A
Dissertation
On

Bluetooth enabled secure ABE based IoT health sensor

Submitted in Partial Fulfillment of the Requirement
For the Award of the Degree of

Master of Technology

in

Computer Science and Engineering

By

Suraj Singh
University Roll No. 2K14/CSE/19

Under the Esteemed Guidance of

Ms Divyashikha Sethia
Assistant Professor

Computer Science & Engineering Department, DTU

COMPUTER SCIENCE & ENGINEERING DEPARTMENT

DELHI TECHNOLOGICAL UNIVERSITY

DELHI - 110042, INDIA

2014-2016

ABSTRACT

Electronic Health data is having a profound and increasing impact in the area of mobile health services. This electronic health data is stored, transferred and processed in these systems with which comes the privacy, authentication, secure storage and accountability issues. In this paper, we propose an approach which provides end-to-end security by encrypted data storage and transmission. We propose a unique Secure Health Sensor Data (SHSD) system with embedded secure microcontroller. This system communicates with systems such as laptop, mobile etc. through Bluetooth interface. Our design at present comprises use of motion sensor i.e. accelerometer based sensor which can be used to detect the x, y, z coordinates of human body as well as fall detection of elderly people. The data is generated, stored and transferred by SHSD by making use of Raspberry Pi (a single board computer system) to a mobile device for e.g. Laptops, Phones etc. We propose a novel approach to implement this on embedded Java Card OS and energy and cost efficient single board system such as Raspberry Pi. Java Card API provides secure and efficient OS defined objects for storage of sensitive data and for cryptographic mechanisms. Access to the data stored in the microcontroller can be safeguarded by controlled access policies enforced by an Applet. The security issues has been addressed in multiple ways. Firstly, the loss of sensor data is prevented by making use of wired connection from Raspberry Pi to ADXL motion sensor chip. Secondly, a secure communication is ensured by encryption of the sensor data. Thirdly, the symmetric key is stored in the tamper resistant secure element of the Java Card. Fourth, Attribute based encryption (ABE) which is fine grained secure access control technique. Thereby, make it impossible for adversary to gain access to sensor data. The combined use of Raspberry Pi and Java Card increase the security of the system and make it more accountable.

Keywords: Health Services, Raspberry Pi, Java Card, Privacy, Authentication, Secure Storage, Motion Sensor, CP-ABE

ACKNOWLEDGEMENT

First of all, I would like to express my deep sense of respect and gratitude to my project supervisor Ms Divyashikha Sethia for providing the opportunity of carrying out this project and being the guiding force behind this work. I am deeply indebted to her for the support, advice and encouragement she provided without which the project could not have been a success.

Secondly, I am grateful to Dr. O.P. Verma, HOD, Computer Science & Engineering Department, DTU for his immense support. I would also like to acknowledge Delhi Technological University library and staff for providing the right academic resources and environment for this work to be carried out.

Last but not the least I would like to express sincere gratitude to my parents and friends for constantly encouraging me during the completion of work.

Suraj Singh

University Roll no: 2K14/CSE/19

M.Tech (Computer Science & Engineering)

Department of Computer Science & Engineering

Delhi Technological University

Delhi - 110042

Certificate

This is to certify that the dissertation titled **"Bluetooth enabled secure ABE based IoT health sensor "** is a bonafide record of work done by **Suraj Singh, Roll No. 2K14/CSE/19** at **Delhi Technological University** for partial fulfilment of the requirements for the degree of Master of Technology in Computer Science & Engineering. This project was carried out under my supervision and has not been submitted elsewhere, either in part or full, for the award of any other degree or diploma to the best of my knowledge and belief.

(Ms Divyashikha Sethia)

Project Guide

Date: _ _ ___

Department of Computer Engineering

Delhi Technological University

TABLE OF CONTENTS

List of Figures

List of Tables

List of Abbreviations

SHS : Secure Health Sensor

PKC : Public Key Cryptography

SKC : Symmetric Key Cryptography

BLE : Bluetooth Low Energy

AES : Advanced Encryption Standard

ABE : Attribute Based Encryption

CPABE : Ciphertext Policy Attribute Based Encryption

KPABE : Key Policy Attribute Based Encryption

RSA : Rivest Shamir Adleman

ECC : Elliptic Curve Cryptography

CA : Certificate Authority

PKI : Public Key Infrastructure

PKE : Public Key Encryption

PKS : Public Key Signature

WSN : Wireless Sensor

IBC : Identity Based Cryptogrphy

PKG : Public Key Generator

ECDLP : Elliptic Curve Discrete Logarithmm Problem

IFP : Integer Factorization Problem

SBC : Single Board Computer

FIFO : First In First Out

TTL : Time To Live

BPM : Beats Per Minute

RBAC : Role Based Access Control

HTTPS : Hyper Text Transport Protocol Security

TLS : Transport Layer Security

CSR : Certificate Signing Request

DES : Data Encryption Standard

CHAPTER 1

INTRODUCTION

E-health is the transfer of health resources and health care by electronic means. E-health defines the usage of health information, for health professionals and health consumers, via Internet and web communications. Providing e-health using the power of information technology to improve public health services, example through the education and training of health workers [1]. The patient concerned with medical data administration service platform is a healthcare service platform that matches the Health Care 3.0 era to meet the needs of the construction of smart healthcare service platform. The architecture comprehensively handles both aspects that are medical data of hospital or patient care system and patient health related data measured using devices such as medical sensors, medical watch tags that provide twenty hour surveillance on patient health. It also provide health information not only to their own medical supervisors and specific hospital information system under the patient's consent but also to patients and general users [1][2].

When employing personalized patient oriented healthcare service platform, the perimeter of health service is extended from the health center to a patient home area, business place or other fitness centers.

The system can reduce repeated visits to the hospital and save in national healthcare costs. Therefore, personal healthcare service platform to support the whole management of personal health data using the smart mobile devices is necessary to customize day life healthcare.

The health information is stored in the database for future retrieval of the patient health status or health history. The data can be used in the future medical diagnosis of the patient. The previous measures can be used to trace up the history of the health status and medication that the patient have been through. In this way the health professional has the guide in his/her health analysis. If not secured the Bad guy can modify the medical record of the patient and this will lead to wrong diagnosis. Health services is a major requirement for both developed countries, where the cost of healthcare is high and security and privacy are critical issues and developing countries like India, where there is huge population to control in hospitals. A

large population will reside in nursing homes and hospitals in near future. According to a study [4], within the next decade, there will be more people aged 65 years and older, than children under five in the world. More number of senior citizens are admitted to hospitals to mitigate the dangers of elderly falling. Among elderly persons, 55 percent of fall injuries occur in person's home. And 23 percent fall accidents occur outside old person's home [3]. This has cause a major shift of these elderly persons to hospitals, care centers where they are put on continuous monitoring. Various sensors are attached to these patients to make continuous health monitoring a possible scenario. With this, these health centers will require continuous medical monitoring, medical data access and emergency communication. An efficient, reliable, robust and secure health flow is important to manage patients, their health records securely and to take the right health service accessible to the patient at the right time. Privacy and security is a very important aspect of healthcare [5]. The body sensor module will be extremely beneficial to the patients and their caretakers who will be able to monitor their patient's health and improve the quality of healthcare, increasing safety and reducing the overall cost of the healthcare incurred by patients**.** These devices are useful in biometric and medical applications for real time monitoring of a patient's state or for acquiring sensitive data which can be used to provide the correct medical diagnosis. Identification of objects for secure medical procedures is very essential for a secure workflow.

Body sensor networks monitor health parameters using body sensor devices [6]. These devices are useful in biometric and medical applications for real-time monitoring of a patient's state or for acquiring sensitive data which can be subsequently analyzed to provide a medical diagnosis. The initial motivation of the work was to develop a secure sensor node prototype for developing secure body sensors. Most of the sensors in the market overlook the security aspect. Raspberry Pi [7] has been used for initial deployment of a secure body sensor based on accelerometer which can be used as a fall detection sensor in elderly people. The system has used accelerometer sensor for initial development of a prototype since we can get the changes in the readings promptly. With this design the sensors for temperature, blood pressure, oxymeter etc can similarly be incorporated in the design to gather vital health parameters. Raspberry Pi has been chosen rightly as the single board computer for this application because it has the highest performance to cost ratio and is one of the smallest single board computers available in the market. This paper describes in detail, the components, design and functioning of one secure sensor node prototype which has been developed, tested and verified for the development of a body sensor module. The sensor

information gathered can be communicated wirelessly to the mobile phone using any of the following three connectivity options: Bluetooth, Bluetooth light or NFC, depending on the application. In this project, Bluetooth technology has been used. It is also important to protect the access structure (defined rules) which is associated with the encrypted data, because sometimes theses access structure might contain vital information about the persons, who are encrypting and decrypting the actual information. Sometimes these access structure may become more important because malicious user may guess the actual information by using the access structure defined by the encryptor.

1.1 Motivation

In recent years, the rapid improvement in technology has been influential in designing the healthcare system. Consequently, many researchers are focusing in u-healthcare system development. The u-healthcare system application and devices were given much attention to provide ubiquity in healthcare services. Now a day, the used of the internet is very essential in our daily transactions. Almost everything rely on the use of the internet, like business, education, security and others are using internet as the main medium to deliver information. Now, medical field is also adopting the use of IT. System used internet-based because this is the best way to deploy the system which can be access anytime, anywhere in internet. This practice is very helpful for the Physician to access the previous medical history of the patient that they are handling. In this case they have the background, and this could help them in their analysis. Mobile devices communicating in networks are extremely vulnerable to software-based attacks and require secure communication with sensor nodes. In this paper, system are SHS system is developed based on work [8] where system used encryption of sensor data using RC4 encryption algorithm. Here, system make use of AES for symmetric encryption and CP-ABE for fine grained access control. The Plug-n-Trust module [6], where a mobile phone is responsible to collect the data from the various sensors (connected to the body) suffers from security issues [9].The secure sensor node prototype designed with the Raspberry Pi [22] make use of similar architecture but does not make use of fine grained access control to safeguard the application. [23][24] use Raspberry Pi and sensors to gather the data but data storage and transmission does not incorporate any security. [25] used Beagle Bone Black development board with an embedded Linux distribution in IoT based system but also does not provide security of data transmission as well as storage. Our system guarantees more security as compared to Plug-n-Trust module [6], Wireless Sensor network using

Raspberry Pi and ZigBee [23], portable spectrometric sensor platform [24], data collector service [25] and secure sensor node using Raspberry Pi [22]:

- Wired communication occurs between the Raspberry Pi and the sensor so there is no fear of information loss or security.
- The collected sensor data is stored in the encrypted form.
- The data transmission always happens in encrypted form.
- Encryption is used twice to increase the security of the system. First, the sensor data is encrypted using the 128-bit AES key followed by encrypting the same AES key using the fine grained access control technique CP-ABE.

1.2 Research Objective

With the motivation explained in the previous section, the objective of our research work can be identified as:

- The system should be energy efficient.
- The system should provide APIs to safely record the patient readings.
- The system should be able to store the readings in a secure storage.
- Data should be encrypted before transmission.
- The system should allow users with necessary attributes to decrypt the sensitive medical data in multi-user environments.
- The system should provide mechanism to do the auditing to provide the non-repudiation property.
- The system should provide way for cipher text transmission using Bluetooth transmission or web interface.

1.3 Thesis Organization

We start this dissertation with introduction in chapter 1. A detailed description of background is presented in chapter 2 which includes cryptographic techniques & its applications, different threats to wireless networks such as SHS and signature and certificate schemes to strengthen the security and provide confidentiality, non-repudiation, integrity which are very important parameters in any wireless sensor networks. The components of SHS are also described in detail. Chapter 3 explains about proposed problem statement with its proposed solution. Chapter 3 also gives a brief about the system design of SHS and the

implementation. Chapter 3 also explains in detail about signature and certificate schemes used in SHS. We evaluate the perf rmance of the proposed design and signature and certificates technique in chapter 4. We conclude about the work done and observations in chapter 5. This includes the future scope; which is promising with the use of Java card as storage element for key.

CHAPTER 2

LITERATURE REVIEW

2.1 Cryptographic Techniques

Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication [10]. Cryptographic goals are confidentiality, authentication, data integrity and non-repudiation. Cryptographic techniques are typically divided into two generic types: symmetric-key and public-key. Figure 2.1 provides a schematic listing of the cryptographic primitives considered and how they relate [10].

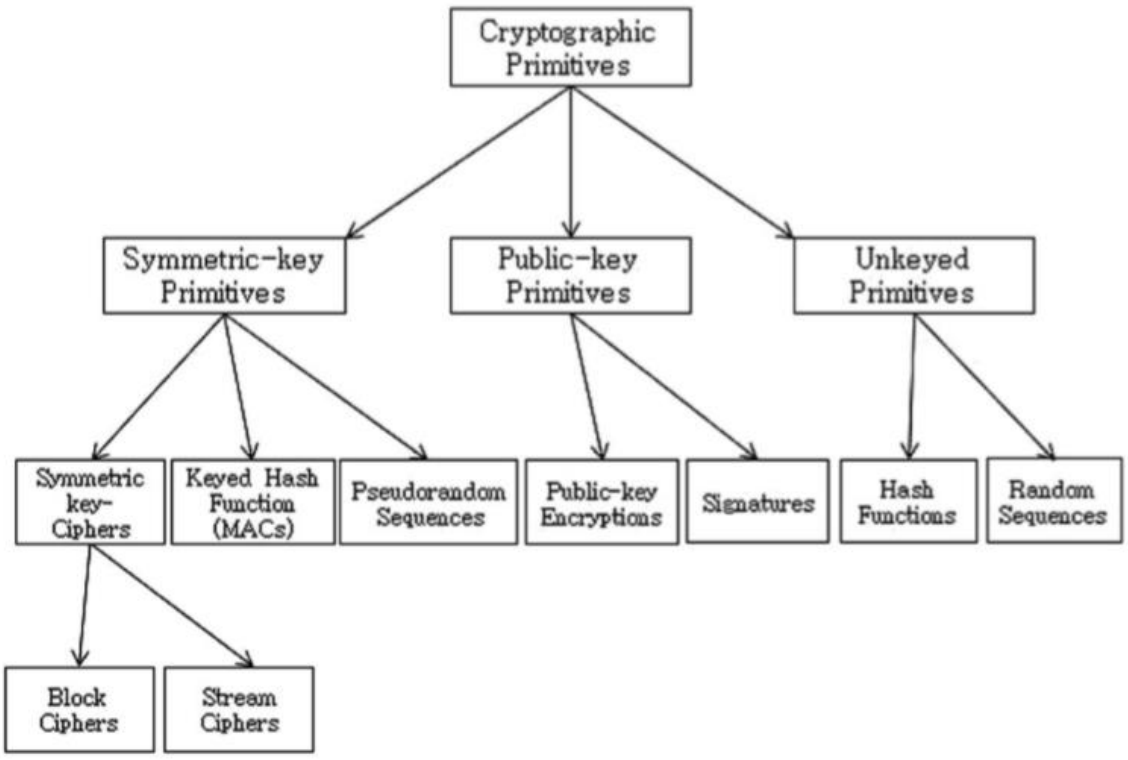


Figure 2.1: Taxonomy of Cryptographic Primitives

2.1.1 PKC Versus Symmetric-Key Cryptography

Symmetric-key cryptography (SKC) is also known as shared key, single-key, and private-key

cryptography. In this type of message encryption, sender and receiver only have to share the same private key in the beginning and then they can begin to encrypt and decrypt messages between them using that key. This key pre distribution process is very difficult. SKC cannot achieve non-repudiation, as both sender and receiver use the same key, messages cannot be verified to have come from a particular user. PKC, also known as asymmetric cryptography uses two keys: a private key, which has to be kept secret, and a public key, which is publicly known. Any operation done with the private key can only be reversed with the public key, and vice versa. This nice property makes all PKC-based algorithms useful for secure broadcasting and authentication purposes. It is also an invaluable tool for allowing the secure exchange of private keys between previously unknown partners. The public key in PKCs must be authenticated. Public-Key Infrastructure (PKI) solves the public key authentication problem using a public-key certificate issued by a Certification Authority (CA). Computational cost of PKC had hindered its application in highly-constrained devices, such as sensor nodes, while an approach using SKC offers advantages in terms of low communication and computational overhead. One may believe that SKC is more suitable for SHS based wireless sensor networks used in health monitoring applications which requires only confidentiality or data integrity. To apply SKC to these health monitoring WSNs, the shared-key distribution is needed. The key pre distribution methods have the following three types :

- A single network-wise private key: this causes a single point failure, i.e., if the private key of a node is revealed then the entire network is broken.
- A pairwise key between a node and the BS or between two nodes: the pairwise keying is very difficult and inefficient, i.e., each node must share ${}^{n}C_{2} = n(n-1)2$ private keys, where n is the number of the sensor nodes or stakeholders . This creates a problem with managing and ensuring the security of all these keys. If the private key of a node is revealed, then the other node with the same key is also compromised.
- A group key among a set of nodes: group keying is more inefficient than the pair wise keying as it is required heavy computational overhead and interactions with more than two rounds among nodes. If the group key of a node in a group is revealed, then all the group of nodes is compromised.

To minimize the effects of private key exposure is an important factor. In fact, the security schemes should guarantee that no matter how many nodes are captured, the private

information extracted from the compromised nodes cannot affect the security among non-compromised nodes, i.e., communications among non-compromised nodes remain secure. However, the three types above cannot satisfy this requirement, while PKC satisfies this requirement.

- In the case of using a public-key encryption scheme, it doesn't matter, as sensor nodes encrypt any message under the BS's (in our case secure health sensor SHS) public key without requiring the nodes' private key. However, if the BS's or SHS private key is compromised to an eavesdropper, then the attacker can decrypt all past cipher texts encrypted by the public key corresponding to the secret key. Thus, the secret key of the BS must be securely stored to prevent such an exposure.
- In the case of using a public-key signature scheme, even though a user's secret key is compromised to an attacker, the security of communications among non compromised nodes cannot be affected.

A number of applications of WSNs such as Secure Health Sensor require various security attributes and functionalities including authentication with non-repudiation, homomorphic property, aggregation, batch verification, signature with message recovery, etc. PKC makes it possible to achieve these functionalities. PKC is considered to be too computationally expensive for small devices if not accelerated by cryptographic hardware. Recent studies [28], [29], [30], [31] showed that it is feasible to apply PKCs to small wireless devices with very limited resources by choosing public key cryptographic algorithms.

2.1.2 Public-Key Encryption

In public-key encryption (PKE) techniques, all entities say A has pair of keys i.e. public key denoted by e and corresponding secret key denoted by d. The challenge of breaking the PKE is given e, find d, which is computationally hard. The public key e is used to encrypt the information which the private/secret key d is used for decryption of the cipher text to generate the plaintext. If some entity, say B wants to communicate with A, it needs to follow:

A. Obtain authentic copy of A's public key, generally this is accomplished my means of digital certificates.
B. Use A's public key to encrypt the information/message say m to generate the cipher text.

$C = ENC_e(m)$

where C is the generated cipher text corresponding to message m

ENC is the encryption mechanism used.

e is A's public key obtained from digital certificate or trusted third party.

C. A uses its secret key i.e. d to decrypt the cipher text C.

$m = DEC_d(C)$

where m is the original plaintext or message

DEC is the decryption mechanism.

d is A's secret key.

The public key of A need not be kept private, in fact should be distributed to all shareholders communicating with A through trusted third party or by means of digital certificates.

2.1.3 Public Key Signatures

A public key signature (PKS) or named formally digital signature is a dual in concept to PKE technique : it digitally signs a message with the secret key of the sender and then obtained digital signature can be publicly verified with the corresponding secret key. PKSs have a lot of applications in entity authentication, data integrity, information security and non-repudiation. The PKS techniques can be classified in following two categories :

- Signature Schemes with Appendix.
  This signature scheme require the original message for verification purpose.
- Signature Schemes with Message Recovery.
  This signature scheme does not necessitate the original message content for the verification purpose. In this case, the original message is recovered from the signature itself.

Different types of forgeries or attacks on PKS are broadly classified into following three classes:

- Universal Forgery or Total Break.
  An attacker has the ability to either extract the private key used to generate the signature, or to find an efficient signature mechanism that is logically equivalent to the original signature algorithm but provided with the original secret key. So, anyone

can forge signatures of any messages.

- Selective Forgery.
  An adversary is able to create a valid signature for a particular message or a class of messages chosen a priori.
- Existential Forgery.
  An adversary is able to forge a valid signed message that signer has not created, but the adversary has little or no control over which message will be the target.

Types of Attacks are divided into the following three classes:

Key - Only Attack.

An adversary knows publicly available information on the scheme.

Known - Message Attack.

An adversary can get valid signatures for a set of messages which are known to the adversary but not chosen by it.

Chosen - Message Attack.

An adversary can obtain valid signatures from a chosen list of messages before attempting to forge another signed message.

Adaptive Chosen - Message Attack.

An adversary is allowed to use a signer as an oracle: the adversary may request signatures of messages which may depend on the signer's signing key and previously obtained signed messages. That is, at any time, the adversary can query the signer with messages chosen at its will, except for the target message.

2.1.4 PKI

Although PKCs which have some advantages than SKCs are computationally feasible on sensor nodes, one of factors which make it difficult to apply the PKCs to real time health monitoring nodes applications is the public-key authentication problem. PKC has two kinds of keys: a public key and a private key. Public keys must be authenticated, as one can be absolutely sure that a public key belongs to the person. Public-key infrastructure (PKI) is an

arrangement that binds public keys with respective users' identities by means of public-key certificates issued by a Certificate Authority (CA). This PKI causes several problems of certificate management including storage, distribution and the computational cost of certificate verification. According to PKIX which pursued the goal of developing Internet standards to support X.509-based PKIs developed by the ITU-T [32], the major components of a PKI are the following:

- Clients, which are the users of public-key certificates.
- CA, which establishes identities and creates digital certificates.
- Registration Authority(RA), which is responsible for the registration and initial authentication of the clients.
- Repository, which stores the certificates and the Certification Revocation Lists (CRLs).

In order to provide the services of PKI, these components and their functionalities must be mapped to the entities of SHS based wireless sensor networks. In order to deploy PKI into SHS based wireless sensor networks, it is also obligatory to select an appropriate hierarchy model. Fortunately, in most cases, the architecture of sensor networks is extremely simple: one BS that serves as the interface to hundreds or thousands of sensor nodes can communicate with the nodes belonging to the same network. Therefore, it is enough to consider that most sensor networks will use a simple hierarchical PKI architecture, with only one root CA. The basic functionalities of PKI, that is, registration, initialization, key generation, certification, and certification retrieval, are done in SHS based wireless sensor networks as follows:

- The BS creates the public/private key pair of a sensor node, assigns an unique identification to it, and creates a certificate that links that unique identification with its public key. Later, it initializes the contents of the sensor node (such as configuration data and internal programming), including its certificate and the certificate of the root CA (i.e., the BS itself).
- When a sensor node retrieves the certificate of one of its neighbors, it will be able to check its validity using the root CA's certificate.

2.1.5 Identity-Based Cryptography

Identity-based cryptography (IBC) introduced by Shamir [33] allows a user's public key to be easily derived from its known identity information such as an email address or a cellular phone number by eliminating the need for public key certificates. Such cryptosystems alleviate the certificate overhead and solve the problems of PKI technology. A Secret Key Generator (PKG) having a master public/private key pair is responsible for generating secret keys for users. IBC is more suitable for WSNs, as the BS can naturally play the role of the PKG. The BS generates sensor nodes' identities and the corresponding secret keys and then embeds the secret keys in the nodes prior to its use in the field, and no private channel is needed for key setup. Thus, only the identities of the sensors are exchanged without sending public keys and their certificates. This results in energy saving for the communication between sensors. In PKI, each sensor node stores its own public key/private key pair together with the corresponding public key certificate issued by CA. Then, any external party that wishes to interact with nodes also requires the nodes' public key certificates. Although the real-time access to the CA is difficult in WSNs, this pre-installation method of the certificates makes it possible to use the PKI. As mentioned in the previous subsection, this PKI is suitable for node-to-BS communications, but it is not suitable for node-to-node communications, as they require exchange of the nodes' public-key certificates. Thus, ID-based schemes are more suitable for these WSN scenarios: each sensor node which has its unique identification information such as serial numbers gets the corresponding secret keys from the BS which serves as the PKG. To authenticate each other, only the identity information should be exchanged without extra public key data. The length of an identity is much shorter than that of a public key and its certificate. Then the validity of the identity information is determined when its signature related to the identity is verified, i.e., if the signature verification ends successfully then the legitimacy of the identity information is also guaranteed. In particular, IBC makes it possible to establish a session key without any interaction. Two parties, each knowing only the identity of the other and without communicating, are then able to derive a private unknown to any other party, and use that private to compute the same cryptographic key for secure communications. In node-to-BS communications, the BS stores only nodes' IDs instead of their relatively large-size public keys. Therefore, ID-based schemes are more suitable for these WSN scenarios, as it does not require the transmission of the public-key certifications and verifications of CA's signatures on the public keys for node-to-node communications as well as node-to-BS communications.

2.1.6 Signature And Certificate Scheme in SHS

ECC and RSA are mature public-key cryptographic algorithms that have been researched by the academic community for many years: RSA was conceived by Rivest, Shamir and Adleman in 1977 [34] and Koblitz and Miller independently proposed ECC in 1985 [35], [36]. Base Problems and Algorithms for Solving the Problems: The fundamental operation of RSA is a modular exponentiation in integer rings and its security stems from the difficulty of factoring large integers. ECC operates on the groups of points over elliptic curves and derives its security from the hardness of the elliptic curve discrete logarithm problem (ECDLP). While sub-exponential algorithms can solve the integer factorization problem (IFP) and discrete logarithm problem (DLP), only exponential algorithms are known for the ECDLP except those over pairing – friendly curves.

- **Integer Factorization Problem(IFP).**
  Given a composite number n=pq, to find prime factors p or q.

- **Discrete Logarithm Problem (DLP)**.
  Given a group G, a generator g of G, and h = g^x, to compute x, where we denote x = $\log_g h$.

ECC achieves the same level of security with smaller key sizes and higher computational efficiency than RSA: ECC-160 (resp., ECC 224) provides comparable security to RSA-1024 (resp., RSA-2048). Small key sizes offer potential reduction in processing power, memory, bandwidth, and energy. Some factoring algorithms are tailored to perform better when the integer n=pq being factored is of a special form: these are called special-purpose factoring algorithms. The running time of such algorithms depends on certain properties of the factors of n. The special-purpose factoring algorithms include trial division, Pollard's rho algorithm, Pollard's p−1 algorithm, the elliptic curve algorithm, and the special number field sieve[37]. The security of ECC is based on the intractability of the ECDLP, which is an elliptic curve version of the DLP. There are several known algorithms for solving discrete logarithms: generic algorithms and group-specific algorithms. The generic algorithms can be generally applied to any type of cyclic group. The group-specific algorithms are specialized algorithms that make use of the structure in the group elements and apply only within certain families of groups. The generic algorithms include Shank's algorithm, which is also called the Baby-Step Giant-Step algorithm, Pollard's Rho and Pollard's Kangaroo algorithms [10] which are

applied to any cyclic group including elliptic curve groups and subgroups of Zp. These are standard "square-root" methods to compute discrete logarithms in a group of prime order l: if we write the group operation multiplicatively, write g for the standard generator of the group, and write h for the DLP input: the objective is to compute $\log_g h$, i.e., the unique integer x modulo l such that h=gx. "Square-root" means that the algorithms take O(√l) multiplications on average over all group elements.

2.2 Secure Health Sensor Components

A unique design and implementation of a secure sensor node has been carried out based on three major components: a single board computer, an accelerometer based sensor- ADXL345 [12], digital heart beat sensor and a possible and promising GoTrust sd-card based Java Card. Currently, libraries for this java card are not provided by the vendor and thus could not be incorporated in our SHS. Subsidiary components have been used to setup the secure sensor node prototype.

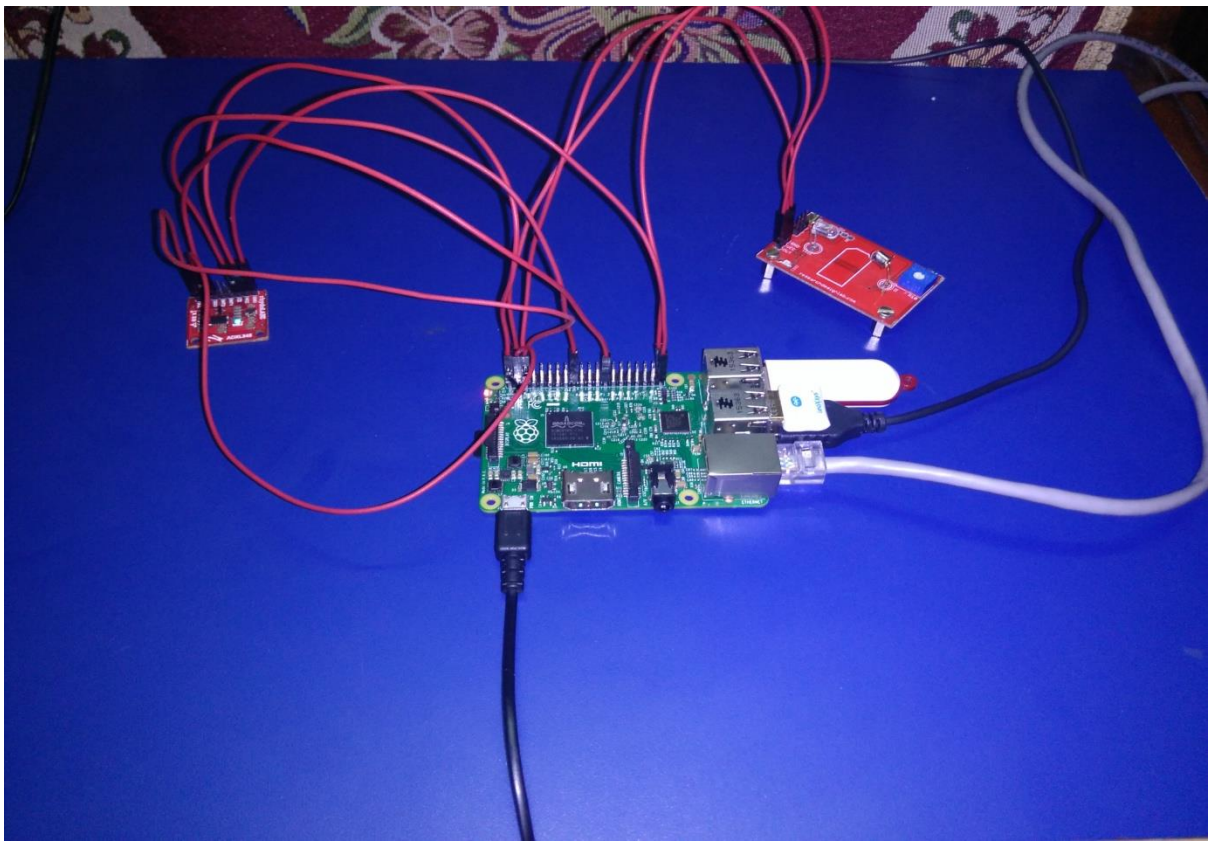


Figure 2.2: Real image of connections of Body sensor

2.2.1 Major Components

2.2.1.1 Single Board Computer

The choice of the single board computer depends on the kind of application which the sensor node uses. Here, we are using it to design a body sensor module which will interface with a number of sensors. Raspberry Pi has been chosen for this application [13]. Among other SBCs, Raspberry Pi is the cheapest single board computer available with the best performance/cost and RAM/cost ratio. Its small size, low cost, low power consumption and high processing power makes it suitable for the design of this body sensor.

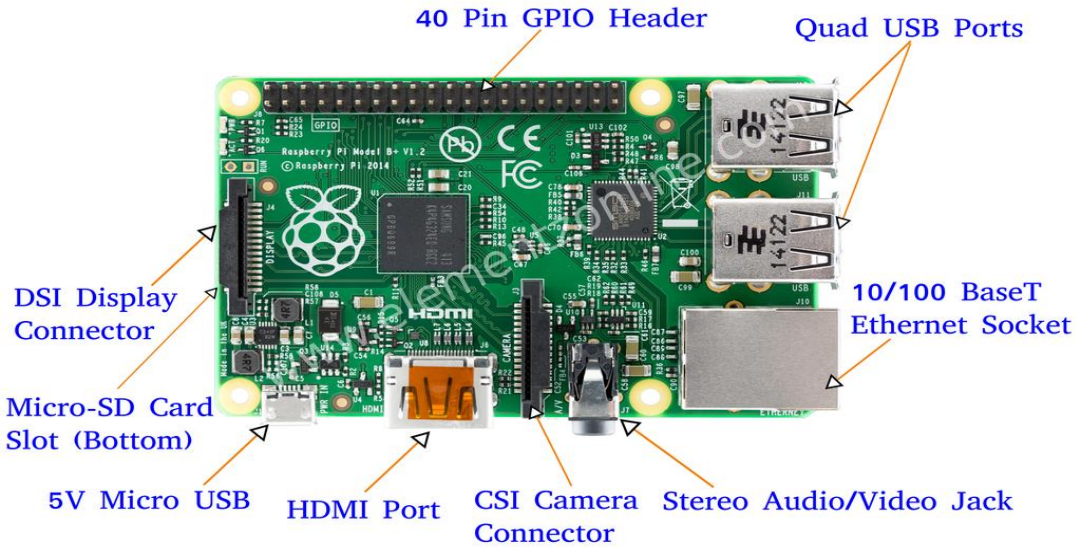


Figure 2.3: Raspberry Pi

Raspberry Pi provides a lot of features at a very low cost. Below are some of the technical specifications of raspberry Pi which makes is versatile among other single board computers.

S.No	Feature	Description
1	Chip	Broadcom BCM2835 SoC
2	Core Architecture	ARM 11
3	CPU	700 MHz Low Power ARM1176JZFS Applications Processor GPU
4	GPU	Dual Core VideoCore IV® Multimedia Co-Processor Provides Open GL ES 2.0, hardware-accelerated OpenVG, and 1080p30 H.264 high-profile decode

		Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure
5	Memory	512MB SDRAM
6	Operating System	Boots from Micro SD card, running a version of the Linux operating system
7	Dimensions	85 x 56 x 17mm
8	Power	Micro USB socket 5V, 2A
9	Ethernet	10/100 BaseT Ethernet socket
10	Video Output	HDMI (rev 1.3 & 1.4) Composite RCA (PAL and NTSC)
11	GPIO Connector	40-pin 2.54 mm (100 mil) expansion header: 2x20 strip Providing 27 GPIO pins as well as +3.3 V, +5 V and GND supply lines
12	Memory Card Slot	SDIO

Table 2.1 : Raspberry Pi Specifications

Some of the advantages and advance features that Raspberry Pi provides are :

- Broadcom BCM2836 Arm7 Quad Core Processor powered Single Board Computer running at 900MHz
- 1GB RAM that can now run bigger and more powerful applications
- Identical board layout and footprint as the Model B+, so all cases and 3rd party add-on boards designed for the Model B+ will be fully compatible.
- Fully HAT compatible
- 40 pin extended GPIO to enhance your "real world" projects. GPIO is 100% compatible with the Model B+ and A+ boards as shown in Figure 2.4. First 26 pins are identical to the Model A and Model B boards to provide full backward compatibility across all boards.
- Connect a Raspberry Pi camera and touch screen display (each sold separately)
- Stream and watch Hi-definition video output at 1080P
- Micro SD slot for storing information and loading your operating systems.
- Advanced power management:

- You can now provide up to 1.2 AMP to the USB port – enabling you to connect more power hungry USB devices directly to the Raspberry PI. (This feature requires a 2Amp micro USB Power Supply)
- 10/100 Ethernet Port to quickly connect the Raspberry Pi to the Internet
- Combined 4-pole jack for connecting your stereo audio out and composite video out

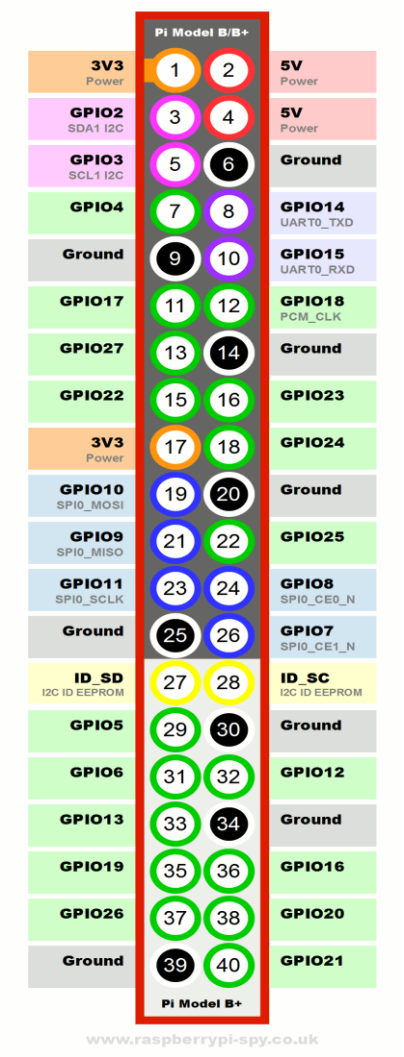


Figure 2.4: Pin Diagram of Raspberry Pi

2.2.1.2 Accelerometer, ADXL345

Accelerometers are devices that measure acceleration, which is the rate of change of the velocity of an object. They measure in meters per second squared (m/s2) or in G-forces (g). A single G-force for us here on planet Earth is equivalent to 9.8 m/s2, but this does vary slightly with elevation (and will be a different value on different planets due to variations in

gravitational pull). Accelerometers are useful for sensing vibrations in systems or for orientation applications. Accelerometers can measure acceleration on one, two, or three axis. 3-axis units are becoming more common as the cost of development for them decreases.

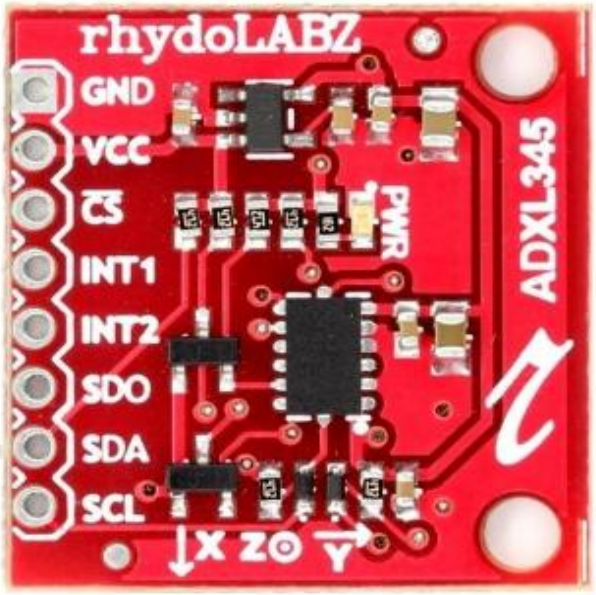


Figure 2.5: ADXL345 Sensor Chip

The ADXL345 [12] is a x, y and z axis accelerometer with a high resolution. It covers a range of ±16 g. The output data present in the data register is formatted as 16bit 2's complement and can be accessed through an SPI (4- or 3-wire) or a I2C digital interface. The ADXL345 is small thin and low-power, hence it is suitable for mobile device applications. It measures static acceleration due to gravity in tilt-sensing applications, and also dynamic acceleration resulting from motion or shock. It has a high resolution (3.9 mg/LSB) which enables measurement of inclination changes of less than 1.0°.

It comes with a 32-level first in, first out (FIFO) buffer which can be used to store data to lower overall system power consumption[12]. This sensor module is chosen since it provides high accuracy and less complex sensor data.

Features:

- 3V-6V DC Supply Voltage
- On board LDO Voltage regulator
- Built in Voltage level convertor (MOSFET based)
- Can be interface with 3V3 or 5V Microcontroller.
- All necessary Components are populated.
- Ultra Low Power: 40uA in measurement mode, 0.1uA in standby@ 2.5V
- Tap/Double Tap Detection

- Free-Fall Detection
- SPI and I2C interfaces

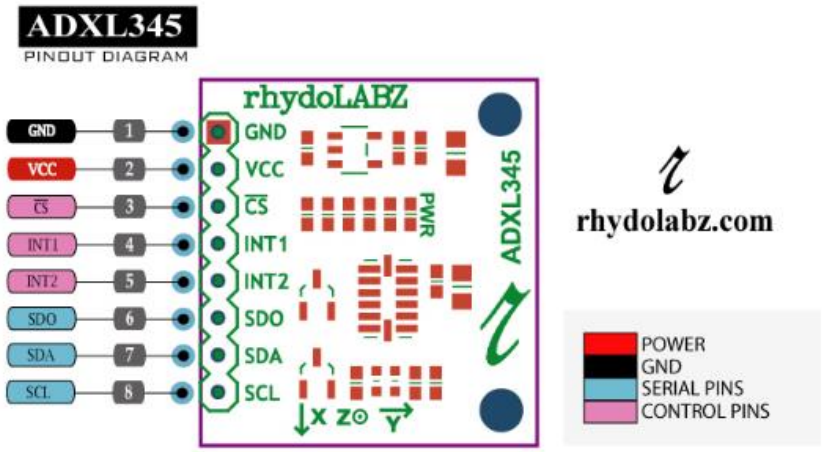


Figure 2.6: ADXL345 Pin Diagram

Connections with Raspberry Pi:

S.No	ADXL345 Module	Raspberry Pi
1	GND	GND
2	VCC	3.3 V
3	SCL0	SCL
4	SDA0	SDA
5	CS	3.3 V
6	SDO	GND

Table 2.2 : ADXL Connections with Raspberry Pi

2.2.1.3 Digital Heart Beat Sensor

The Heart Beat Sensor is designed to provide digital output of heart beat when a finger is placed on it. When the Heart detector starts working, the top most LED will starts flashing with every heart beat. The output of this sensor can be connected to Micro Controller directly to measure the heart beat per minute (BPM) rate. It functions on the principle of light modulation by blood flow through the nerves of the finger at every pulse. The module output mode, Digital output mode is simple, Serial Output is with exact readings.

Features:

- Heart beat indication by LED
- Instant output digital signal for directly connecting to micro controller
- Total heartbeat count can be obtained serially (TTL) every minute.
- Module dual output mode, digital output is simple, serial output with exact readings.
- Compact Size
- Working Voltage +5V DC
- High quality PCB FR4 Grade with FPT Certified.

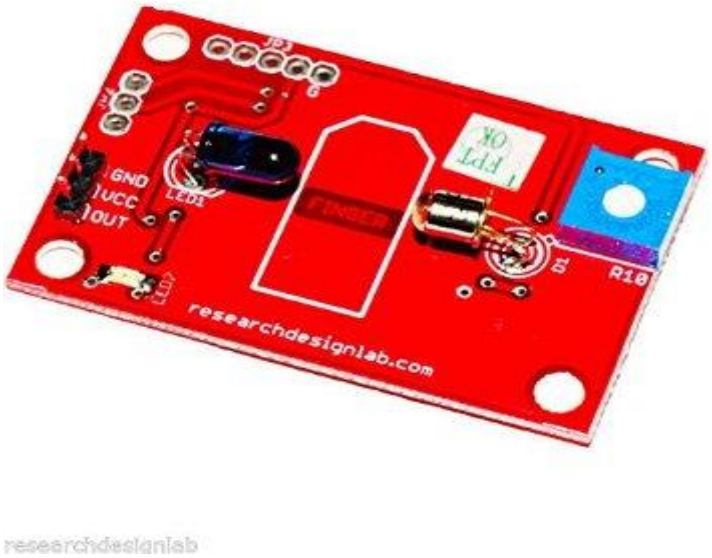


Figure 2.7: Digital Heart Beat Sensor

Connections with Raspberry Pi:

S.No	Heart Beat Sensor	Raspberry Pi
1	Tx	Tx
2	Rx	Rx
3	GND	GND
4	VCC	3.3 V

Table 2.3 Heart Sensor Connections with Raspberry Pi

2.2.1.4 Subsidiary Components

A powered USB hub consisting of four USB ports, a micro USB port (to connect to the USB port of the Pi) and a power jack (to provide a connection to an external power source) is used. It is used during the design of the secure sensor node prototype to connect keyboard and

mouse. A monitor, an HDMI to VGA converter, a power source (Samsung Charger) with output current rating of 700mA-1000mA and output voltage rating of 5V, a USB keyboard, a USB mouse, a PCB board and 8 single pin connectors are other subsidiary components that are used during the design of the secure sensor node prototype.

Note – SHS also used without keyboard and other wired connections providing our model the typical terminal capability. This mode of operation is called headless mode of operation. The system is basically accessed using ssh command from our system to connect to the Raspberry Pi device using terminal window from any Linux based system. More information can be found from [20].

2.2.1.5 Bluetooth Low Energy Dongle

Leoxsys LB4 Bluetooth low energy wireless USB adapter is a plug and play device used to add Bluetooth capability to single board computer such as Raspberry Pi. When plugged in, the micro dongle quickly installs itself and user has Bluetooth capability. This dongle supports dual-mode Bluetooth transmission just by plugging into Raspberry Pi's USB slot. This low energy Bluetooth dongle provide with a 3MB/s data rate for distances of up to 15-20 meters. This Bluetooth dongle can be used in different environments for work, in the office and at home.

Figure 2.8 : Bluetooth LE Dongle

The Leoxsys LB4 has a low profile and rounded edges so leaving it plugged in full time will not be an issue. The dongle is compatible with many windows operating systems such as windows 7, Windows 8, Windows Vista and Windows XP.

- Bluetooth low energy wireless USB adapter
- Wireless local area network share interface
- 15-20 meter sending range

Specifications:

- Bluetooth V4.0 class2 (also compliant Bluetooth 2.1+EDR)
- Operation System: Windows 7, Windows 8, Vista, XP
- Dual-mode Blutooth
- Universal serial USB interface: USB 2.0/3.0
- Transmission speed: 20MB/s
- Enhanced Data Rate (EDR)
- Receiving/sending range: 20m
- Main product dimensions : 25x11x5mm (L x W x D)

2.3 Encryption Algorithm

There is one-to-one communication in the conventional cryptography, which means, owner encrypt the message by receiver's public key and that message can only be decrypt by the receiver, because only the receiver will have the corresponding secret key. In identity-based encryption, public key can be any identity, such as social security number of a person. But in real life situation, there are many application of one-to-many communication, therefore there is a need of broadcast encryption. For example, in a shared environment many user can access the same documents (files), if anyone wants to send any documents to more number of persons and if he will encrypt the information by using the individual's public key then surely it will be secure but increase the computation overhead. It will also increase the communication overhead because for many number of users, encryptor will store the encrypted data (same information) on the cloud again and again. Thus to avoid these overheads one prefer the efficient broadcast encryption.

Popularity of cloud storage is growing day by day, nowadays most of the organizations and enterprises prefers cloud storage to store and distribute the information. Protecting the information in cloud computing is very important, because it is out of the user's control, after being stored on the cloud and the cloud service provider may not be reliable. There are many real life examples of distributed system in which accessing of information are defined by some rules and these rules can lead to a complex access structure defined on the set of

attributes of users. Information is encrypted under this complex structure and a user can access the information if he/she satisfies the access structure.

In attribute-based encryption (ABE) scheme, without having the exact knowledge of the receivers set, an owner can encrypt the information that can be decoded only by those users, who are eligible to decrypt it. Attribute based encryption applies the access structures, defined on the set of attributes, during the encryption mechanism. Fuzzy IBE scheme is introduced by Sahai and Waters [2], an application where both encoded message and secret keys are associated with set of attributes. In which if there is an overlap of at least (t) attribute between the attributes associated with ciphertext and attributes associated with secret keys of a user then only user should be eligible to decipher the ciphertext.

There are two kinds of ABE schemes. Key-policy attribute based encryption (KP-ABE) [14] and Ciphertext - policy attribute based encryption (CP-ABE) [15]. This categorization of ABE scheme depends on the association of access structure, whether it is associated with ciphertext or private keys of a user. If the access structure is associated with the private keys of a user, then it is a KP-ABE scheme, and a user can decrypt the encrypted message only if the attributes associated with ciphertext satisfy the access structure. In CP-ABE scheme, ciphertext is labeled with access structure and user's private key is labeled with attributes, if attributes associated with user's private key fulfill the requirement of the access structure then only the user can decipher the ciphertext.

There are six main properties of a model ABE scheme, these properties are discussed as follows.

(1) Data confidentiality- malicious user cannot know that, what was the actual information before encryption? (2) Access control- encryptor defines some rule before encrypting the information, is called access structure. Access structure defines who can access the encrypted information and who cannot? (3) Scalability- if the number of users increases and the users are authorized, then it does not affect the overall performance of the system. (4) Attribute/User revocation-if any user leaves the system then all the access rights of that particular user is revoked by the scheme, means that user cannot access the information in future. Sometimes attributes revocation is also possible. (5) Collusion resistance-there might also be a situation, where two or more no. of users can combine their private keys and try to decrypt the encrypted information, it must be avoided by an ideal ABE scheme. (6)

Accountability- an ideal ABE scheme should also be accountable because key abuse can be prohibited by the accountability.

Sometimes access structures may contain important information about the users, who are encoding or decoding the information, or about the data being encrypted. Cheung and Newport [17] proposed a CP-ABE scheme and this scheme supports do not care element for the attributes, which are not present in the AND gate access structure of the ciphertext and also deals with the negative attributes but the policies need to be publicized. Kapadia et al. [19] also presented a CP-ABE scheme that have the same flexibility and realizes hidden access structures, but this scheme does not prevent the collusion of user private keys, means more than one user can collude and combine their private1s key and be able to decrypt the ciphertext, provided that individually, no one is able to decrypt the ciphertext, and also requires an online semi-trusted authority. This type of authority can compromise with the effectiveness of the overall system.

Katz et al. [21] proposed a predicate encryption scheme that supports inner product encryption. This scheme can realize both key-policy ABE and ciphertext-policy ABE. In predicate encryption scheme, ciphertexts are labeled with attributes and private1e keys corresponds to predicates. Private keys would be able to decrypt the ciphertexts if and only if attributes associated with ciphertexts satisfy the corresponding predicates.

Nishide et al. [18] presented the idea of hiding the access structure associated with the ciphertext and gave two constructions. The permissible structures are AND gate based structures. In [18] they have considered the access structures of attributes connected with the AND gates. In this paper they also consider the do not care condition for some attributes in the access structure, means if, some attributes are not present in the access structures then users need not to have private keys for those attributes. In [18], only the authorized user can decrypt the encrypted data and not authorized users cannot decrypt the encrypted data or even know something about the attributes of the access structures, means malicious users cannot get any information about the encryptor and decryptor of the encrypted data and also he cannot guess about the actual information.

2.3.1 Attribute Based Encryption - ABE

Attribute-based encryption (ABE) is a relatively new asymmetric key cryptography technique in which the secret key of the user and cipher text are dependent on the attribtues. Generally,

in asymmetric key cryptography, data is enciphered for a specific receiver using the receiver's public-key. ABE on the other hand, defines the identity of users not as atomic but as a set of attributes, e.g., age, data of birth etc., and messages can be enciphered with a set of attributes (key-policy ABE - KP-ABE) or policies defined over a set of attributes (ciphertext-policy ABE - CP-ABE).

2.3.1.1 Key-Policy ABE

KP-ABE was introduced by Goyal et al. [14], for sharing the information among the multiple users. In KP-ABE scheme owners encrypt the information and labeled it with a set of user's attributes and secret-key of users is labeled with the access structure. Secret-key are given in advance to the users according to their corresponding attributes from the trusted authority. To enable the decryption, set of attributes must satisfy the access structure.Key-policy attribute-based encryption (KP-ABE) is an important type of ABE, which enables senders to encrypt messages under a set of attributes and secret keys are associated with access structures that specify which ciphertexts the key holder will be allowed to decrypt. In most existing KP-ABE scheme, the ciphertext size grows linearly with the number of attributes embedded in ciphertext. KP-ABE is the dual to CP-ABE in the sense that an access policy is encoded into the users private key, e.g., (A and C) or D, and a ciphertext is computed with respect to a set of attributes, e.g., {A,B}. In this example the user would not be able to decrypt the ciphertext but would for instance be able to decrypt a ciphertext with respect to {A,C}.

2.3.1.2 Ciphertext-Policy ABE

CP-ABE was first introduced by Sahai and Waters [22]. In CP-ABE cipher text is labeled with access structure and attributes are labeled with the user's secret-keys, a user would be able to decode the encoded information only if the set of attributes associated with secret-key of user satisfies the access structure. In CP-ABE access structures needs to be public along with the cipher rtext, because decryptor should know, how private key components will be combined to cipher text component to decrypt the encrypted data.

In CP-ABE model [22], secret keys will be identified with a set S of descriptive attributes. A party that wishes to encrypt a message will specify a policy (access tree) that secret keys must satisfy in order to decrypt. Access tree T. Each non-leaf node of the tree represents a threshold gate, described by its children and a threshold value. If num_x is the number of

children of a node x and k_x is its threshold value, then $0 < k_x \leq num_x$. Each leaf node x of the tree is described by an attribute and a threshold value $k_x = 1$.

2.3.1.2.1 Satisfying an access tree.

Let T be an access tree with root r. Denote by T_x the sub-tree of T rooted at the node x. Hence T is the same as T_r. If a set of attributes γ satisfies the access tree T_x, we denote it as $T_x(\gamma)=1$. We compute $T_x(\gamma)$ recursively as follows: if x is a non-leaf node, evaluate $T_x(\gamma)$ for all children x of node x. $T_x(\gamma)$ returns 1 if and only if at least kx children return 1. If x is a leaf node, then $T_x(\gamma)$ returns 1 if and only if $att(x) \in \gamma$. Figure 2 shows the access tree structure with 4 attributes with the policy as given below :

Doctor and (Nurse or (Cardiology and Neurology))

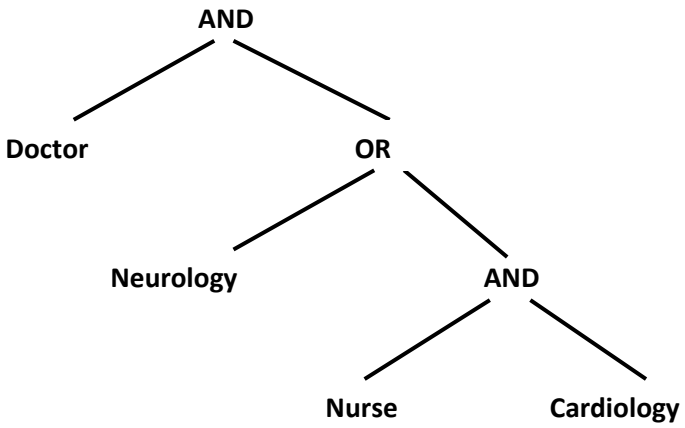


Figure 2.9: Example of Access tree structure with 4 attributes

2.3.1.2.2 CPABE toolkit

The cpabe toolkit provides a set of programs implementing a ciphertext-policy attribute-based encryption scheme. It uses PBC (Pairing Based Cryptography) library for the algebraic operations. The cpabe toolkit might not compile against versions of PBC older than 0.5.4. The code is split into two packages, libbswabe (a library implementing the core crypto operations) and cpabe (higher level functions and user interface). We need to install the libbswabe library first since cpabe has dependency on it.

In a cipher text policy attribute-based encryption scheme, each user's secret key is associated with a set of attributes representing their capabilities, and a cipher text is encrypted such that

only users whose attributes satisfy a certain policy can decrypt. For example, we can encrypt a cipher text such that in a health monitoring system such as SHS, cipher text related to patient can be deciphered only by person having roles "Doctor", "Nurse", "Health Professional" or "Caretaker" of the patient. One interesting application of this tool is that we can do Role-Based Access Control (RBAC) without requiring trusted data storage. The toolkit provides four command line tools used to perform the various operations of the scheme.

Below are the different steps used in cpabe implementation using cpabe toolkit [16]:

1. Setup

It produces the public parameters public key (pub_key) and master key (master_key) as outputs using the cpabe-setup command.

2. Key-Generation (master_key,pub_key, S)

This step takes three inputs, master private key (master_key), public key (pub_key) and set of attributes of a user S. It uses cpabe-keygen command provided by cpabe-toolkit[16]. It outputs the secret key (SK) for the user according to the set of attributes.

3. Encrypt (pub_key, M ,w)

This step uses the public key and policy access structure(w) defined by a set of attributes to encrypt the message (M). The cpabe-enc command is used to encrypt the message M. The output of this step is cipher text with .cpabe extension to input message file. For example, if the input file is directions.txt then the output file will be directions.txt.cpabe.

4. Decrypt (pub_key,C, SK)

The Decryption step takes three inputs, public parameter (pub_key), a ciphertext (C) containing embedded access policy(w), and private key(SK). If the attribute set on which private key is defined, satisfies the access structure, the algorithm will decipher the ciphertext and generate the plaintext, otherwise it returns error message that user attribute set does not satisfy the policy.

2.3.1.2.3 CPABE Toolkit Commands

2.3.1.2.3.1 Library Installation

First download, untar, compile, and install the most recent tarball of libbswabe, the support library. Each can be can be installed with the standard GNU build system commands.

1. $ ./configure
2. $ make
3. $ make install

The "$" denotes your shell's prompt.

2.3.1.2.3.2 Set-Up

To generate the public key and master keys, user need to run cpabe-setup command.

```
$ cpabe-setup

$ ls
master_key  pub_key
```

The master_key file is used to produce secret keys associated with various sets of attributes. To generate the private keys for different stakeholders in health monitoring system such as SHS to provide role based access mechanism, we need to use cpabe-keygen.

2.3.1.2.3.3 cpabe-keygen command

```
$ cpabe-keygen -o user2_priv_key pub_key master_key \
   doctor hospital_employee health_professional

$ cpabe-keygen -o user1_priv_key pub_key master_key \
   health_professional hospital_staff health_professional

$ ls
master_key  pub_key  user2_priv_key  user1_priv_key
```

Some attributes are assigned a value, while others a key simply "has" without further qualification. If later someone wants to encrypt a sensitive document. All is needed is the public key, then can use cpabe-enc to encrypt it under a specified policy.

2.3.1.2.3.4 cpabe-enc Command

```
$ ls
pub_key  patient_record.pdf

$ cpabe-enc pub_key patient_record.pdf
   (doctor and hospital_employee) or
   (hospital_staff and health_professional)
```

```
^D
$ ls
pub_key  patient_record.pdf.cpabe
```

In this case, they typed the policy on stdin. Note that the attributes of User's 1 key satisfy this policy, but the attributes of User2's key do not.
If User1 wants to decrypt the document, he can use cpabe-dec

2.3.1.2.3.5 cpabe-dec

```
$ ls
pub_key user1_priv_key patient_record.pdf.cpabe

$ cpabe-dec pub_key user1_priv_key patient_record.pdf.cpabe

$ ls
pub_key  user1_priv_key  patient_record.pdf
```

If User2 were to try to decrypt it, an error would be reported.

2.4 Public Key Infrastructures and Digital Certificates for the Internet of Things

Peer-to-peer (P2P) network communication can be compromised via the principal attack mechanisms of interception (e.g. eavesdropping), interruption (i.e. DoS attacks), modification (i.e. packet payload manipulation in transit) and fabrication (see Figure 1). In an IoT environment communications take place between autonomous embedded devices (i.e. sensors and actuators) or IoT devices and their (cloud) backend. The above attack mechanisms enforce the need for a set of IoT-specific requirements for machine-to-machine (M2M) communication as shown in the example in Figure 2: Confidentiality and integrity are concerned with the data itself that is being transmitted between peers. Both provide a foundation (complementary to additional protocol features like protocol sequence numbers or timestamps) to deal with interception, interruption and modification. Furthermore, authentication and authorisation provide assurance that (i) a peer is an entity it claims to be and (ii) a peer is authorized to conduct a certain action, i.e. a smart meter backend server being allowed to reset a smart meter. Therefore it is a viable mechanism to protect against fabrication. The above data-centered requirements are dealt with by modern P2P communication protocols like TLS on transport layer, IEEE 802.15.4 / MACSec on data-link layer or IPSec on network layer. An additional piece of information these protocols require is a shared private token that is used to provide (i) confidentiality and (ii) integrity, i.e. a shared

128 bit AES key to encrypt and decrypt either (i) packet payloads or (ii) payload hash values.

2.4.1 Public Key Infrastructures

Public Key Infrastructure is a centralized solution to the problem of trust. The idea is to have a trusted entity (organization, corporation) that will do the job of certifying that a given public key belongs really to a given person. This person must be identified by his name, address and other useful information that may allow to know who this person is. Once this work his done, the PKI emits a public certificate for this person. This certificate contains between others:

- All the information needed to identify this person (name, birth date, ...).
- The public key of this person.
- The date of creation of the certificate.
- The date of revocation of the certificate (a certificate is valid during 1 or 3 years in practice).
- The digital signature of all this previous information emitted by the PKI.

So now, if a user want to send a secret message to Bob, he/she can ask for his certificate. When a user received the certificate, he must check the signature of the PKI who emitted it and for the date of revocation. If verifications pass then user can safely use the public key of the certificate to communicate with Bob. Indeed, in practice the way a PKI works is much more complicated. For example sometimes a certificate may be revocated before the date of end of validity has been reached. So a kind of list of revocated certificated has to be maintained and accessed every time you want to use a certificate. The problem of certificate revocation is really difficult in practice.

A PKI provides indirectly a mechanism to provide such a shared private token between two peers. Its principal task is to provide and manage digital certificates (also called identity certificates) that bind a public key to a peer (or end-entity) identity in such a way that a 3rd party can validate this binding. The required steps to issue a digital certificate can be seen in Figure 3: An end-entity (EE) sends a certificate request (containing identity descriptors and a public key) to a registration authority (RA), which validates the request (i.e. the end-entity details) before sending the request to the CA for signing. The CA is in possession of a public

/ secret master key pair (i.e. a RSA or ECC key pair). It generates a certificate based on the parameters passed, calculates a hash value over it and signs it using its secret key. The CA then returns the created digital certificate back to the end entity. The public key of the CA is known to all parties (e.g. via a self-signed certificate issued by the CA to itself), so the end-entity's certificate can be independently validated by a 3rd party to which it wants to connects to. The 3rd party can also request the status of the certificate (e.g. valid or revoked) by querying a validation authority (VA) that keeps track on all issued certificates.

PKI provides the core framework for a wide variety of components, applications, policies and practices to combine and achieve the three principal security functions (integrity, authentication and non- repudiation). A PKI is a combination of hardware and software products, policies and procedures. It provides the basic security required for secure communications so that users who do not know each other or are widely distributed, can communicate securely through a chain of trust. Digital certificates are a vital component in the PKI infrastructure as they act as 'digital passports' by binding the user's digital signature to their public key.

2.4.1.1 Components of a PKI

- Security policy
- Certificate Authority (CA)
- Registration Authority (RA)
- PKI-enabled applications

Today's secure internet communication is provided by 3 principal components: (i) Network protocols (on data link, network or application layer) that provide secure and authenticated peer-to-peer communication, (ii) a verifiable digital identity for each peer and (iii) an infrastructure that allows for the generation, management and revocation of the latter. The de-facto standards for (i) are on application and network layer the TLS and the IPSec protocol respectively, while MACSec provides a similar service on data link layer. Components (ii) and (iii) are provided by digital certificates and public key infrastructures respectively as defined by X.509.

2.4.1.2 Non – Repudiation

Non-repudiation guarantees that a party cannot deny having received/sent the message. Even,

if it is not the most used security property, it can come in handy for scenarios involving trust during sensitive exchanges. Different types of non repudiation have been proposed, depending on who (sender or recipient) is applying the non-repudiation mechanism. From the sender point of view, one would be willing to be sure that her/his message was received by the recipient (non-repudiation of receipt (NRR)) or her/his message was well sent to the recipient (non-repudiation of submission (NRS)) or her/his message has been delivered to the recipient (non-repudiation of delivery (NRD)). From the recipient point of view, one would be willing to be sure that the message she/he received has been sent by a genuine sender (non-repudiation of origin (NRO)). NRR is quite simple to implement because it only needs a nuncio (i.e. a particular document that attests the validity of the transaction). The nuncio is generated by the sender and transmitted to the recipient. In order to terminate the transaction, the recipient needs to send the nuncio back.

CHAPTER 3

PROPOSED WORK

3.1. Problem Statement

In multi user environment providing role based access. The objective is to design and implement a system that satisfies the following constraints.

1. The system should be energy efficient.
2. The system should provide APIs to safely record the patient readings.
3. The system should be able to store the readings in a secure storage.
4. Data should be encrypted before transmission.
5. The system should allow users with necessary attributes to decrypt the sensitive medical data in multi-user environments.
6. The system should provide mechanism to do the auditing to provide the non-repudiation property.
7. The system should provide way for cipher text transmission using Bluetooth transmission or web interface.

3.2. Proposed Solution

- Wired connections provided between sensors and the processor to remove the loss of sensor readings.
- CP-ABE provides the role based access mechanism.
- Symmetric AES key is used to encrypt the medical data

- Use of Bluetooth low energy dongle make system efficient.
- CP-ABE has been found as one of the recent encryption scheme to provide role based access mechanism.
- Web Interface provided to gather and store the readings in database.

3.3. System Design

In our system SHS, the medical data is gathered at patient site through the bio-medical sensor and is sent over the air through Bluetooth medium to remote site. The data is generally collected by medical professionals who are also responsible for creating the access policy for the doctors, nurses etc. The transmitted data is accessible to anyone having the required set of attributes; thus making it possible for multiple shareholders to access this sensitive medical data such as doctors, relatives of the patient etc. Thus, SHS is very flexible and provide fine grained access control over the medical data.

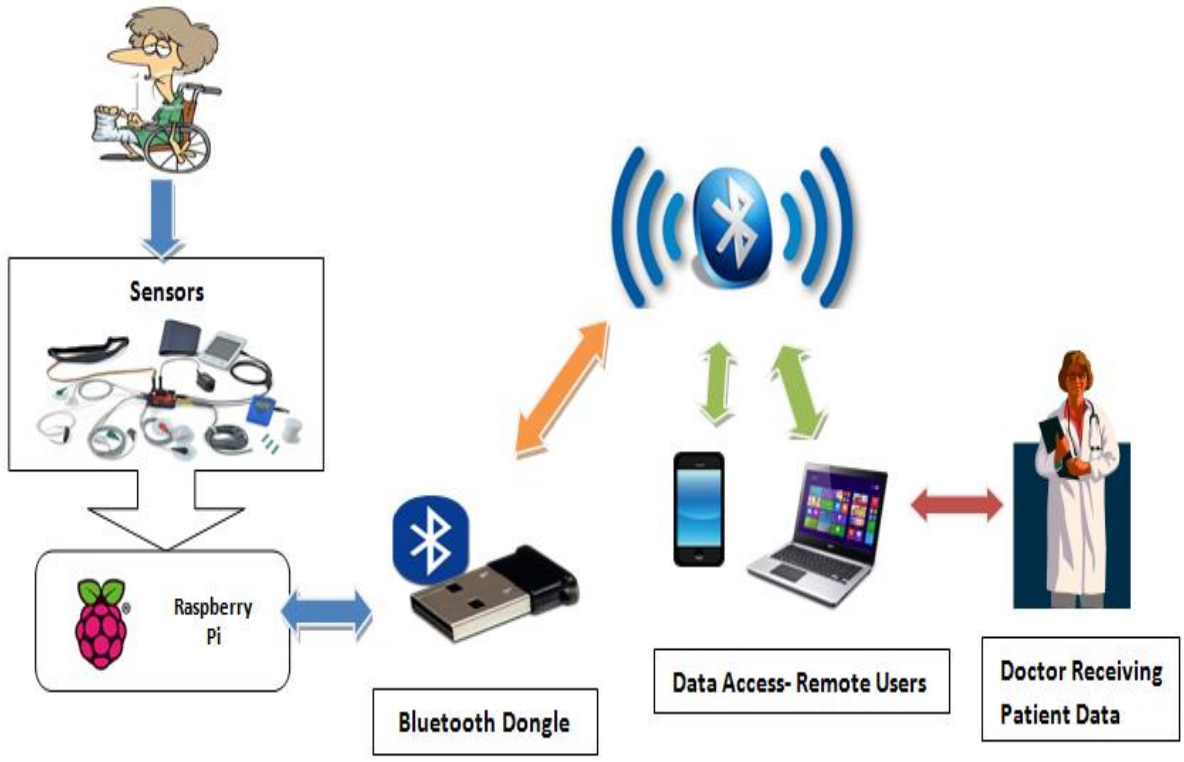


Figure 3.1: SHS System Architecture

Figure 3.1 shows the architecture as well as the flow of medical data starting from sensors and till finally it is collected at remote site by user. The bio-medical sensors are attached to patient to take the readings. The readings are then securely transmitted through the wires to the raspberry pi device. Raspberry pi performs the encryption using the CP-ABE to provide the fine grained access control. This encrypted data is then transmitted through the Bluetooth device over the air to the remote user on demand. At remote site, the doctors, nurses, caretakers of the patient etc will be able to see this sensitive medical data. Any user without having valid set of attributes will not be able to decrypt the file.

3.4. Implementation

3.4.1 Setting up Raspberry Pi / Personalization

We used Raspberry Pi SBC to store following components on it:

Raspbion Operating System – Jessie Lite.

- Latest stable version of Debian based Wheezy.
- Debian Jessie Lite, provides the GUI which makes it easier to interface with Raspberry Pi.
- Updated version GTK 3+ for user interface.
- Ease of use and wide developer support.

3.4.2 Connections of the Secure Sensor Node

The accelerometer based sensor consists of eight pins, two of which are the power and ground pins. Two Interrupt pins are available but are left unconnected. The CS' and VDD pins are supplied with 3.3 V from the Raspberry Pi. The SDO and the GND pin of the accelerometer based sensor are connected to the GND pin of the Raspberry Pi. The SDA pin for data interchange is connected to the third pin of the Raspberry Pi and the SCLK pin to the fifth pin of the Raspberry Pi. Communication between the accelerometer based sensor and the Raspberry Pi can take place using either of the two serial protocols: SPI [8] or I2C [8].

The accelerometer based sensor (ADXL345) supports standard (100 kHz) and fast (400 kHz) data transfer modes if the given timing parameters are met.

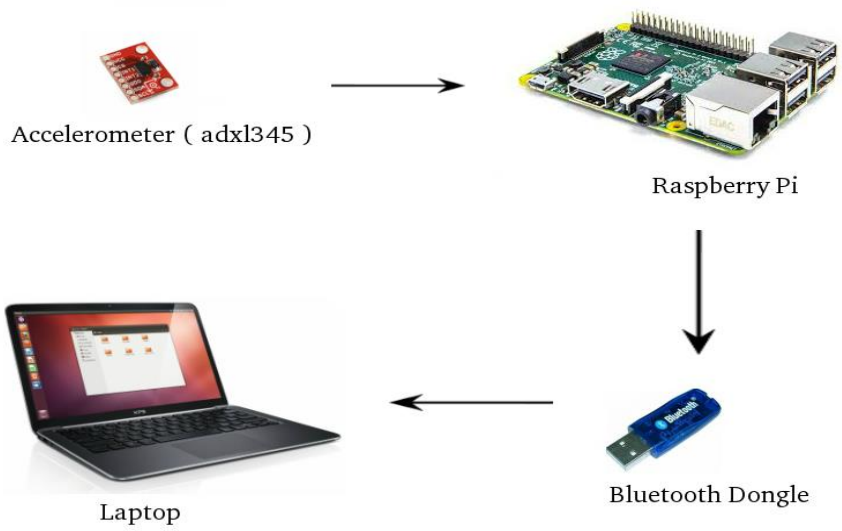


Figure 3.2: Body Sensor Flow Diagram

I2C address of 0x53 (followed by the R/W bit) is chosen by grounding the SDO/ALT ADDRESS pin. If other devices are connected to the same I2C bus, these devices cannot have an operating voltage exceeding VDD I/O by more than 0.3 V [8]. Therefore external pull-up resistors, Rp are necessary for proper I2C operation. The I2C driver [13] is enabled and the I2C modules [13] and the python-smbus module [13] are downloaded and installed. Figure. 1 shows how the Raspberry Pi is connected to the accelerometer based sensor in this project. The Program Flow of the secure sensor prototype is shown in Figure. 6.

3.4.3 Receiving Data from the Accelerometer Based Sensor

The accelerometer based sensor is set to measurement mode. The range of measurement of the accelerometer based sensor is set according to the user. After this, the 16 bit Two's Complement data of each axis is retrieved from the registers of the accelerometer based sensor, ADXL345. The Raspbian Jessie comes with a pre-installed version of Python 2.7 where the program is written. The secure sensor node prototype program aims to achieve the following objectives:

- Retrieving sensor data from the data registers of the sensor and processing it to give acceleration in m/s2.
- Encryption of this processed data.
- Sending this data to the phone using Bluetooth.

The following modules have been imported in the python program to assist in the above processes:

- The 'SMbus' module, known as python-smbus: it is a Python module which allows SMbus access through the I2C/dev interface on Linux hosts.
- The os and the sub-process module: they are used to execute command line instructions in python.

3.4.4 Encryption

The encryption is a two step process. Before starting the encryption, the secure node requests the secure symmetric key from the java card applet stored on secure pi node. This is required since we assume that the secure node will never store the symmetric key in plain text anywhere in its memory. We store the symmetric key on card to increase the security of the system. Once we get the symmetric key, we can start the encryption mechanism. We used Cpabe toolkit to perform the encryption of the symmetric key so that it can be transferred over the air to the mobile user so that only authenticated users can determine the sensor data. We have used encrypted Bluetooth transfer from the raspberry pi to the smart phone. This section provides a brief description of the encryption algorithm used in the program. The first step in the encryption step is encrypting the sensor data file using the symmetric AES key. We used symmetric key K_{AES}(128-bit) AES key for our program, though to increase the strength of the encryption we can use 24 byte (192-bit) or 32 byte (256-bit) AES key. The results in generation of cipher text, say C1. Now, to secure this symmetric key we again encrypt it using our cpabe setup. This results in cipher text, say C2. The secure sensor node sends this combined cipher text (C1 + C2) over the air through the Bluetooth dongle to the mobile user.

Bluetooth is used in the secure sensor node to perform below major tasks:

- Sending the file, 'sensor_data.txt' containing the encrypted data to a mobile phone
- Sending the file, 'AES_key.cpabe' containing the encrypted symmetric key.

Note, mobile phone user MAC address is hard coded in the program. At the receiver side, the sequence of operations will be in reverse order.

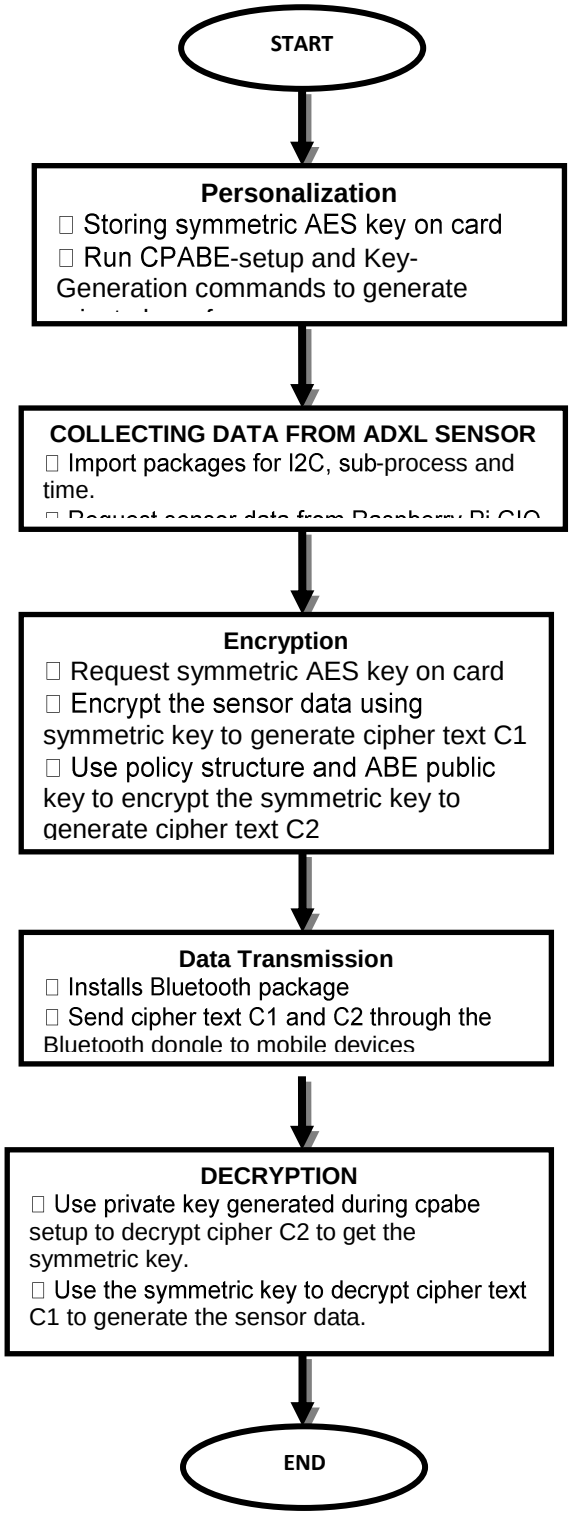


Figure 3.3: Program Flow of SHS

3.4.5 Openssl

OpenSSL is a versatile command line tool that can be used for a large variety of tasks related to Public Key Infrastructure (PKI) and HTTPS (HTTP over TLS). This cheat sheet style guide provides a quick reference to OpenSSL commands that are useful in common, everyday scenarios. This includes OpenSSL examples of generating secret keys, certificate signing requests, and certificate format conversion. It does not cover all of the uses of OpenSSL. To find list of all possible commands supported in a specific version of OpenSSL the following command can be used:

openssl list-standard-commands

Let's see a brief description of each command:

- ca To create certificate authorities.
- dgst To compute hash functions.
- enc To encrypt/decrypt using private key algorithms. It is possible to generate using a password or directly a private key stored in a file.
- genrsa This command permits to generate a pair of public/secret key for the RSA algorithm.
- password Generation of "hashed passwords".
- pkcs12 Tools to manage information according to the PKCS #12 standard.
- pkcs7 Tools to manage information according to the PKCS #7 standard.
- rand Generation of pseudo-random bit strings.
- rsa RSA data management.
- rsautl To encrypt/decrypt or sign/verify signature with RSA.
- verify Checkings for X509.
- x509 Data managing for X509.

To perform encryption, below command can be used:

openssl rsautl -encrypt -in <input_file> -inkey <llave> -out <output_file>

where:

- input_file is the file to encrypt. This file must no be longer that 116 bytes =928 bits because RSA is a block cipher, and this command is low level command, i.e. it does not do the work of cutting your text in piece of 1024 bits (less indeed because a few bits are used for special purposes.)
- key File that contains the public key. If this file contains only the public key (not both secret and public), then the option -pubinmust be used.
- output_file the encrypted file.

To decrypt only replace -encrypt by -decrypt, and invert the input / output file as for decryption the input is the encrypted text, and the output the plain text.

3.4.5.1 Signature Generation using OpenSSL

The next step is to be create a digital signature and to verify it. It is not very efficient to sign a big file using directly a public key algorithm. That is why first we compute the digest of the information to sign. Note that in practice things are a bit more complex. The security provided by this scheme (hashing and then signing directly using RSA) is not the same (is less in fact) than signing directly the whole document with the RSA algorithm.

A). Openssl Command to create signatures :

openssl dgst -<hash_algorithm> -out <digest> <input_file>

where:

- hash_algorithm is the hash algorithm used to compute the digest. Among the available algorithm there are: SHA-1 (option -sha1which computes a 160 bits digests), MD5(option -md5) with 128 bits output length and RIPEMD160 (option -ripemd160) with 160 bits output length.
- digest is the file that contains the result of the hash application on input_file.
- input_file file that contains the data to be hashed.

This command can be used to check the hash values of some archive files like the openssl source code for example.

B). To compute the signature of the digest:

openssl rsautl -sign -in <digest> -out <signature> -inkey <key>

C). To check to validity of a given signature:

openssl rsautl -verify -in <signature> -out <digest> -inkey <key> -pubin

–pubin is public key, which is natural as we are verifying a signature.

D) To complete the verification, one needs to compute the digest of the input file and to compare it to the digest obtained in the verification of the digital signature.

3.4.5.2 Certificate Generation

SSL makes use of what is known as asymmetric cryptography, commonly referred to as public key cryptography (PKI). With public key cryptography, two keys are created, one public, one secret. Anything encrypted with either key can only be decrypted with its corresponding key. Thus if a message or data stream were encrypted with the server's secret key, it can be decrypted only using its corresponding public key, ensuring that the data only could have come from the server.

A certificate is not really necessary -the data is secure and cannot easily be decrypted by a third party. However, certificates do serve a crucial role in the communication process. The certificate, signed by a trusted Certificate Authority (CA), ensures that the certificate holder is really who he claims to be. Without a trusted signed certificate, the data may be encrypted, however, the party you are communicating with may not be whom sender think. Without certificates, impersonation attacks would be much more common.

Steps In Public Certificate Generation:

a). Generate Secret Key:

The openssl toolkit is used to generate an RSA Secret Key and CSR (Certificate Signing Request). It can also be used to generate self-signed certificates which can be used for testing purposes or internal usage.

The first step is to create your RSA Secret Key. This key is a 1024 bit RSA key which is encrypted using Triple-DES and stored in a PEM format so that it is readable as ASCII text.

openssl genrsa -des3 -out server.key 1024

Output:

Generating RSA secret key, 1024 bit long modulus

.....++++++

.....++++++

e is 65537 (0x10001)

Enter PEM pass phrase:

Verifying password - Enter PEM pass phrase:

Where pass phrase is used as a private whenever the user wants to use his secret key in certificate generation process.

b). Generate a Certificate Signing Request.

Once the secret key is generated a Certificate Signing Request can be generated. The CSR is then used in one of two ways. Ideally, the CSR will be sent to a Certificate Authority, such as Thawte or Verisign who will verify the identity of the requestor and issue a signed certificate. The second option is to self-sign the CSR.

During the generation of the CSR, user will be prompted for several pieces of information. These are the X.509 attributes of the certificate. One of the prompts will be for "Common Name (e.g., Certificate User name)". It is important that this field be filled in with the fully qualified domain name of the server to be protected by SSL. If the website to be protected will be https://public.akadia.com, then enter public.akadia.com at this prompt. The command to generate the CSR is as follows:

openssl req -new -key server.key -out server.csr

Output:
Country Name (2 letter code) [GB]:IN
State or Province Name (full name) [Berkshire]:DELHI

Locality Name (eg, city) [Newbury]:ROHINI
Organization Name (eg, company) [My Company Ltd]:DELHI TECHNOLOGICAL UNIVERSITY
Organizational Unit Name (eg, section) []:Information Technology
Common Name (eg, your name or your server's hostname) []:dtu.org
Email Address []:surajrider@gmail.com
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []: Delhi Technological University

c). Remove passphrase from key:

One unfortunate side-effect of the pass-phrased secret key is that Web Server such as Apache will ask for the pass-phrase each time the web server is started. Obviously this is not necessarily convenient as someone will not always be around to type in the pass-phrase, such as after a reboot or crash. mod_ssl includes the ability to use an external program in place of the built-in pass-phrase dialog, however, this is not necessarily the most secure option either. It is possible to remove the Triple-DES encryption from the key, thereby no longer needing to type in a pass-phrase. If the secret key is no longer encrypted, it is critical that this file only be readable by the root user! If system is ever compromised and a third party obtains your unencrypted secret key, the corresponding certificate will need to be revoked. The following command can be used to remove the pass-phrase from the key:

cp server.key server.key.org
openssl rsa -in server.key.org -out server.key

The newly created server.key file has no more passphrase in it.

d). Generate a Self Signed Certificate

At this point you will need to generate a self-signed certificate because you either don't plan on having your certificate signed by a CA, or you wish to test your new SSL implementation while the CA is signing your certificate. This temporary certificate will generate an error in the client browser to the effect that the signing certificate authority is unknown and not trusted.

To generate a temporary certificate which is good for 365 days, issue the following command:

openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt

Output:
Signature ok
subject=/C=CH/ST=Bern/L=Oberdiessbach/O=Akadia AG/OU=Information Technology/CN=public.akadia.com/Email=martin dot zahn at akadia dot ch
Getting Secret key

3.4.6 Decryption

The decryption process is actually 2- step process which is reverse of the encryption process. First, the mobile user uses his secret key to decrypt the cipher text C2 (encrypted AES key by cpabe) to obtain the symmetric AES key. After this, the user uses this symmetric key, say Key_{AES} to decrypt the cipher text, C1 (encrypted using AES at sender site) to finally obtain sensor data. Code for sensor interfacing is written in python. Implementation of the command line compilation as well as the execution of the encryption code is done in the python program using sub-process module.

3.4.7 Bluetooth Interfacing

Bluetooth is a wireless technology (IEEE 802.15.1) used to exchange data over short distances (using short-wavelength radio transmissions in the ISM band from 2400–2480 MHz).

Bluez package has been used in the design of this secure sensor system SHS, which contains the Bluetooth protocol stack for Linux. A Bluetooth USB dongle namely LB4 – Bluetooth 4.0 Dongle is connected to one of the USB ports of the Raspberry Pi to establish a connection with a mobile phone. Any mobile device having the Bluetooth technology is able to receive the files from the Raspberry Pi device over the bluetooth. Obexftp has been used for the transfer of files using Bluez. It is implemented as a collection of command line instructions in the python program by using os and subprocess modules. Different mobiles have different data channel numbers (OPush channel number). Real time values are received from the accelerometer based sensor continuously for a specified period of time. Command line instructions are implemented in the python program to complete the above process of reading and storing values. The Raspberry Pi runs a python script which requests the private key stored on the memory/Java card used by Raspberry Pi to boot up. This symmetric key K_{AES}(128-bit) is then used by python script to encrypt the sensor data containing a list of readings of accelerometer sensor. The mobile device, which is an Android device in our case simply starts the Bluetooth and simply awaits the encrypted sensor data as well as AES key

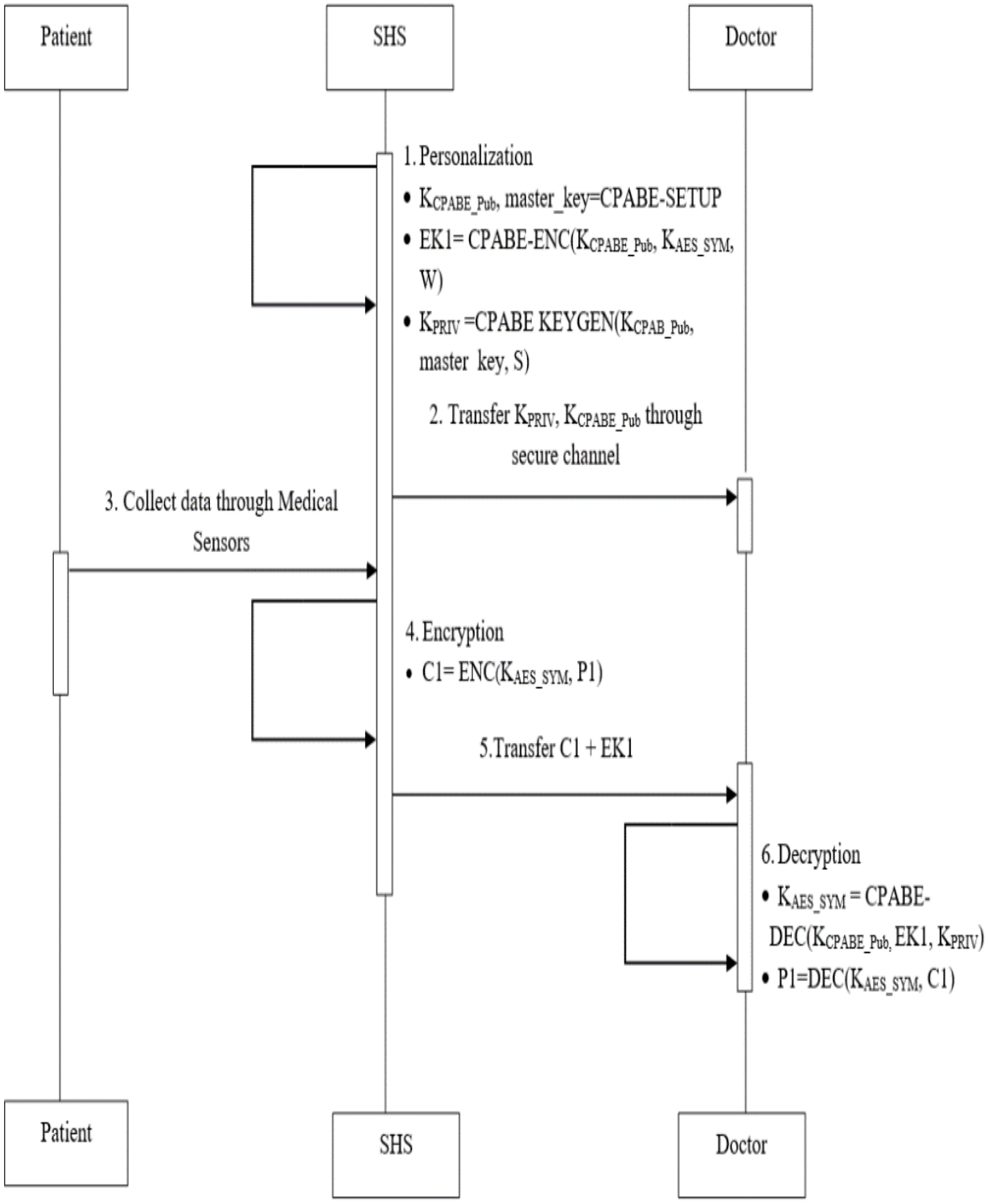


Figure 3.4 : Sequence Diagram of SHS system

which is encrypted. The mobile device receives the encrypted data and then decrypts it for further processing.

3.4.8 Data Transmission

The encrypted data can be transmitted to user over the Bluetooth medium. For this Bluetooth sockets and Client-Server architecture was used. On SHS, a server application is initiated which listens for incoming connections and transmits the encrypted data. Bluetooth client application at the user knows the mac address of SHS and requests the data. Data can also be transmitted over the web using network sockets with the help of similar Client-Server architecture. For transmission over Bluetooth, Pybluez and bluetooth python packages were used for making the server application at the Health Sensor node.

Key Abbreviation	Description
K_{CPABE_Pub}	CPABE Public key
K_{AES_SYM}	AES symmetric key
K_{PRIV}	CPABE - User Secret key
EK1	Encrypted AES key
C1	Encrypted Medical Data
P1	Patient Data
master_key	CPABE- Master Key

Table 3.1 : Different Notations used in Sequence Diagram

CHAPTER 4

RESULTS & ANALYSIS

In this chapter, the experimentation results of SHS described in implementation part of previous chapter is done. The system basically comprised of secure health sensor SHS which also acts as a server (service provider) which can be contacted by a client (service requestor) using traditional web browser or a smart phone with Android operating system. The performance of the proposed design is shown in terms of number of users, execution time for symmetric encryption and decryption, cpabe encryption time and decryption time, comparison of ECC and RSA on certification generation, key generation, signature generation and verification times.

4.1 Environment Setup Specifications

SHS employed various hardware and software to accomplish the desire task of secure data transmission between different stack holders while providing the role based access mechanism. There are two ways provided for data transfer:

(1) Using Web Interface

(2) Bluetooth Transmission.

The overall software and hardware configurations used for experimentation are shown in below tables :

Software's Configuration:

Operating System	Raspbion – Jessie Lite
Platform	Python Flask
Technology	Python, PyBluez, Openssl
Language	Python
Editor	GEdit, Vim
Backend	SQLite 3
Design	HTML, JQuery, Java Script

Hardware's Configuration:

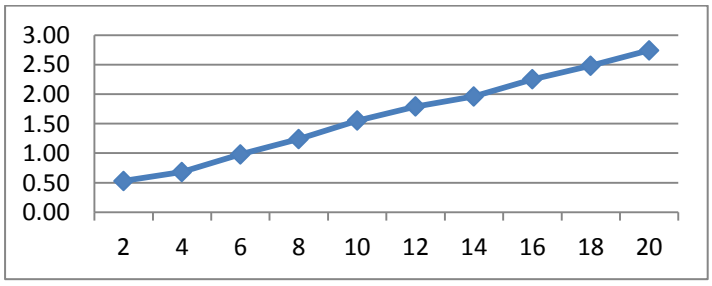
Chip	Broadcom BCM2835 SoC
Core Architecture	ARM11
CPU	700 MHz Low Power ARM1176JZFS Applications Processor GPU
Memory	512MB SDRAM

Table 4.1 Software/ Hardware Configuration

The hardware configuration listed above is specific to SHS which employs Raspberry Pi as a sensor node with networking and computation capability. The client or different stakeholders will need traditional web browser to interface with SHS.

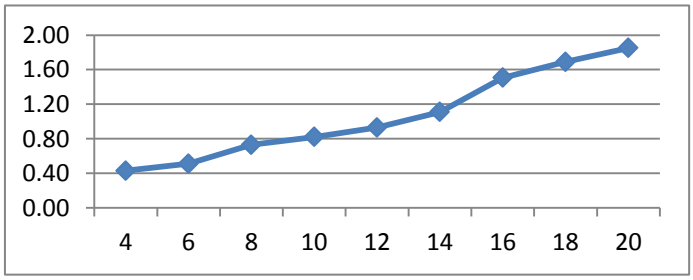
4.2 CPABE Secret Key Generation Time

Graph 4.1, we found that cpabe secret key generation time increases with number of attributes.



Graph 4.1 : Key Generation Time against number of attributes

4.3 CPABE – AES Symmetric Key Encryption Time



Graph 4.2: CPABE – KEY1 Encryption Time against number of attributes

Graph 4.2 above shows that time required for KEY1 encryption depends upon the number of attributes used in the policy.

4.4 Sensor Data Encryption and Decryption Time for AES algorithm

Table 2 shows the time taken for encryption and decryption on Raspberry Pi. We used different length AES key KEY1 for encrypting the sensor readings and time was nearly same on all three cases. While decryption time increases with key size KEY1.

Length of KEY1 (bytes)	**Average Encryption Time (millisecond)**	**Average Decryption Time (millisecond)**
16	2.4229	20.6866
24	2.6389	21.4863
32	2.6252	22.0068

Table 4.2 : Sensor data Ecnrytion and Decryption Time

4.5 Signature Generation

4.5.1 ECC based Signatures

S.No	Curve Type	Signature Generation Time	Signature Verification
1	secp112r1	42 ms	41 ms
2	secp128r2	43 ms	41 ms
3	secp160r2	44 ms	43 ms
4	secp192k1	47 ms	46 ms
5	secp224k1	49 ms	48 ms
6	secp256k1	52 ms	52 ms
7	secp384r1	57 ms	66 ms
8	secp521r1	92 ms	101 ms

Table 4.3 : ECC - Signature Generation and Verification Time

4.5.2 RSA based Signature

S.NO	Key Size	Signature Gen	Verification Time
1	512	3.639 ms	0.613 ms
2	768	6.639 ms	0.596 ms
3	1024	11.485 ms	0.91195 ms
4	2048	63.215 ms	2.01 ms
5	4096	371.656 ms	6.39 s

Table 4.4 : RSA - Signature Generation and Verification Time

4.6 Key Generation and Certificates

4.6.1 ECC Key and Certificate Generation Time

S.No	Curve Type	Key Generation Time	Certificate Generation Time
1	secp112r1	70 ms	90 ms
2	secp128r2	73 ms	92 ms
3	secp160r2	75 ms	85 ms
4	secp192k1	85 ms	101 ms
5	secp224k1	97 ms	108 ms
6	secp256k1	102 ms	112 ms
7	secp384r1	104 ms	159 ms
8	secp521r1	138 ms	240 ms

Table 4.5 : ECC – Certificate and Key Generation Time

4.6.2 RSA Key and Certificate Generation Time

S.NO	Key Size	Key Generation	Certificate Generation
1	512	0.130 s	0.0092 s
2	768	0.0876 s	0.005 s
3	1024	0.5303 s	0.01306 s
4	2048	6.068 s	0.059628 s
5	4096	48.276 s	0.3737

Table 4.6 : RSA – Certificate and Key Generation Time

4.7 Data transmission time over Bluetooth

The average transmission time over bluetooth medium was found to be 4.756 seconds. The experiment was conducted several times and the average of all times is taken.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this work, we have designed a secure health sensor (SHS) node capable of securing sensitive medical data. SHS designed is flexible and provides fine grained access control. The size of the Raspberry Pi being that of a mobile device provides compactness and portability to the sensor node. The use of Bluetooth Low Energy dongle makes system more energy and cost efficient. The encryption, key generation and transmission time suggests that our system can be incorporated in existing health care centres providing health services. The Raspberry Pi along with Bluetooth Low Energy has interfaces which are easy to use, and can help designers in investigation and development of new sensors as well as encryption and compression algorithms for future sensors. The web interface can be used to remotely access the SHS by different stakeholders.

Java cards can be used as memory element for secure symmetric key storage on secure element (SE), thus enhancing the security of SHS. Instead of storing the symmetric key (K_{AES_SYM}) in encrypted form as EK2, it can be stored securely in java card and accessed by java card APIs, for encrypting sensor data. We could not incorporate it into the current design because java card interaction libraries with Raspberry pi (Arm platform) were not provided. We have implemented this on Linux based Intel 64-bit architecture systems. But, we could not replicate the same on raspberry pi since the libraries were not compatible with it and we couldn't get the compatible ones from the java card supplier. However, we can use Single Board Computer (SBC) like Intel Galileo 2 based on Intel architecture for implementing SHS using Java cards which can support java applet libraries. Raspberry Pi is based on ARM architecture.

Currently, our system includes sensors with wired connections. Wireless sensors can be incorporated into SHS by including appropriate wireless inventor kit for Raspberry Pi. We plan to improve upon the design of SHS by making it battery operated with switches to start the sensor. The limitation of data transmission through Bluetooth can be removed by making use of cloud servers to store the data. Users can request the data from the cloud server through appropriate user interface APIs. We are also planning to use QR codes for pairing between SHS and user mobile device used for accessing sensor information. This is an

alternate to using NFC controller. We use Bluetooth since it has higher throughout and supports easy bidirectional security handshake.

References

1. Shin, M. S.; Jeon, H. S.; Ju, Y. W.; Lee, B. J.; Jeong, S.P.; "Constructing RBAC Based Security Model in u-Healthcare Service Platform", The Scientific World Journal, (2015), Volume 2015 pp. 1-13,
2. Mohammed, S.; Fiaidhi, J.; "Ubiquitous Health and Medical Informatics: The Ubiquity 2.0 Trend and Beyond", Medical Information Science Reference, (2010) ISBN 978-1-61520-777-0
3. Chen, J.; Kwong, K.; Chang, D.; Luk, J.; Bajcsy, R. "Wearable Sensors For Reliable Fall Detection", Engineering in Medicine and Biology 27th Annual Conference(IEEE) Shanghai China, September 1-4, 2005.
4. "World's elderly to overtake number of infants",an article in the The Telegraph, UK,18th June ,2013.
5. Avancha, S.; Baxi, A.; Kotz, D.; "Privacy in mobile technology for personal healthcare", ACM Computing Surveys (CSUR), vol. 45 Issue 1, article 3, 2012.
6. Plug-n-Trust: Practical Trusted Sensing for mHealth .Jacob Sorber, Minho Shiny, Ron Peterson, David Kotz,Institute for Security, Technology, and Society, Dartmouth College, Hanover, NH, USA Dept. of Computer Engineering, Myongji University, South Korea
7. https://www.raspberrypi.org/
8. Banerjee, S.; Sethia, D.; Mittal, T.; Arora, U.; Chauhan, A., "Secure sensor node with Raspberry Pi," Multimedia, Signal Processing and Communication Technologies (IMPACT), 2013 International Conference on , vol., no., pp.26,30, 23-25 Nov. 2013.
9. Dimitriou, T.; Ioannis, K.; "Security Issues in Biomedical Wireless Sensor Networks", Applied Sciences on Biomedical and Communication Technologies First International Symposium, conference publication, 2008.
10. Menezes, A. J.; Oorschot, P. C.; Vanstone, S. A.; "Handbook of Applied Cryptography" Boca Raton, FL, USA: CRC Press, 1997.
11. Amini, S.; Verhoeven, R.; Lukkien,J.; Chen,S.; "Toward a Security Model for a Body Sensor Platform", IEEE International Conference on Consumer Electronics (ICCE), 2011
12. Datasheet archives, contains datasheet of various ICs(ADXL345), www.datasheetarchive.com.
13. Getting started with Raspberry Pi ,Matt Richardson and Shawn Wallace, published by O'Reilly Media, First release December 2012.

14. Goyal, V., Pandey, O., Sahai,A., Waters,B.(2006) "Attribute –based encryption for fine grained access control of encrypted data", ACM Conference on Computer and Communication Security, pp. 89-98.

15. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, pp. 321–334. IEEE Computer Society, Los Alamitos (2007)

16. http://acsc.cs.utexas.edu/cpabe/

17. Cheung, L.; Newport, C. (2007) "Provably secure Ciphertext police ABE", CCS 2007: Proceedings of the 14th ACM conference on Computer and Communications security, pp.456 -465, ACM Press, New York.

18. Nishide, T., Yoneyama, K., Ohta, K.,(2008) "ABE with partially hidden encryptor specified access structure", ACNS'08, LNCS 5037, pp 111-129, Springer.

19. Kapadia, A., Tsang, P.P. and Smith, S.W. (2007) 'Attribute-based publishing with hidden credentials and hidden policies', in NDSS, Vol. 7, pp.179–192.

20. Smart Card Standards for contact and contactless interfaces, http://www.smartcardalliance.org/pages/smart-cards-intro-standards

21. R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. Journal of Cryptology, 20(3):265–294, 2007

22. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, pp. 321–334. IEEE Computer Society, Los Alamitos (2007)

23. https://www.raspberrypi.org/forums/viewtopic.php?f=91&t=74176

24. Sudhir G. Nikhade, "Wireless Sensor Network System using Raspberry Pi and Zigbee for Environmental Monitoring Applications" Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), 2015 International Conference on, vol., no., pp 376-381 May 2015

25. Anuradha Kar, Asim Kar "A novel design of a portable double beam-in-time spectrometric sensor platform with cloud connectivity for environmental monitoring applications"

26. M. Saari, P. Sillberg, P. Rantanen, J. Soini and H. Fukai, "Data Collector Service – Practical Approach with Embedded Linux", MIPRO 2015, 25-29 May 2015

27. V. Vujović, and M. Maksimović, "Raspberry Pi as a wireless sensor node: performances and constraints," 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) , pp. 1247–1252, 2014.

28. L. B. Oliveira, A. Kansal, B. Priyantha, M. Goraczko, and F. Zhao, "Secure-TWS: Authenticating node to multi-user communication in shared sensor networks," Comput. J., vol. 55, no. 4, pp. 384–396, 2012.

29. P. Czypek, S. Heyse, and E. Thomae, "Efficient implementations of MQPKS on constrained devices," Cryptogr. Hardware Embedded Syst., vol. 7428, pp. 374–389, 2012.

30. N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing elliptic curve cryptography and RSA on 8-bit CPUs," Cryptogr. Hardware Embedded Syst., vol. 3156, pp. 119–132, 2004.

31. A. S. Wander, N. Gura, H. Eberle, V. Gupta, and S. C. Shantz, "Energy analysis of public-key cryptography for wireless sensor networks," Proc. IEEE PerCom, 2005, pp. 324–328.

32. Public-Key Infrastructure (X.509), (pkix). [Online]. Available: http:// www.ietf.org/proceedings/59/211.htm

33. A. Shamir, "Identity-based cryptosystems and signature schemes," in Proc. Crypto, 1984, pp. 47–53.

34. R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Commun. ACM, vol. 32, no. 2, pp. 130–126, Feb. 1978.

35. N. Koblitz, "Elliptic curve cryptosystems," Math. Comput., vol. 48, pp. 203–209, 1987

36. V. S. Miller, "Use of elliptic curves in cryptography," in Proc. CRYPTO, 1986, pp. 417–426.

37. X. Chen, K. Makki, K. Yen, and N. Pissinou, "Sensor network security: A survey,"IEEE Commun. Surveys Tuts., vol. 11, no. 2, pp. 52–73, 2nd Quart. 2009.