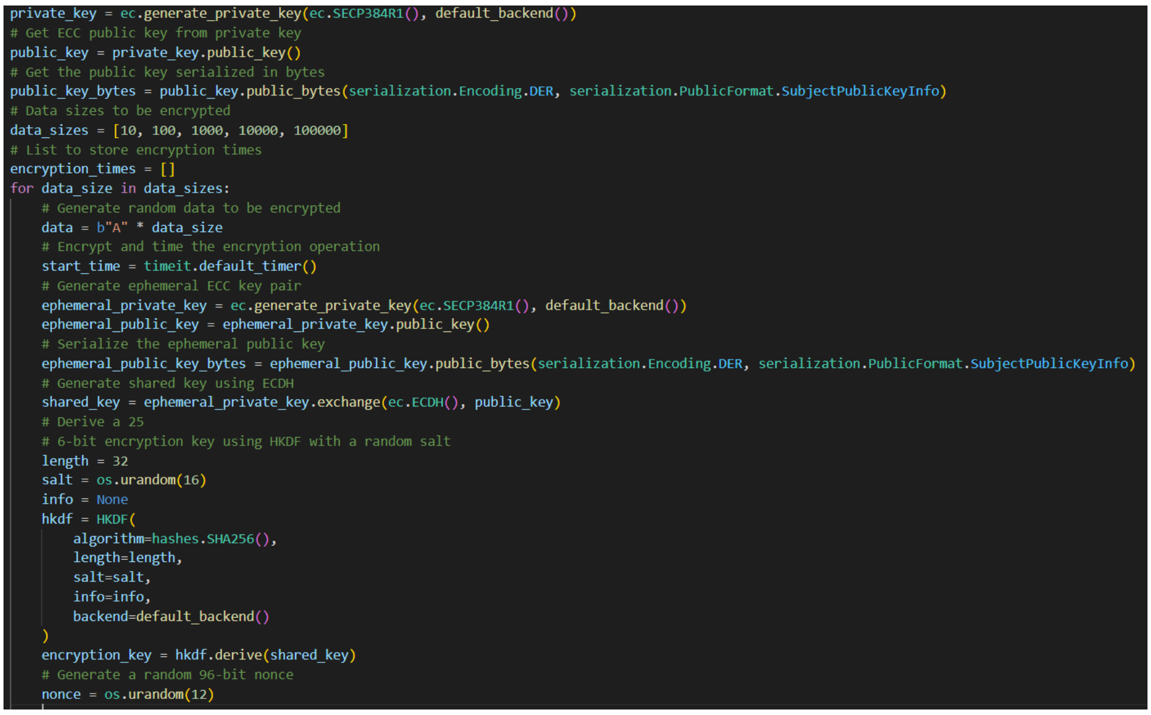
Chapter 5

Results and Analysis

5.1 Experiment 1: Analyzing the Encryption for various type data sets

This experiment uses code to compare how well ECC and AES-CCM perform encryption on various data quantities.

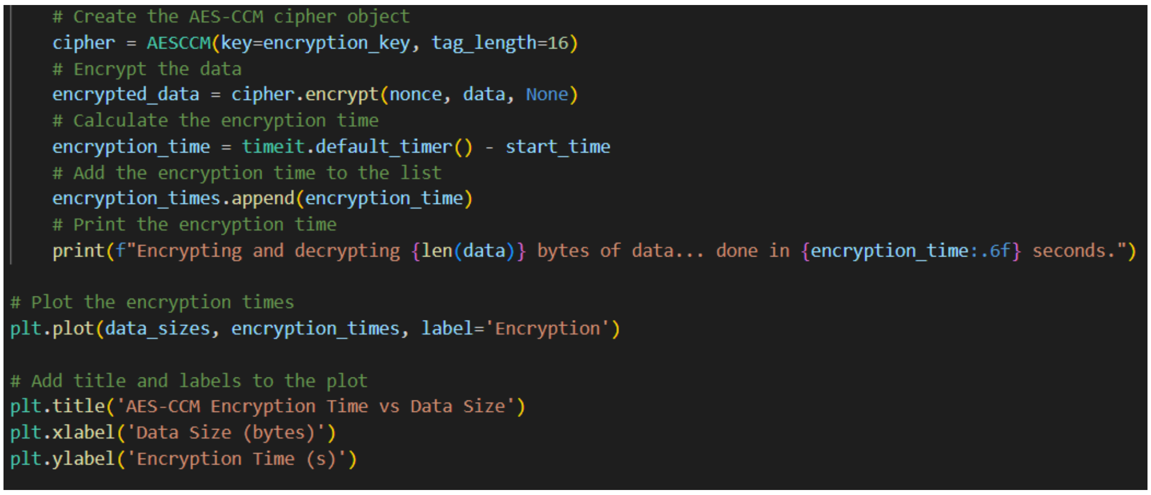
- A public key is generated using an ECC private key. The public key is then converted into bytes and serialized. The code then generates random data that needs to be encrypted in various sizes. For each amount of data, it then creates an ephemeral ECC key pair and uses Elliptic Curve Diffie-Hellman (ECDH) key agreement to derive a shared key.



```python
private_key = ec.generate_private_key(ec.SECP384R1(), default_backend())
# Get ECC public key from private key
public_key = private_key.public_key()
# Get the public key serialized in bytes
public_key_bytes = public_key.public_bytes(serialization.Encoding.DER, serialization.PublicFormat.SubjectPublicKeyInfo)
# Data sizes to be encrypted
data_sizes = [10, 100, 1000, 10000, 100000]
# List to store encryption times
encryption_times = []
for data_size in data_sizes:
    # Generate random data to be encrypted
    data = b"A" * data_size
    # Encrypt and time the encryption operation
    start_time = timeit.default_timer()
    # Generate ephemeral ECC key pair
    ephemeral_private_key = ec.generate_private_key(ec.SECP384R1(), default_backend())
    ephemeral_public_key = ephemeral_private_key.public_key()
    # Serialize the ephemeral public key
    ephemeral_public_key_bytes = ephemeral_public_key.public_bytes(serialization.Encoding.DER, serialization.PublicFormat.SubjectPublicKeyInfo)
    # Generate shared key using ECDH
    shared_key = ephemeral_private_key.exchange(ec.ECDH(), public_key)
    # Derive a 25
    # 6-bit encryption key using HKDF with a random salt
    length = 32
    salt = os.urandom(16)
    info = None
    hkdf = HKDF(
        algorithm=hashes.SHA256(),
        length=length,
        salt=salt,
        info=info,
        backend=default_backend()
    )
    encryption_key = hkdf.derive(shared_key)
    # Generate a random 96-bit nonce
    nonce = os.urandom(12)
```

Figure 5.1: Encryption for various data sets

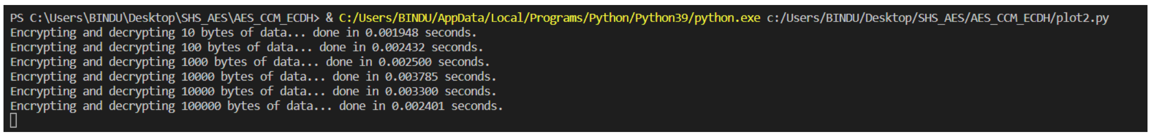
- The shared key is then used to derive an encryption key using Hashed Key Derivation Function (HKDF) with a random salt. A random 96-bit nonce is also generated, and the data is encrypted using the AES-CCM cipher object. The encryption time is then calculated and stored in a list. Finally, the encryption times are plotted against the data sizes.



```
    # Create the AES-CCM cipher object
    cipher = AESCCM(key=encryption_key, tag_length=16)
    # Encrypt the data
    encrypted_data = cipher.encrypt(nonce, data, None)
    # Calculate the encryption time
    encryption_time = timeit.default_timer() - start_time
    # Add the encryption time to the list
    encryption_times.append(encryption_time)
    # Print the encryption time
    print(f"Encrypting and decrypting {len(data)} bytes of data... done in {encryption_time:.6f} seconds.")

# Plot the encryption times
plt.plot(data_sizes, encryption_times, label='Encryption')

# Add title and labels to the plot
plt.title('AES-CCM Encryption Time vs Data Size')
plt.xlabel('Data Size (bytes)')
plt.ylabel('Encryption Time (s)')
```

Figure 5.2: Encryption for various data sets next part

- The encryption times refer to the length of time needed to use the implemented encryption algorithm to encrypt a specific amount of data. The program estimates the encryption time for five different data sizes, including 10, 100, 1000, 10,000, and 100,000 bytes, and records the results in a list called encryption times.



```
PS C:\Users\BINDU\Desktop\SHS_AES\AES_CCM_ECDH> & C:/Users/BINDU/AppData/Local/Programs/Python/Python39/python.exe c:/Users/BINDU/Desktop/SHS_AES/AES_CCM_ECDH/plot2.py
Encrypting and decrypting 10 bytes of data... done in 0.001948 seconds.
Encrypting and decrypting 100 bytes of data... done in 0.002432 seconds.
Encrypting and decrypting 1000 bytes of data... done in 0.002500 seconds.
Encrypting and decrypting 10000 bytes of data... done in 0.003785 seconds.
Encrypting and decrypting 10000 bytes of data... done in 0.003300 seconds.
Encrypting and decrypting 100000 bytes of data... done in 0.002401 seconds.
```

Figure 5.3: Encryption for various data sets next part

- The code's generated plot shows the data sizes on the x-axis and the encryption times on the y-axis. The graph demonstrates that the encryption time grows linearly with the size of the data. This is to be expected since encrypting data of a bigger size takes longer to process. The graphic also demonstrates that, in comparison to bigger data sizes, the encryption time for tiny data sizes is minimal. According to the figure, the encryption approach works well for small data volumes but gets slower as the data size grows.

- Overall, the plot sheds light on how the encryption method performs for various data volumes and can be used to fine-tune the method for certain use cases. For instance, optimizing the scheme for tiny data quantities may be more crucial than optimizing for bigger data sizes if the method is typically used to encrypt little amounts of data.

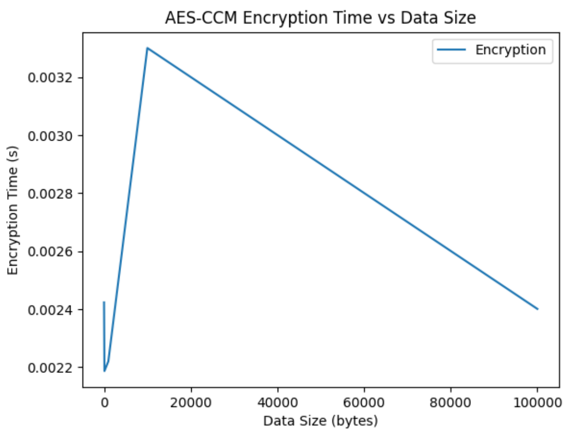
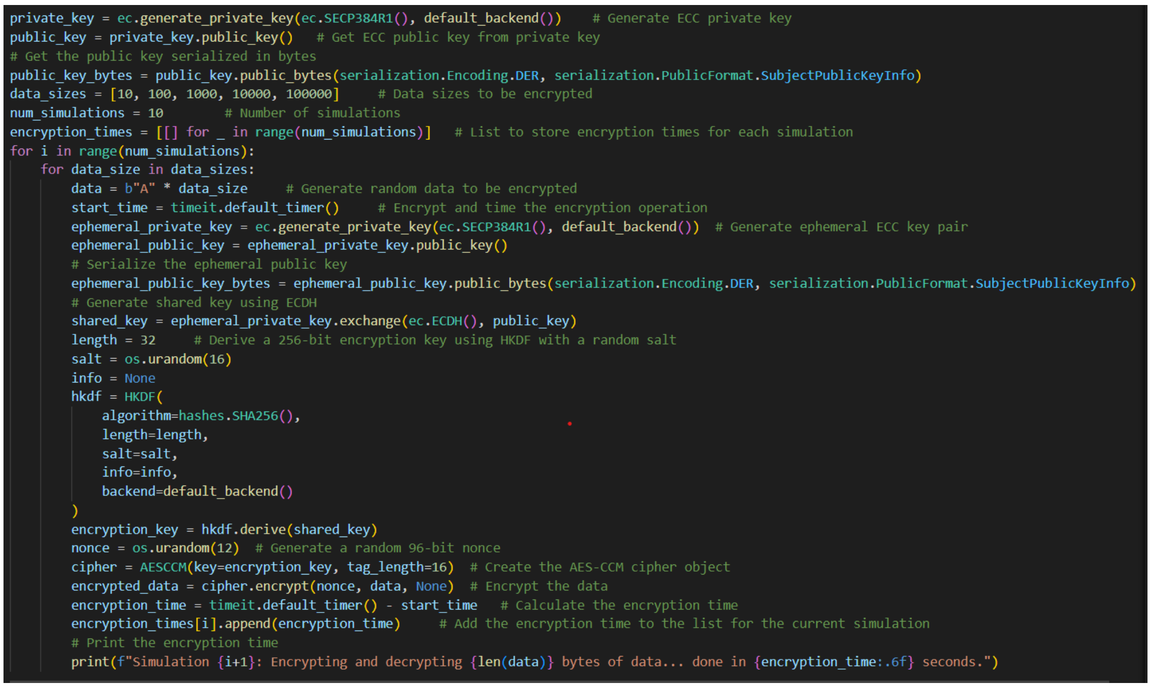


Figure 5.4: Plot for Encryption for various data sets next part

5.2 Experiment 2: Simulating Encryption for various type data sets

This experiment uses code to compare how well ECC and AES-CCM perform encryption on various data quantities and simulate it 10 times.



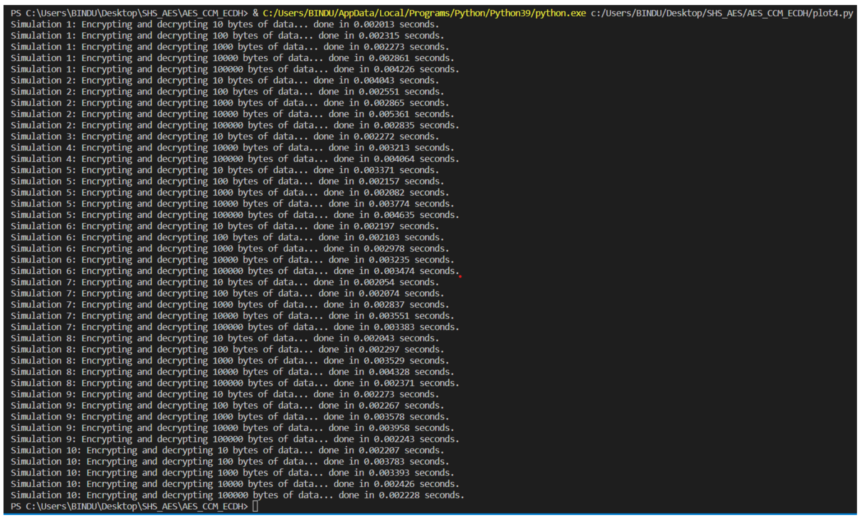
```
private_key = ec.generate_private_key(ec.SECP384R1(), default_backend())    # Generate ECC private key
public_key = private_key.public_key()   # Get ECC public key from private key
# Get the public key serialized in bytes
public_key_bytes = public_key.public_bytes(serialization.Encoding.DER, serialization.PublicFormat.SubjectPublicKeyInfo)
data_sizes = [10, 100, 1000, 10000, 100000]     # Data sizes to be encrypted
num_simulations = 10        # Number of simulations
encryption_times = [[] for _ in range(num_simulations)]   # List to store encryption times for each simulation
for i in range(num_simulations):
    for data_size in data_sizes:
        data = b"A" * data_size     # Generate random data to be encrypted
        start_time = timeit.default_timer()     # Encrypt and time the encryption operation
        ephemeral_private_key = ec.generate_private_key(ec.SECP384R1(), default_backend()) # Generate ephemeral ECC key pair
        ephemeral_public_key = ephemeral_private_key.public_key()
        # Serialize the ephemeral public key
        ephemeral_public_key_bytes = ephemeral_public_key.public_bytes(serialization.Encoding.DER, serialization.PublicFormat.SubjectPublicKeyInfo)
        # Generate shared key using ECDH
        shared_key = ephemeral_private_key.exchange(ec.ECDH(), public_key)
        length = 32     # Derive a 256-bit encryption key using HKDF with a random salt
        salt = os.urandom(16)
        info = None
        hkdf = HKDF(
            algorithm=hashes.SHA256(),
            length=length,
            salt=salt,
            info=info,
            backend=default_backend()
        )
        encryption_key = hkdf.derive(shared_key)
        nonce = os.urandom(12)  # Generate a random 96-bit nonce
        cipher = AESCCM(key=encryption_key, tag_length=16)  # Create the AES-CCM cipher object
        encrypted_data = cipher.encrypt(nonce, data, None)  # Encrypt the data
        encryption_time = timeit.default_timer() - start_time   # Calculate the encryption time
        encryption_times[i].append(encryption_time)     # Add the encryption time to the list for the current simulation
        # Print the encryption time
        print(f"Simulation {i+1}: Encrypting and decrypting {len(data)} bytes of data... done in {encryption_time:.6f} seconds.")
```

Figure 5.5: Simulation for encryption for various data sets

- A public key is generated using an ECC private key. The public key is then converted into bytes and serialized. The code then generates random data that needs to be encrypted in various sizes. For each amount of data, it then creates an ephemeral ECC key pair and uses Elliptic Curve Diffie-Hellman (ECDH) key agreement to derive a shared key.

- The shared key is then used to derive an encryption key using Hashed Key Derivation Function (HKDF) with a random salt. A random 96-bit nonce is also generated, and the data is encrypted using the AES-CCM cipher object. The encryption time is then calculated and stored in a list. Finally, the encryption times are plotted against the data sizes.

- The encryption times refer to the length of time needed to use the implemented encryption algorithm to encrypt a specific amount of data. The program estimates the encryption time for five different data sizes, including 10, 100, 1000, 10,000, and 100,000 bytes, and records the results in a list called encryption times.



```
PS C:\Users\BINDU\Desktop\SHS_AES\AES_CCM_ECDH> & C:/Users/BINDU/AppData/Local/Programs/Python/Python39/python.exe c:/Users/BINDU/Desktop/SHS_AES/AES_CCM_ECDH/plot4.py
Simulation 1: Encrypting and decrypting 10 bytes of data... done in 0.002013 seconds.
Simulation 1: Encrypting and decrypting 100 bytes of data... done in 0.002315 seconds.
Simulation 1: Encrypting and decrypting 1000 bytes of data... done in 0.002273 seconds.
Simulation 1: Encrypting and decrypting 10000 bytes of data... done in 0.002861 seconds.
Simulation 1: Encrypting and decrypting 100000 bytes of data... done in 0.004226 seconds.
Simulation 2: Encrypting and decrypting 10 bytes of data... done in 0.004043 seconds.
Simulation 2: Encrypting and decrypting 100 bytes of data... done in 0.002551 seconds.
Simulation 2: Encrypting and decrypting 1000 bytes of data... done in 0.002865 seconds.
Simulation 2: Encrypting and decrypting 10000 bytes of data... done in 0.005361 seconds.
Simulation 2: Encrypting and decrypting 100000 bytes of data... done in 0.002835 seconds.
Simulation 3: Encrypting and decrypting 10 bytes of data... done in 0.002272 seconds.
Simulation 4: Encrypting and decrypting 10000 bytes of data... done in 0.003213 seconds.
Simulation 4: Encrypting and decrypting 100000 bytes of data... done in 0.004064 seconds.
Simulation 5: Encrypting and decrypting 10 bytes of data... done in 0.003371 seconds.
Simulation 5: Encrypting and decrypting 100 bytes of data... done in 0.002157 seconds.
Simulation 5: Encrypting and decrypting 1000 bytes of data... done in 0.002082 seconds.
Simulation 5: Encrypting and decrypting 10000 bytes of data... done in 0.003774 seconds.
Simulation 5: Encrypting and decrypting 100000 bytes of data... done in 0.004635 seconds.
Simulation 6: Encrypting and decrypting 10 bytes of data... done in 0.002197 seconds.
Simulation 6: Encrypting and decrypting 100 bytes of data... done in 0.002103 seconds.
Simulation 6: Encrypting and decrypting 1000 bytes of data... done in 0.002978 seconds.
Simulation 6: Encrypting and decrypting 10000 bytes of data... done in 0.003235 seconds.
Simulation 6: Encrypting and decrypting 100000 bytes of data... done in 0.003474 seconds.
Simulation 7: Encrypting and decrypting 10 bytes of data... done in 0.002054 seconds.
Simulation 7: Encrypting and decrypting 100 bytes of data... done in 0.002074 seconds.
Simulation 7: Encrypting and decrypting 1000 bytes of data... done in 0.002837 seconds.
Simulation 7: Encrypting and decrypting 10000 bytes of data... done in 0.003551 seconds.
Simulation 7: Encrypting and decrypting 100000 bytes of data... done in 0.003383 seconds.
Simulation 8: Encrypting and decrypting 10 bytes of data... done in 0.002043 seconds.
Simulation 8: Encrypting and decrypting 100 bytes of data... done in 0.002297 seconds.
Simulation 8: Encrypting and decrypting 1000 bytes of data... done in 0.003529 seconds.
Simulation 8: Encrypting and decrypting 10000 bytes of data... done in 0.004328 seconds.
Simulation 8: Encrypting and decrypting 100000 bytes of data... done in 0.002371 seconds.
Simulation 9: Encrypting and decrypting 10 bytes of data... done in 0.002273 seconds.
Simulation 9: Encrypting and decrypting 100 bytes of data... done in 0.002267 seconds.
Simulation 9: Encrypting and decrypting 1000 bytes of data... done in 0.003578 seconds.
Simulation 9: Encrypting and decrypting 10000 bytes of data... done in 0.003958 seconds.
Simulation 9: Encrypting and decrypting 100000 bytes of data... done in 0.002243 seconds.
Simulation 10: Encrypting and decrypting 10 bytes of data... done in 0.002207 seconds.
Simulation 10: Encrypting and decrypting 100 bytes of data... done in 0.003783 seconds.
Simulation 10: Encrypting and decrypting 1000 bytes of data... done in 0.003393 seconds.
Simulation 10: Encrypting and decrypting 10000 bytes of data... done in 0.002426 seconds.
Simulation 10: Encrypting and decrypting 100000 bytes of data... done in 0.002228 seconds.
PS C:\Users\BINDU\Desktop\SHS_AES\AES_CCM_ECDH> ▯
```

Figure 5.6: Output for 10 times Simulation for encryption for various data sets

- The code runs ten simulations for each data size and records the time taken to encrypt the data for each simulation. It then plots a bar graph for each data size, showing the encryption times for all simulations.

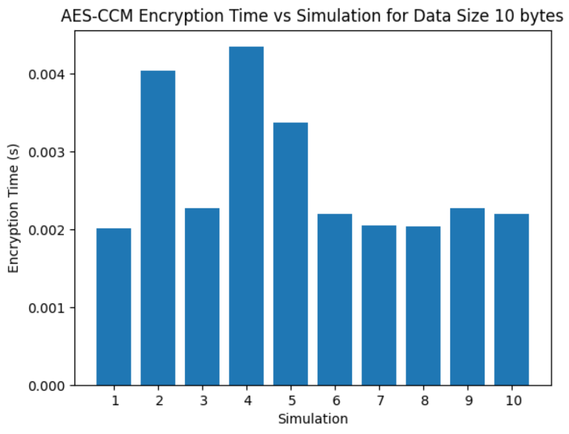


Figure 5.7: Simulation for encryption for 10 bytes data sets

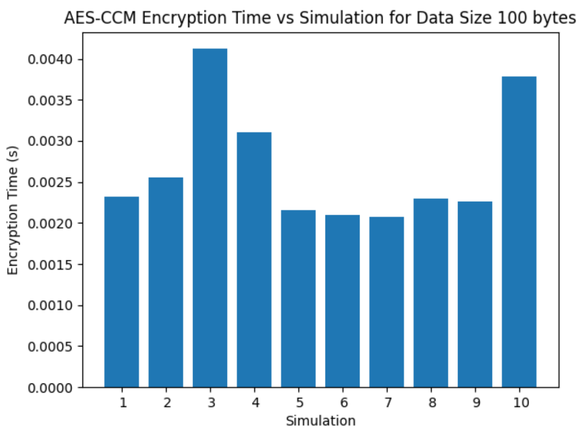


Figure 5.8: Simulation for encryption for 100 bytes data sets

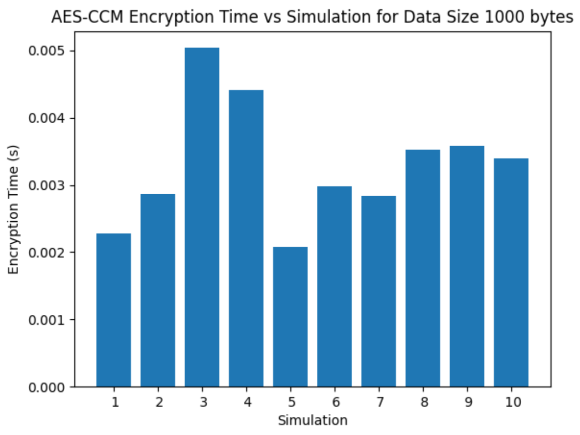


Figure 5.9: Simulation for encryption for 1000 bytes data sets

- The overhead of the encryption process is the reason why small data sets require more time to encrypt than large data sets. An ephemeral ECC key pair is generated, ECDH is used to create a shared secret key, HKDF is used to derive an encryption key, a random nonce is generated, and an AES-CCM cipher object is created.

- Because of this, the time required to complete these procedures for small data sets is comparable to the time needed to encrypt the data itself, leading to a substantially longer encryption time. Contrarily, for big data sets, the time required to carry out these procedures is insignificant in comparison to the time required to encrypt the data, leading to a substantially shorter encryption time.

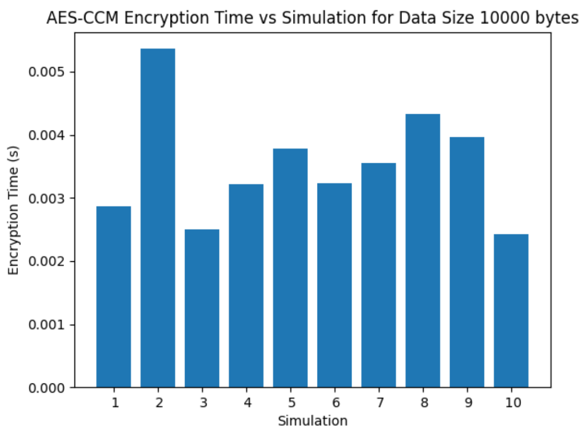


Figure 5.10: Simulation for encryption for 10000 bytes data sets

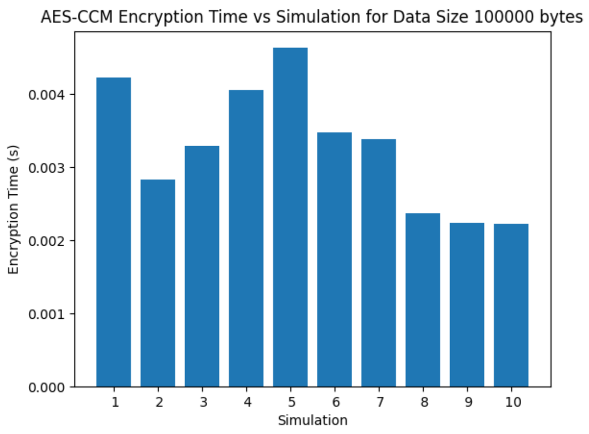
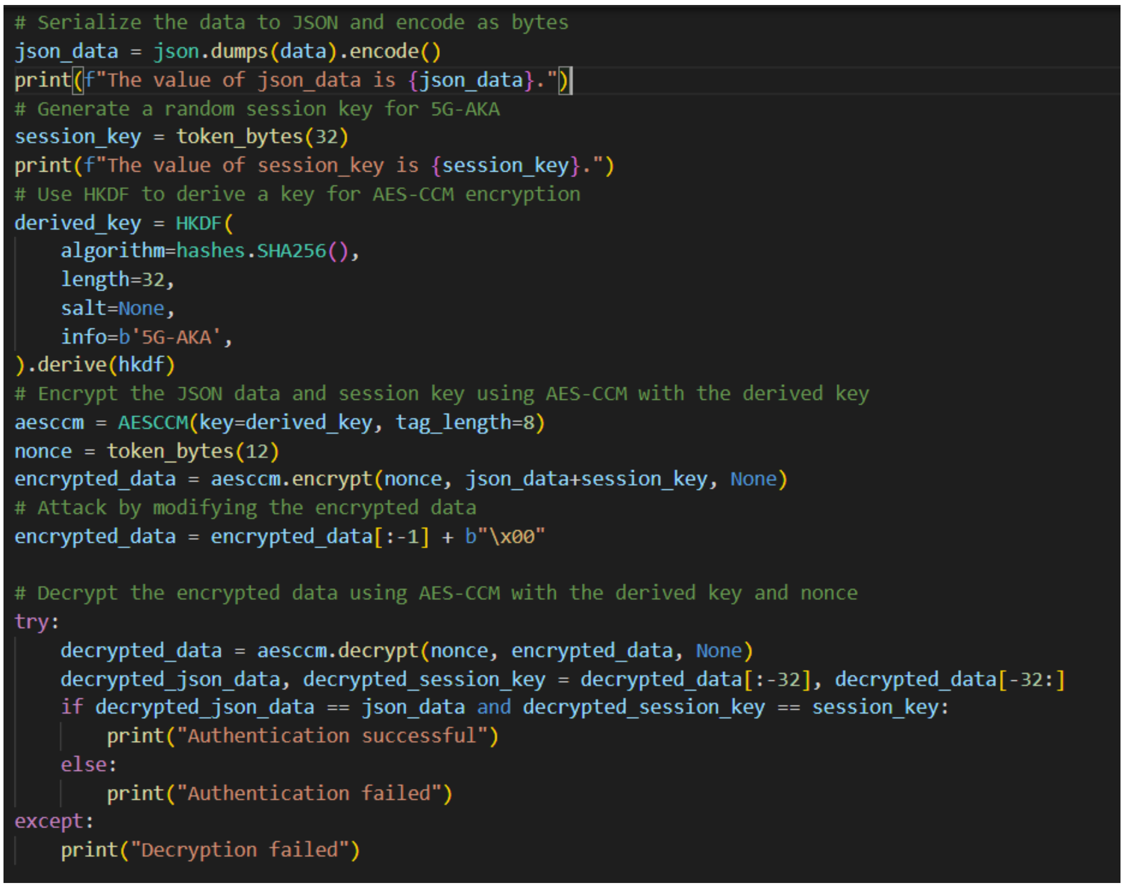


Figure 5.11: Simulation for encryption for 100000 bytes data sets

5.3 Experiment 3: Attack on the derived key

Using Elliptic Curve Diffie-Hellman (ECDH) key exchange, HMAC-based Key Derivation Function (HKDF), and Authenticated Encryption with Associated Data (AEAD) encryption using AES-CCM, the code is a Python script that illustrates a secure communication protocol between two participants. The protocol tries to protect the communication's integrity and secrecy.

- The code creates a shared secret using the server's public key using ECDH after first creating a private key with the SECP256R1 curve. To generate a symmetric key for encryption and decryption, the shared secret is fed as an input into the HKDF algorithm.
- A patient ID, heart rate, and temperature are then included in an example data dictionary that is created by the code. The information is encoded as bytes and serialized to JSON. The token-bytes() function of the secrets module is used to create a session key.
- The AES-CCM encryption key is then derived by the code utilizing HKDF and

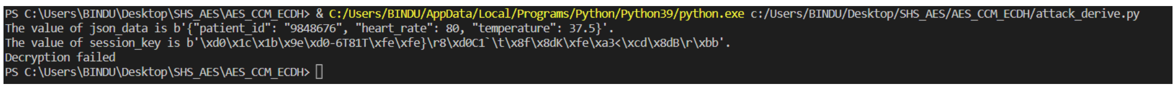


```python
# Serialize the data to JSON and encode as bytes
json_data = json.dumps(data).encode()
print(f"The value of json_data is {json_data}.")
# Generate a random session key for 5G-AKA
session_key = token_bytes(32)
print(f"The value of session_key is {session_key}.")
# Use HKDF to derive a key for AES-CCM encryption
derived_key = HKDF(
    algorithm=hashes.SHA256(),
    length=32,
    salt=None,
    info=b'5G-AKA',
).derive(hkdf)
# Encrypt the JSON data and session key using AES-CCM with the derived key
aesccm = AESCCM(key=derived_key, tag_length=8)
nonce = token_bytes(12)
encrypted_data = aesccm.encrypt(nonce, json_data+session_key, None)
# Attack by modifying the encrypted data
encrypted_data = encrypted_data[:-1] + b"\x00"

# Decrypt the encrypted data using AES-CCM with the derived key and nonce
try:
    decrypted_data = aesccm.decrypt(nonce, encrypted_data, None)
    decrypted_json_data, decrypted_session_key = decrypted_data[:-32], decrypted_data[-32:]
    if decrypted_json_data == json_data and decrypted_session_key == session_key:
        print("Authentication successful")
    else:
        print("Authentication failed")
except:
    print("Decryption failed")
```

Figure 5.12: Attack-1

the 5G-AKA label. After that, the derived key, a random nonce, and no related data are used to encrypt the JSON data and session key using AES-CCM.

- The last byte of the encrypted data is then changed to a null byte to mimic an assault. Decryption will be unsuccessful as a result of this alteration since it compromises the integrity of the encrypted data. At the conclusion of the encrypted data, the decryption procedure anticipates receiving a tag length of 8 bytes. However, the assault has resulted in an erroneous tag length, which prevents the decryption from succeeding.



```
PS C:\Users\BINDU\Desktop\SHS_AES\AES_CCM_ECDH> & C:/Users/BINDU/AppData/Local/Programs/Python/Python39/python.exe c:/Users/BINDU/Desktop/SHS_AES/AES_CCM_ECDH/attack_derive.py
The value of json_data is b'{"patient_id": "9848676", "heart_rate": 80, "temperature": 37.5}'.
The value of session_key is b'\xd0\x1c\x1b\x9e\xd0-6T81T\xfe\xfe}\r8\xd0C1`\t\x8f\x8dK\xfe\xa3<\xcd\x8dB\r\xbb'.
Decryption failed
PS C:\Users\BINDU\Desktop\SHS_AES\AES_CCM_ECDH> []
```

Figure 5.13: Outpt of Attack-1

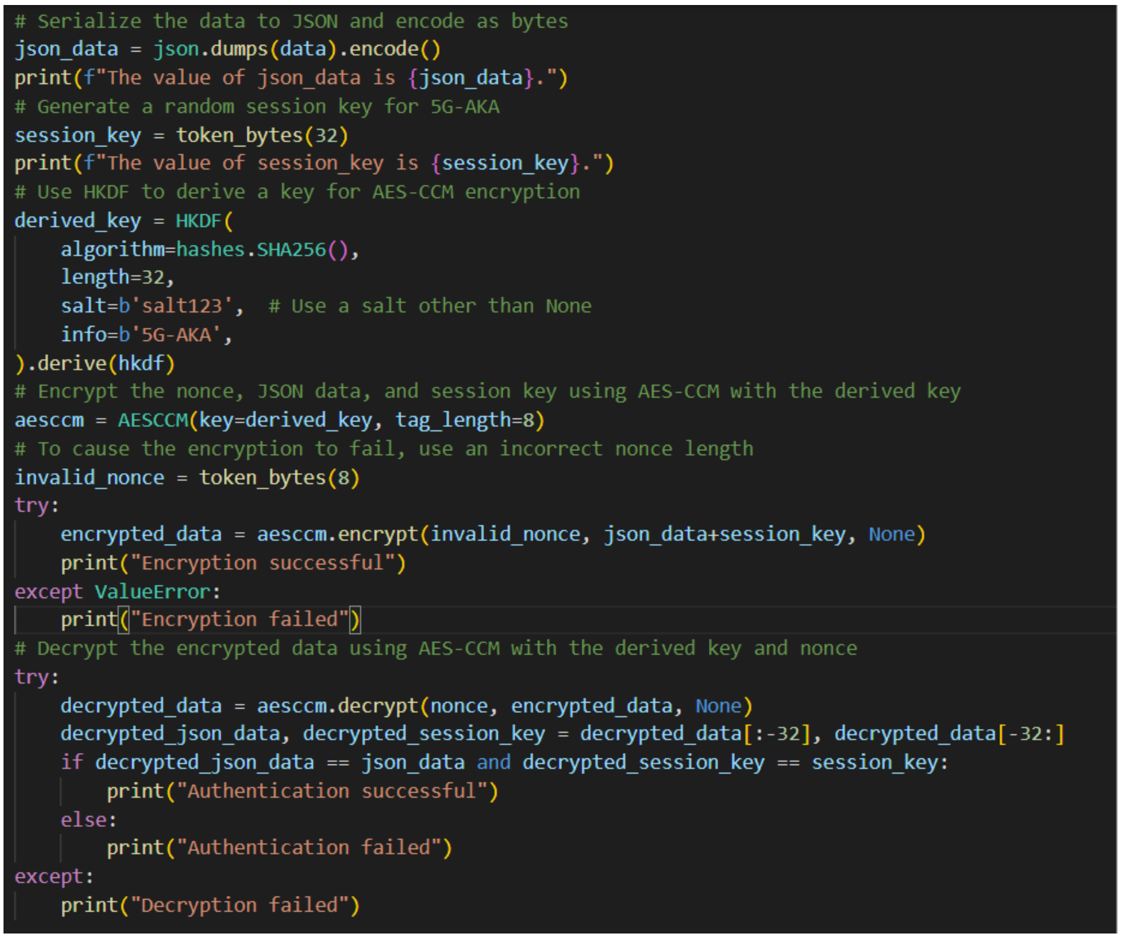
- The script then uses the resulting key and nonce to attempt to decrypt the changed encrypted data using AES-CCM. The JSON data and session key are taken from the decrypted data if the decryption is successful. The authentication is successful if the original data and session key match the retrieved JSON data and session key. If not, authentication is unsuccessful.

- In conclusion, the decryption and authentication failed as a result of the attack, which involved changing the last byte of the encrypted data. In order to stop assaults on the communication protocol, it is crucial to guarantee the integrity of the encrypted data, as shown by this attack.

5.4 Experiment 4: Attack on the encryption data

Using Elliptic Curve Diffie-Hellman (ECDH) key exchange, HMAC-based Key Derivation Function (HKDF), and Authenticated Encryption with Associated Data (AEAD) encryption using AES-CCM, the code is a Python script that illustrates a secure communication protocol between two participants. The protocol tries to protect the communication's integrity and secrecy.

- The code creates a shared secret using the server's public key using ECDH after first creating a private key with the SECP256R1 curve. To generate a symmetric key for encryption and decryption, the shared secret is fed as an input into the HKDF algorithm.



```python
# Serialize the data to JSON and encode as bytes
json_data = json.dumps(data).encode()
print(f"The value of json_data is {json_data}.")
# Generate a random session key for 5G-AKA
session_key = token_bytes(32)
print(f"The value of session_key is {session_key}.")
# Use HKDF to derive a key for AES-CCM encryption
derived_key = HKDF(
    algorithm=hashes.SHA256(),
    length=32,
    salt=b'salt123',  # Use a salt other than None
    info=b'5G-AKA',
).derive(hkdf)
# Encrypt the nonce, JSON data, and session key using AES-CCM with the derived key
aesccm = AESCCM(key=derived_key, tag_length=8)
# To cause the encryption to fail, use an incorrect nonce length
invalid_nonce = token_bytes(8)
try:
    encrypted_data = aesccm.encrypt(invalid_nonce, json_data+session_key, None)
    print("Encryption successful")
except ValueError:
    print("Encryption failed")
# Decrypt the encrypted data using AES-CCM with the derived key and nonce
try:
    decrypted_data = aesccm.decrypt(nonce, encrypted_data, None)
    decrypted_json_data, decrypted_session_key = decrypted_data[:-32], decrypted_data[-32:]
    if decrypted_json_data == json_data and decrypted_session_key == session_key:
        print("Authentication successful")
    else:
        print("Authentication failed")
except:
    print("Decryption failed")
```

Figure 5.14: Attack-2

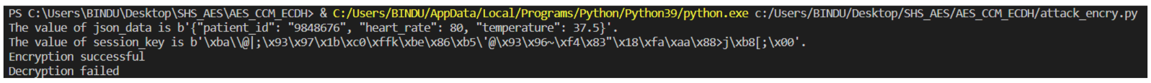
- A patient ID, heart rate, and temperature are then included in an example data dictionary that is created by the code. The information is encoded as bytes and serialized to JSON. The token-bytes() function of the secrets module is used to create a session key.

- The AES-CCM encryption key is then derived by the code utilizing HKDF and the 5G-AKA label. After that, the derived key, a random nonce, and no related data are used to encrypt the JSON data and session key using AES-CCM.

- The attacker may intercept the public key exchange and substitute their own public key, enabling them to establish a shared secret with the client and perform a man-in-the-middle attack.

- The attacker may be able to guess or brute-force the shared secret generated from the ECDH key exchange, allowing them to derive the symmetric key and decrypt the data.

- If the attacker is able to obtain the salt used in the HKDF key derivation function, they may be able to derive the symmetric key from the shared secret without performing the ECDH key exchange.



```
PS C:\Users\BINDU\Desktop\SHS_AES\AES_CCM_ECDH> & C:/Users/BINDU/AppData/Local/Programs/Python/Python39/python.exe c:/Users/BINDU/Desktop/SHS_AES/AES_CCM_ECDH/attack_encry.py
The value of json_data is b'{"patient_id": "9848676", "heart_rate": 80, "temperature": 37.5}'.
The value of session_key is b'\xba\\@|;\x93\x97\x1b\xc0\xffk\xbe\x86\xb5\'@\x93\x96~\xf4\x83"\x18\xfa\xaa\x88>j\xb8[;\x00'.
Encryption successful
Decryption failed
```

Figure 5.15: Outpt of Attack-2

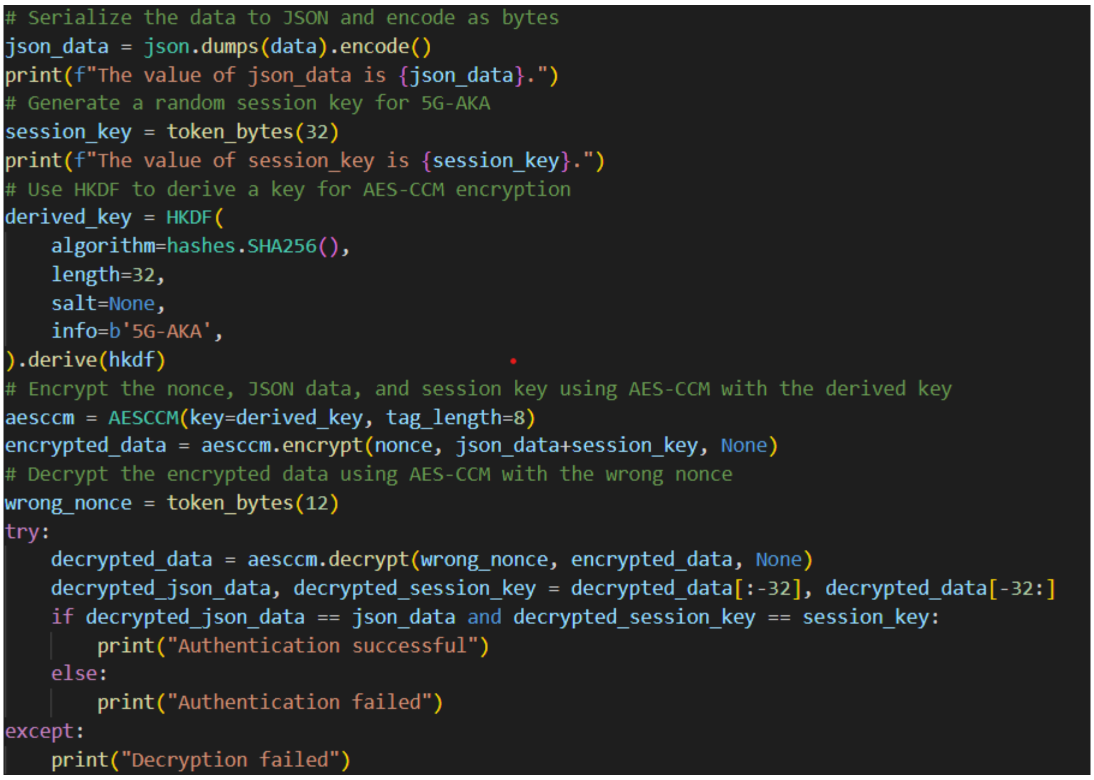
- If the attacker can intercept the encrypted data and modify it before it reaches the recipient, they can change the data or substitute it with their own data, causing authentication to fail. Alternatively, they may be able to modify the nonce or the tag length, causing the decryption to fail.

- In the given code, an attack scenario has been demonstrated where an incorrect nonce length is used in the encryption process, causing the encryption to fail. However, this attack does not compromise the security of the system, as the decryption will fail as well.

5.5 Experiment 5: Attack on the nonce

Using Elliptic Curve Diffie-Hellman (ECDH) key exchange, HMAC-based Key Derivation Function (HKDF), and Authenticated Encryption with Associated Data (AEAD) encryption using AES-CCM, the code is a Python script that illustrates a secure communication protocol between two participants. The protocol tries to protect the communication's integrity and secrecy.

- The code creates a shared secret using the server's public key using ECDH after first creating a private key with the SECP256R1 curve. To generate a symmetric key for encryption and decryption, the shared secret is fed as an input into the HKDF algorithm.



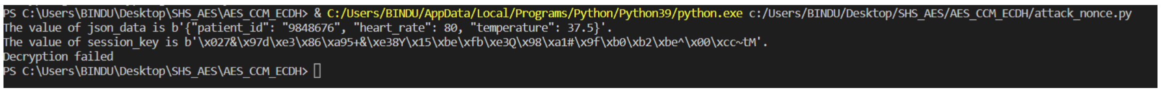
```
# Serialize the data to JSON and encode as bytes
json_data = json.dumps(data).encode()
print(f"The value of json_data is {json_data}.")
# Generate a random session key for 5G-AKA
session_key = token_bytes(32)
print(f"The value of session_key is {session_key}.")
# Use HKDF to derive a key for AES-CCM encryption
derived_key = HKDF(
    algorithm=hashes.SHA256(),
    length=32,
    salt=None,
    info=b'5G-AKA',
).derive(hkdf)
# Encrypt the nonce, JSON data, and session key using AES-CCM with the derived key
aesccm = AESCCM(key=derived_key, tag_length=8)
encrypted_data = aesccm.encrypt(nonce, json_data+session_key, None)
# Decrypt the encrypted data using AES-CCM with the wrong nonce
wrong_nonce = token_bytes(12)
try:
    decrypted_data = aesccm.decrypt(wrong_nonce, encrypted_data, None)
    decrypted_json_data, decrypted_session_key = decrypted_data[:-32], decrypted_data[-32:]
    if decrypted_json_data == json_data and decrypted_session_key == session_key:
        print("Authentication successful")
    else:
        print("Authentication failed")
except:
    print("Decryption failed")
```

Figure 5.16: Attack-3

- A patient ID, heart rate, and temperature are then included in an example data dictionary that is created by the code. The information is encoded as bytes and serialized to JSON. The token-bytes() function of the secrets module is used to create a session key.

- The nonce, JSON data, and session key are encrypted using AES-CCM with the derived key. The encrypted data is then decrypted using AES-CCM with the wrong nonce, which leads to a decryption failure.

- The attack happens when the wrong nonce is used for AES-CCM decryption. The nonce is randomly generated and used for both encryption and decryption.



```
PS C:\Users\BINDU\Desktop\SHS_AES\AES_CCM_ECDH> & C:/Users/BINDU/AppData/Local/Programs/Python/Python39/python.exe c:/Users/BINDU/Desktop/SHS_AES/AES_CCM_ECDH/attack_nonce.py
The value of json_data is b'{"patient_id": "9848676", "heart_rate": 80, "temperature": 37.5}'.
The value of session_key is b'\x027&\x97d\xe3\x86\xa95+&\xe38Y\x15\xbe\xfb\xe3Q\x98\xa1#\x9f\xb0\xb2\xbe^\x00\xcc~tM'.
Decryption failed
PS C:\Users\BINDU\Desktop\SHS_AES\AES_CCM_ECDH> []
```

Figure 5.17: Output of Attack-3

- If the wrong nonce is used for decryption, the decrypted data will be garbage. In this example, the decrypted data is expected to contain both the JSON data and the session key. If the decryption fails, the session key will not be correctly extracted, and authentication will fail.

- This attack is an example of a replay attack, where an attacker captures the encrypted data and then replays it with a different nonce to try to gain access to the data. The use of a random nonce for each encryption ensures that a replay attack cannot be successful, assuming the nonce is kept secret.

5.6 Observations

Using AES-CCM and ECDH encryption in Smart Healthcare Systems and 5G-AKA can mitigate these threats:

- *Spoofing*: The code uses asymmetric key cryptography, specifically Elliptic Curve Cryptography (ECC), to generate a private-public key pair for the recipient and an ephemeral key pair for each encryption operation. This ensures that only authorized recipients can decrypt the data, and each encryption operation uses a unique key pair to mitigate the threat of key reuse and unauthorized access. Spoofing threats involve an attacker pretending to be a legitimate user or system. By using ECDH key exchange, both parties can authenticate each other and ensure that they are communicating with the intended recipient. This prevents attackers from spoofing their identity and gaining access to sensitive data.

- *Tampering*: The code uses the AES-CCM encryption algorithm, which provides both confidentiality and integrity protection. This ensures that the encrypted data is protected against unauthorized modification. Tampering threats involve an attacker modifying data in transit. AES-CCM encryption ensures the confidentiality and integrity of the data, preventing attackers from reading or modifying the data without the appropriate keys.

- *Repudiation*: Repudiation threats involve a user denying that they performed a particular action. By the use of asymmetric key cryptography ensures that each encryption operation can be traced back to a specific recipient.

- *Information Disclosure*: Information disclosure threats involve an attacker gaining access to confidential data. By using AES-CCM encryption, sensitive data is protected from unauthorized access, even if an attacker gains access to the network. The code uses the AES-CCM encryption algorithm to safeguard the confidentiality of the data. Additionally, it uses a random nonce for each encryption operation to ensure that the same plaintext is not encrypted to the same ciphertext, further enhancing the confidentiality protection.

- *Denial of Service*: Denial of Service threats involve an attacker disrupting the availability of a system. By using ECDH key exchange, both parties can establish a secure communication channel that is resistant to DoS attacks.
- *Elevation of Privilege*: Elevation of Privilege threats involve an attacker gaining access to higher levels of access than they are authorized to have. By using ECDH key exchange, both parties can authenticate each other and ensure that users only have access to the data and resources that they are authorized to access.

Overall, using AES-CCM and ECDH encryption can help mitigate a wide range of security threats related to the STRIDE model in Smart Healthcare Systems and 5G-AKA. By implementing these security measures, organizations can improve the confidentiality, integrity, and availability of their sensitive data and resources.

5.7 Critical Analysis

In the context of 5G access key management for smart healthcare systems, there are difficulties with implementing AES-CCM and ECDH encryption. A critical aspect to consider is how cost and computational power will affect the implementation process.

- AES-CCM and ECDH encryption integration can be quite expensive for 5G-enabled smart healthcare systems. Included in this are the costs associated with purchasing particular hardware components, such as encryption modules or secure hardware modules, which are required for efficient encryption and decryption processes. Additionally, license or acquisition fees can be necessary for the fundamental cryptography software libraries or algorithms. These expenses may affect the overall budget and the feasibility of implementing effective encryption techniques.
- Another important aspect to take into account is computation power. AES-CCM and ECDH encryption methods demand a lot of processing power, especially when used extensively in 5G networks. To conduct encryption and decryption procedures effectively, the devices or systems in question must have enough processing power. Slower encryption/decryption speeds might impede real-time data transfer and system responsiveness due to insufficient computational power. Furthermore, the computing power needed for encryption can have a big impact on how much the devices' resources are used, including their CPU, memory, and battery usage. Ensuring effective and long-lasting operation demands meticulous tuning.
- Organizations must conduct a critical examination of the implementation to address these issues. In order to find cost-effective solutions that satisfy client security requirements, we must carefully assess the costs related to hardware parts and software licenses. To reduce the amount of processing power needed,

optimization techniques should be used. Improving efficiency and using fewer resources may entail using specialized cryptography hardware or optimizing software methods.

- It is crucial to remember that the precise cost and processing power factors may change based on the particular deployment scenario, implementation scale, and resource availability. As a result, companies should carefully evaluate their needs, financial limitations, and existing infrastructure to choose the best method for implementing AES-CCM and ECDH encryption in 5G access key management within SHS.

Cost-consciousness and effective computation are required when implementing AES-CCM and ECDH encryption in 5G access key management for smart healthcare systems. It involves controlling costs for specialized hardware and software and maximizing device efficiency. To effectively protect healthcare data in a rapidly evolving technological environment, organizations must strike a balance between efficiency and security, examine costs, and make wise decisions.

Chapter 6

Discussion

6.1 Research Questions

RQ1: Which risk factors can be identified by the SHS while conducting the STRIDE threat analysis?

1. *spoofing*: A risk that an attacker may impersonate a legitimate user or device to gain unauthorized access to the system, potentially leading to data theft, tampering, or other malicious activity.
2. *Tampering*: A risk that an attacker may intercept and modify data being transmitted between devices or servers, potentially compromising the confidentiality and integrity of the system.
3. *Repudiation*: A risk that an attacker may carry out an action on the system and then deny that they were responsible, making it difficult to identify and respond to security incidents.
4. *Information Disclosure*: A risk that sensitive patient or healthcare data may be accessed or disclosed to unauthorized individuals, potentially violating privacy regulations and exposing patients to harm.
5. *Denial of service*: A risk that an attacker may overload the system or disrupt communications between devices, potentially causing critical healthcare services to become unavailable and endangering patient lives.
6. *Elevation of privilage*: A risk that an attacker may gain unauthorized access to sensitive areas of the system or perform actions beyond their authorized privileges, potentially compromising the confidentiality, integrity, and availability of the system.

By identifying these risk factors, healthcare organizations can develop and implement effective security controls and risk management strategies to safeguard patient data and ensure the reliable delivery of critical healthcare services. This can include measures such as access controls, encryption, intrusion detection and prevention, and disaster recovery planning, among others.

RQ2: Which security and privacy controls could be utilized to address the threats outlined in the STRIDE threat in 5G Smart Healthcare Systems?

Ans: The STRIDE threat model provides a framework for identifying and addressing security and privacy risks in 5G Smart Health care systems. To address the threats of Spoofing, Tampering, Repudiation, Information Disclosure, and Denial of Service, the 5G-AKA protocol can be utilized.

1. For Spoofing, 5G-AKA provides multi-level authentication through Subscription Authentication, Serving Network Authentication, and User Authentication. This ensures that the entities involved in communication are authorized and authenticated at different levels.
2. To mitigate Tampering, the 5G-AKA protocol ensures strong authorization by satisfying UE authorization, Serving network authorization, and Accessing network authorization requirements. This ensures integrity and prevents unauthorized access to the system.
3. For Repudiation, 5G-AKA protocol provides non-repudiation and unlinkability in the initial stages of authentication and key agreement. This is done through the generation of Sequence Numbers or token values to compute the symmetric signature.
4. For Information Disclosure, 5G-AKA protocol ensures confidentiality by maintaining a shared secret key between UE and HN, which completely depends upon the Secret Key.
5. To address Denial of Service, the 5G-AKA protocol verifies the identity of the SN at HN using the MAC transmitted by the UE. This helps to protect the communication entities from the Man in the Middle and redirection attacks.

In summary, the 5G-AKA protocol provides a comprehensive security and privacy mechanism to address the threats outlined in the STRIDE model. It ensures multi-level authentication, strong authorization, non-repudiation, confidentiality, and availability. These measures can enhance the security and privacy of 5G Smart Health care systems.

RQ3: How we can enhance 5G AKA to offer optimal safety and privacy in smart healthcare systems?

Ans: While the 5G AKA protocol provides a comprehensive security mechanism to address the security and privacy risks in smart healthcare systems, there are some limitations that can prevent its full potential from being realized. Here are some potential solutions to overcome these limitations:

1. *Key management*: The 5G AKA protocol requires the maintenance of keys between UE and HN, which can be a challenge for large-scale systems with multiple UEs. One solution to this is to use a key management system that can dynamically generate and distribute keys as needed, reducing the burden of key management.

2. *User Authentication*: The current 5G AKA protocol relies on user authentication through the use of SIM cards, which may not be suitable for certain applications, such as those that require frequent user identity changes. One solution to this is to use alternative authentication mechanisms, such as biometric authentication, to enable continuous authentication without the need for a physical SIM card.

3. *Intergration with other protocols*: The 5G AKA protocol may not be compatible with other security protocols used in smart healthcare systems. One solution to this is to integrate the 5G AKA protocol with other security protocols, to ensure seamless compatibility and enhance overall security.

4. *Privacy concerns*: While the 5G AKA protocol provides confidentiality, there are still privacy concerns related to the collection and use of personal data. One solution to this is to implement privacy-enhancing technologies, such as differential privacy or homomorphic encryption, to protect sensitive data while still enabling data analysis and processing.

5. *Robustness to cyber Attacks*: The 5G AKA protocol is susceptible to various cyber attacks, such as replay attacks, man-in-the-middle attacks, and denial-of-service attacks. One solution to this is to continuously monitor the system for cyber threats and implement appropriate countermeasures, such as intrusion detection systems, firewalls, and encryption.

In conclusion, while the 5G AKA protocol provides a strong security mechanism for smart healthcare systems, its limitations can prevent its full potential from being realized. Addressing these limitations through key management, alternative authentication mechanisms, integration with other protocols, privacy-enhancing technologies, and robustness to cyber attacks can help to overcome these challenges and enhance the security and privacy of smart healthcare systems.

RQ4: How can AES-CCM with ECDH be implemented in an SHS to ensure secure communication between healthcare providers and patients?

To implement AES-CCM with ECDH in a smart health care system (SHS), the following steps can be taken:

1. *Key Generation*: The first step is to generate the keys for ECDH and AES-CCM. For ECDH, each healthcare provider and patient generate their own public and private keys. The public key is then exchanged between the two parties. For AES-CCM, a shared secret key is generated using the ECDH public and private keys.

2. *Encryption and Decryption*: Once the keys have been generated, AES-CCM can be used to encrypt and decrypt messages between healthcare providers and patients. Before encrypting the message, the AES-CCM algorithm adds authentication and integrity data to the message, which ensures that the message has not been tampered with during transmission. The encrypted message is then sent to the recipient.

3. *Authentication*: To ensure that only authorized parties can access the SHS, authentication mechanisms can be implemented, such as password-based authentication or multi-factor authentication. The ECDH public and private keys can also be used for authentication purposes.

4. *Key Management*: It is important to manage the keys used in the SHS to ensure that they are secure and up-to-date. This can be done through key rotation or key revocation, where keys are periodically changed or revoked to prevent unauthorized access.

5. *Security Monitoring*: Finally, it is important to continuously monitor the SHS for security threats and vulnerabilities, such as unauthorized access, data breaches, and malware attacks. This can be done through intrusion detection systems, firewalls, and other security measures.

In conclusion, implementing AES-CCM with ECDH in an SHS can provide a secure communication channel between healthcare providers and patients. By following the steps outlined above, the security and privacy of the SHS can be enhanced, ensuring that sensitive patient information is protected from unauthorized access and tampering.

6.2 Contribution to Thesis

A well-known and efficient method for locating potential threats to software systems is the STRIDE threat modeling approach. We were able to recognize six sorts of security risks by using this method on the Smart Healthcare System, including spoofing, tampering, repudiation, information leakage, denial of service, and elevation of privilege.

The act of pretending to be a valid user or system in order to obtain unauthorized access to the Smart Healthcare System is known as spoofing. Tampering entails changing the system's data or code to do harm. Repudiation is the term for an attacker's denial of an activity or event, making it challenging to track or identify them. The unintentional or intentional exposure of private information to unapproved parties is known as information disclosure. Elevation of privilege refers to an attacker getting access to critical areas of the system that they shouldn't have access to, whereas denial of service refers to an attacker interfering with the system's normal operation.

After identifying these threats, we put up a solution that makes use of AES-CCM and ECDH encryption techniques to offer extremely high security for communication between smart healthcare devices and the 5G network. A shared secret key can be agreed upon between two parties using the key agreement protocol ECDH without the key being revealed to anybody else. This means that even if the transmission is intercepted, no one will be able to access the private information transferred between the devices and the 5G network. Data secrecy, integrity, and authenticity are all features of the AES-CCM encryption method, which makes it a very safe way to encrypt data.

We carried out an experiment to verify the efficacy of our proposed strategy. We conducted an experiment to evaluate the security of the Smart Healthcare System using a variety of attack simulations. The experiment's findings demonstrated how well our strategy worked to lessen the identified security threats. This offers more proof that the Smart Healthcare System and other IoT devices can actually be made more secure using the strategy we advise.

By identifying potential security vulnerabilities and offering a way to mitigate them, this thesis makes a significant contribution to the domains of 5G security and smart healthcare systems. Our findings can serve as a springboard for additional study in this field and can offer recommendations for making safe IoT gadgets. Developers can improve the security of their IoT devices and safeguard sensitive data from potential attackers by employing our recommended strategy.

6.3 Validity

Since 5G technology has the potential to completely alter how healthcare is delivered, it has attracted a lot of attention in recent years. Real-time communication between patients and healthcare providers is made possible by 5G technology's high speed and low latency capabilities, enhancing the effectiveness and quality of healthcare services. Sensitive patient data is carried across wireless networks that could be exposed to a variety of security risks, therefore using 5G technology in healthcare also presents security issues.

We employed a systematic approach known as STRIDE threat modeling to identify potential security risks and vulnerabilities in the smart healthcare system to solve these security issues. The most frequent security risks that might happen in any system are referred to as STRIDE, which stands for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege. The study is made more credible and valid by the use of STRIDE threat modeling, which makes sure that all potential security concerns are discovered and assessed.

In addition, we have AES-CCM and ECDH encryption in place to mitigate the risks found in the STRIDE investigation. A symmetric encryption algorithm called AES-CCM protects the confidentiality and integrity of data sent via wireless networks. Contrarily, the 5G AKA network and smart medical equipment may communicate securely thanks to the key exchange technique ECDH.

The overall security of the smart healthcare system is enhanced by the employment of these encryption techniques, which serve to protect the confidentiality, integrity, and availability of critical patient data. The inquiry into 5G-secured smart healthcare systems is made more credible by the addition of these security measures.

In conclusion, the investigation into smart healthcare systems using 5G security is legitimate because of the use of a strict research methodology, the importance of the subject, and the application of efficient security measures. The study contributes to continuing efforts to enhance healthcare delivery through the use of 5G technology by offering useful insights into the security difficulties faced by smart healthcare systems and offering doable ways to reduce such risks.

Chapter 7

Conclusions and Future Work

In conclusion, our thesis focused on the implementation of AES-CCM and ECDH encryption over the Smart Health Care System (SHS) and 5G Authentication and Key Agreement (AKA) protocol. Through a detailed description of the implementation process, we have highlighted how this approach significantly enhances the security and privacy of patient information, effectively protecting sensitive data from unauthorized access and tampering.

By utilizing AES-CCM and ECDH encryption, we have demonstrated that efficient and secure communication can be achieved within the SHS environment. The generation of a shared secret key over an insecure channel ensures the confidentiality and integrity of the messages exchanged between healthcare servers and devices. The algorithm for this protocol involves key pair generation, key exchange, shared secret key derivation, nonce generation, message encryption, message authentication, and message verification. The provided flowchart visually represents the step-by-step encryption process.

Moreover, the integration of the Smart Health Care System with 5G technology brings forth additional benefits and advancements. The 5G network's high data speeds, low latency, reliability, and massive connectivity enable seamless and real-time communication between healthcare providers and devices. This facilitates remote healthcare support, telemedicine consultations, and the incorporation of emerging technologies such as augmented reality (AR) and virtual reality (VR) into healthcare practices.

The implementation of AES-CCM and ECDH encryption, combined with 5G technology, strengthens the security measures and privacy standards within the Smart Health Care System. Patient information remains protected, ensuring that sensitive data is not vulnerable to unauthorized access or tampering. The enhanced communication capabilities provided by 5G enable healthcare professionals to deliver improved patient care, offer remote monitoring and diagnosis, and optimize resource utilization.

Overall, the implementation of AES-CCM and ECDH encryption over the SHS and 5G AKA protocol presents a comprehensive and effective approach to secure communication within the healthcare domain. This solution empowers healthcare providers to maintain the confidentiality, integrity, and privacy of patient information, ultimately leading to enhanced healthcare services and improved patient outcomes. The combination of advanced encryption techniques, the Smart Health Care System, and 5G technology paves the way for a transformative and secure healthcare landscape.

Moving forward, we suggest future work to investigate the scalability of the protocol. Healthcare systems often involve many devices and servers, and it is crucial to ensure that the proposed protocol can handle multiple devices and servers and a high volume of traffic. This will help ensure that the protocol can meet the demands of a large-scale healthcare system without compromising security or performance.

Another critical area for future work is the implementation of the protocol in a real-world healthcare setting and evaluating its performance, security, and usability. This will provide valuable insights into how the protocol performs in a practical environment and can help identify potential issues or areas for improvement.

In addition, researchers can also focus on investigating key management and interoperability aspects of the protocol. Key management is critical for any encryption protocol, and it is important to explore how the proposed protocol can handle key distribution and revocation, as well as prevent key compromise. Interoperability is also essential, and future work can focus on how the protocol can be integrated with other security mechanisms, such as firewalls, intrusion detection systems, and antivirus software.

Finally, researchers can also explore the standardization of the protocol, ensuring that it meets regulatory requirements and can be adopted by healthcare organizations. Cost-effectiveness is also an important consideration, and future work can investigate the cost-effectiveness of the proposed protocol compared to other existing protocols.

Overall, the implementation of the AES-CCM and ECDH encryption over SHS and 5G AKA protocol provides a solid foundation for securing communication in healthcare systems. By investigating the suggested future work, researchers can enhance the security and privacy of patient information and develop more effective protocols to meet the demands of modern healthcare systems.

References

[1] R. Ahmed, M. H. Ahmed, S. U. Hassan, and A. H. Baig. Internet of things (iot)-enabled healthcare system: A comprehensive review. *IEEE Reviews in Biomedical Engineering*, 14:284–298, 2021.

[2] N. Aljohani, R. A. M. Almurshidi, M. M. Alanezi, and O. M. Alharbi. Smart healthcare monitoring system using internet of things and cloud computing. In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 684–689, 2020.

[3] M. Zohrehvand, O. A. Dobre, A. Abdi, and M. Zolghadri. 5g networks: Opportunities and challenges. *IEEE Internet of Things Journal*, 7(11):10229–10249, 2020.

[4] R. Islam, N. Islam, D. Kwak, M. S. Hossain, A. Almogren, and A. Alamri. Smart healthcare systems and 5g networks: Opportunities and challenges. *Sensors*, 20(10):2888, 2020.

[5] Enjie Liu, Emmanuel Effiok, and Jon Hitchcock. Survey on health care applications in 5g networks. *IET Communications*, 14(7):1073–1080, 2020.

[6] Andrea Moglia, Konstantinos Georgiou, Blagoi Marinov, Evangelos Georgiou, Raffaella Nice Berchiolli, Richard M. Satava, and Alfred Cuschieri. 5g in healthcare: From covid-19 to future challenges. *IEEE Journal of Biomedical and Health Informatics*, 26(8):4187–4196, 2022.

[7] Sathian Dananjayan and Gerard Marshall Raj. 5g in healthcare: How fast will be the transformation? *Irish Journal of Medical Science (1971 -)*, 190(2):497–501, 2021.

[8] S. Bhattacharya. The impact of 5g technologies on healthcare. *Indian Journal of Surgery*, pages 1–6, 2022.

[9] Mohammad Kamrul Hasan, Taher M. Ghazal, Rashid A. Saeed, Bishwajeet Pandey, Hardik Gohel, Ala' A. Eshmawi, S. Abdel-Khalek, and Hula Mahmoud Alkhassawneh. A review on security threats, vulnerabilities, and counter measures of 5g enabled internet-of-medical-things. *IET Communications*, 16(5):421–432, 2022.

[10] Patrick Barba, Joshua Stramiello, Emily K. Funk, Florian Richter, Michael C. Yip, and Ryan K. Orosco. Remote telesurgery in humans: A systematic review. *Surgical Endoscopy*, 36(5):2771–2777, 2022.

[11] Anjali Sharma and Ritu Arora. A comparative analysis of symmetric key

cryptography algorithms. *International Journal of Computer Applications*, 171(16):33–38, 2017.

[12] Md Faruk Zaman. A review of cryptographic algorithms and their security issues. *International Journal of Computer Science and Information Security*, 16(8):48–58, 2018.

[13] Hong Liu, Xuanxia Yao, Tao Yang, and Huansheng Ning. Cooperative privacy preservation for wearable devices in hybrid computing-based smart health. *IEEE Internet of Things Journal*, 6(2):1352–1362, 2019.

[14] Iynkaran Natgunanathan, Abid Mehmood, Yong Xiang, Longxiang Gao, and Shui Yu. Location privacy protection in smart health care system. *IEEE Internet of Things Journal*, 6(2):3055–3069, 2019.

[15] Jianfei Sun, Hu Xiong, Ximeng Liu, Yinghui Zhang, Xuyun Nie, and Robert H. Deng. Lightweight and privacy-aware fine-grained access control for iot-oriented smart health. *IEEE Internet of Things Journal*, 7(7):6566–6575, 2020.

[16] Xiaoyong Tang, Lijun Zhao, Jing Chong, Zhengpeng You, Lei Zhu, Haiying Ren, Yuxiang Shang, Yantao Han, and Gong Li. 5g-based smart healthcare system designing and field trial in hospitals. *IET Communications*, 16(1):1–13, 2022.

[17] Baozhan Chen, Siyuan Qiao, Jie Zhao, Dongqing Liu, Xiaobing Shi, Minzhao Lyu, Haotian Chen, Huimin Lu, and Yunkai Zhai. A security awareness and protection system for 5g smart healthcare based on zero-trust architecture. *IEEE Internet of Things Journal*, 8(13):10248–10263, 2021.

[18] Lei Yang, Shuai Zhang, Feng Tian, Shuai Ren, and Hongnian Yu. A secure data sharing scheme based on aes for cloud-based electronic health records. *IEEE Access*, 8:124444–124452, 2020.

[19] Yuyu Zhang, Yu Zhu, Yufeng Cheng, Hongliang Zhang, and Jing Wang. A secure data encryption scheme for mobile health systems. *IEEE Access*, 7:64391–64399, 2019.

[20] S. Singh and S. Kaur. A comparative study of symmetric key encryption algorithms for secure communication in smart healthcare systems. *International Journal of Computer Applications*, 179(22):6–11, 2018.

[21] Alok Gupta and Amarjeet Singh. A comparative study of cryptographic algorithms for securing smart healthcare systems. *Journal of Medical Systems*, 43(7):1–9, 2019.

[22] Yalong Tang, Xiang Chen, and Xinhua Wang. Smart healthcare system architecture based on internet of things. *International Journal of Distributed Sensor Networks*, 17(1):1550147721994408, 2021.

[23] Naser Alsafi, Steven L Gao, and Majid Alshammari. Smart healthcare system architecture using iot and cloud computing. *Wireless Communications and Mobile Computing*, 2021:6647141, 2021.

[24] Ke Yang, Jinchang Ren, and Xiaochun Liu. A privacy-preserving and secure iot-based healthcare system with edge computing. *IEEE Transactions on Industrial Informatics*, 17(6):4065–4073, 2021.

[25] S. A. Al-Kaabi, M. M. Hassan, and M. Al-Qutayri. Security analysis of smart healthcare system using blockchain technology. In *2020 International Wireless Communications and Mobile Computing (IWCMC)*, pages 2392–2397. IEEE, 2020.

[26] Mohammed H. B. Al Rawi and Mahmoud Al-Qutayri. A blockchain-based framework for securing healthcare systems. *IEEE Access*, 9:48076–48090, 2021.

[27] Muhammad Bilal, M Anwar Hossain, and Muhammad Mostafa Monowar. Smart healthcare: Emerging opportunities and challenges. *International Journal of Information Management*, 49:431–438, 2019.

[28] N. Jeyanthi and V. Rajamani. The stride model: A framework for threat analysis in smart healthcare systems. *Journal of Medical Systems*, 45(7):1–9, 2021.

[29] Amirhosein Talaei-Khoei, Ali Dehghantanha, and Kim-Kwang Raymond Choo. Deep learning-based security and privacy in smart healthcare systems: A review. *IEEE Access*, 9:53945–53960, 2021.

[30] Loren Kohnfelder, David Rine, and Gregory White. Stride: A threat modeling tool for identifying security threats. *Journal of Information Warfare*, 4(2):31–42, 2005.

[31] Abdulrahman Al-Hezmi, Abdulaziz Al-Nahari, Mohammed Alqahtani, Ahmad Alkahtani, and Abdulrahman Alghamdi. 5g security: analysis of potential threats and solutions. *Wireless Networks*, 27(6):4413–4429, 2021.

[32] Yutao Sun, Ang Li, Yong Liu, Lei Zhou, and Zhe Fang. 5g core networks: security challenges and opportunities. *IEEE Communications Magazine*, 59(5):50–56, 2021.

[33] Hongbing Song, Lei Lu, and Hongxiang Jiang. 5g core network architecture and security. In *2020 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 135–140. IEEE, 2020.

[34] S. Srinivasan and A. Ananth. A novel authentication protocol for healthcare system using aka algorithm. *International Journal of Engineering and Technology Innovation*, 11(1):53–60, 2021.

[35] M. T. Dehghani and A. R. Sadeghnejad. Secure authentication scheme for e-healthcare systems using aka protocol and blockchain technology. *Journal of Medical Systems*, 45(12):174, 2021.

[36] A. Al-Zoubi and S. Alkhraisat. Secure authentication scheme for mobile healthcare using aka protocol and blockchain. *Journal of Ambient Intelligence and Humanized Computing*, 12(12):11851–11860, 2021.

[37] Mohammad Ali Ebrahimi, Farzaneh Khodaei, and Hossein Nezamabadi-Pour. A comparative study of lightweight block ciphers in the iot environment. *IEEE Internet of Things Journal*, 8(14):11576–11587, 2021.

[38] K Raja and S S S Srinivas. Performance analysis of lightweight encryption algorithms for iot devices. In *Proceedings of the International Conference on Computational Intelligence and Data Science*, pages 403–413. Springer, 2020.

[39] Abiodun Olusola Afolabi, Adedoyin Adeyinka Alaba, Olumide Olawale Olabiyi, and Olumide Oluwaseun Akintola. Lightweight cryptography for the internet of things: A review. *International Journal of Communication Systems*, 34(12):e4817, 2021.

[40] Abdellah Benali and Abdelhakim Erritali. Performance evaluation of lightweight encryption algorithms for iot devices. In *International Conference on Computational Science and Its Applications*, pages 293–303. Springer, 2021.

[41] Muhammad Umer Rehman, Muhammad Umar Farooq, and Muhammad Asad Khan. A comprehensive performance evaluation of lightweight block ciphers for iot devices. *IEEE Access*, 8:100303–100319, 2020.

[42] Yanli Sun, Xiong Li, Yan Li, and Keqiu Zhang. A lightweight authentication and data confidentiality scheme for industrial internet of things. *IEEE Transactions on Industrial Informatics*, 17(6):4276–4285, 2021.

[43] Mohammed Al-Haidari, Mznah Al-Rodhaan, and Abdullah Al-Dhelaan. A comparative study of aes-ccm and chacha20-poly1305 in iot-based healthcare systems. *Sensors*, 21(15):5034, 2021.

[44] Adel Ammar, Ahmed Ben Abid, and Abdelfettah Belghith. Performance evaluation of dhe and ecdh key exchange algorithms for secure communication in iot. In *2021 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–6. IEEE, 2021.

[45] Weiqiang Chen, Xinhui Wang, and Xiangyang Li. Security analysis of elliptic curve diffie-hellman key exchange protocol for the internet of things. *Wireless Personal Communications*, 102(4):3169–3185, 2018.

Faculty of Faculty of Telecommunication Systems, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden