LETTER OF TRANSMITTAL

June, 2018

To

Md. Ashraf Kamal

Sr. Lecturer

Department of Computer Science and Engineering

World University of Bangladesh

House #3/A, Road #4, Dhanmondi, Dhaka-1205, Bangladesh

Subject: Submission of thesis Report.

Dear Sir,

We are pleased to submit the report entitled "**Design and Implementation of a Lightweight Encryption Algorithm for IoT Devices**". It was grated pleasure to work on such an important topic. The report is prepared according to the requirements and guideline of the Department of Computer Science and Engineering, World University of Bangladesh (WUB).

We believe that the report will help you evaluate our thesis work. It would be a great pleasure for us to interpret a part or whole of the report whenever necessary.

Sincerely yours	Sincerely yours	Sincerely yours
________________	________________	________________
Arif Hussain	**Harun Ur Rashid**	**Riaz Rahman**
WUB 03/15/33/1463	WUB 03/15/33/1453	WUB 03/15/33/1458

World University of Bangladesh

DECLARATON

We hereby solemnly declare that the thesis work entitled "**Design and Implementation of a Lightweight Encryption Algorithm for IoT Devices**" has been supervised by **Md. Ashraf Kamal** Sr. lecturer of the department of Computer Science and Engineering, Would University of Bangladesh. We ensure that the thesis report has not been submitted either in whole or part for any degree in any university.

We hereby proof that the work we have presented does not breach any existing copyright rule.

We further undertake to indemnify the university against any loss or damage arising from breach of the forgoing obligation.

Sincerely yours	Sincerely yours	Sincerely yours
Arif Hussain	**Harun Ur Rashid**	**Riaz Rahman**
WUB 03/15/33/1463	WUB 03/15/33/1453	WUB 03/15/33/1458

Department of Computer Science and Engineering

World University of Bangladesh

CERTIFICATE

We hereby certify that the thesis Report on "**Design and Implementation of a Lightweight Encryption Algorithm for IoT Devices**" is a confide record of thesis work done by **Arif Hussain**, **Harun Ur Rashid** and **Riaz Rahman** for partial fulfillment of the requirements for award degree of the Bachelor of Computer Science and Engineering from World University of Bangladesh(WUB).

The thesis has been carried out under my guidance and is a record of the bona-fide work carried out successfully by the students.

Supervisor

____

Md. Ashraf Kamal

Sr. Lecturer

Department of Computer Science and Engineering

World University of Bangladesh

Department of Computer Science and Engineering

World University of Bangladesh

ACKNOWLEDGEMENT

We are extremely grateful and remain indebted to almighty ALLAH who has guided us in ventures to complete our thesis work successfully.

We would like to thank our family who always support us in our study. They were the reason that keeps us go further and to be successful in our life.

It is pleasure to thank the vice chancellor of World University of Bangladesh, **Professor Dr. Abdul Mannan Chowdhury**, to whom we owe a lot for giving us the opportunity to complete our thesis.

Also we would like to take the chances to thank our supervisor **Md. Ashraf Kamal** who was the light that we used in the darkness which guided us to the right way. He has been the most helpful to us and supported us when we needed his. We are thankful for his constant constructive criticism and valuable suggestion, which benefited us lot while developing the thesis. He was the source of inspiration and motivation for hard work.

Finally, we would like to thank all teachers and staffs of World University of Bangladesh for their kind assistance and support that provide us a perfect environment and good facility to help us in accomplishment of our thesis.

Sincerely yours

Arif Hussain

WUB 03/15/33/1463

Sincerely yours

Harun Ur Rashid

WUB 03/15/33/1453

Sincerely yours

Riaz Rahman

WUB 03/15/33/1458

ABSTRACT

This project work is based on data security. Providing security on less computational IoT devices (like wearable's) is a big concern. We discuss the important features of wearable's devices and the security challenges. Existing security algorithms are not feasible for this types of devices. Traditional encryption algorithms like AES, DES are not suitable, as these devices have less power and processing speed.

A literature review usually precedes a research proposal. Our study of literature review are AES, DES, RSA algorithm and related works on lightweight encryption, BRA (Byte Rotation Algorithm).

The methodology is the general research strategy that outlines the way in which research is to be undertaken and, among other things, identifies the methods to be used in it. These methods, described in the methodology, define the means or modes of data collection or, sometimes, how a specific result is to be calculated.

We completed the project work using programming language Java (SE), mobile platform language, Socket Programming and platform Core Java (NetBeans IDE), mobile platform, Server/ Client nodes.

We first randomly shuffle the entered data. Same time we store the index number of shuffle data. Then we get ASCII value of the shuffle data and add the key. Then we get encrypted data. For decryption, firstly we subtract the key from encrypted data and re-shuffle the data. Then we found original data. Similar ways we try more than 10 and system provide successfully result 100% positive.

We have implemented this algorithm only on small text and small IoT based devices. In Future, we will implement this algorithm on all types of file and file size.

Table of Contents

List of Figure

CHAPTER 1 INTRODUCTION

1.1 Introduction

Data Security is an important issue now a days. It is a digital privacy which is prevent unauthorized access to computer, database and websites. We know that IoT based devices increase rapidly. Information as IDC (International Data Corporation), within the years of 2025 approximately 80 billion devices will be connected to the internet. In 2020 there will be approximately 25 billion IoT enabled devices. Many of times when this device communicate by internet it is infected by different types of malware or malicious code at a high rate. On the other hand, IoT devices is generally smaller in size and low powered. So this device need less complexity algorithm but high security for transmission data. In this approach we are trying to develop an encryption algorithm for IoT or lightweight data transmission.

IoT can be connected using wireless connections which are RFID (Radio Frequency Identification), Bluetooth, ZigBee, WSN (Wireless Sensor Network), WAN (Wide Area Network) or Wi-Fi (Wireless Fidelity). Different types of bands network are available for communication which network are licensed and unlicensed.

The main objective of this project is to anlysis and find challenges of various encryption algorithm. Then generate an ideal solution and develop an encryption algorithm for IoT or lightweight security.

1.2 Objectives

- To develop a lightweight encryption algorithm with small size of keys.
- To reduce processing complexity and provide powerful data security for wearable devices.

1.3 Scope of Study

Our main goal was to develop an application for the IoT based devices to provide data security. Because present situation there is no good protection for this huge number of devices. Also, they are less powerful. So many encryption algorithms are not running on this devices for its less configuration. So our goal was to develop an algorithm for this device for providing data security. Finally, we make an application which is not needed any kind of existing algorithm. Finally it can send secure data by communication medium without facing any problem.

1.4 Motivation

IoT devices is still young at this time. It helps in creating connections between dissimilar things present in different types of environment. This kind of openness and very less human intervention can make Iot exposed to number of attacks like man in middle attack, Denial of Service (DoS) attack. Moreover, any device can access the network that leads to unauthorized access to data. So there are lots of work in this field to establish this. This is our main motivation. Also, this technology will be the famous shortly. Still, this field can’t provide the best security to exchange data. So we are trying to work this field.

1.5 Problem Statement

In the all previous work proposed by the researchers, we found the majority of the algorithm could be the modification traditional encryption algorithms. But nevertheless these algorithms have security issues and they're not light weight. Encryption algorithms proposed for IoT devices must certainly be fast and it will need less memory. With a view to the here we've proposed a brand new method which will be fast, light-weight and secure.

CHAPTER 2 LITERATURE REVIEW

2.1 Related Works

IoT based devices can use internet only for connecting and establishing communication between thins in the network. Data encryption is a technique which translates data into another form, or code so that only people with access to a secret key (formally called a decryption key) or password can read it. So, the Encrypted information ordinarily alluded to as figure content, while decoded information is called plaintext. Right now, encryption is a standout amongst the most famous and successful information security strategies utilized by associations.

2.2 Related works on lightweight encryption

In 1978, Ron Rivest, Adi Shamir, and Leonard Adleman presented a cryptographic calculation, which was basically to supplant the less secure National Bureau of Standards (NBS) calculation. Generally significantly, RSA actualizes an open key cryptosystem, and in addition computerized marks. RSA is persuaded by the distributed works of Diffie and Hellman from quite a while previously, who portrayed the possibility of such an calculation, yet never really created it.

This concept introduced two important ideas. One is Public-key encryption and other is Digital signature. This cryptosystem called asymmetric encryption. Each user has their own encryption and decryption procedures, E and D, with the former in the public file and the latter kept secret. These procedures are related to the keys, which, in RSA specifically, are sets of two special numbers. We of course start out with the message itself, symbolized by M, which is to be "encrypted". There are four procedures that are specific and essential to a public-key cryptosystem: a) Deciphering an enciphered message gives you the original message, specifically

$$D(E(M)) = M . (1)$$

b) Reversing the procedures still returns M: $E(D(M)) = M$. (2)

c) E and D are easy to compute.

d) The publicity of E does not compromise the secrecy of D, meaning you cannot easily figure out D from E. With a given E, we are still not given an efficient way of computing D. If $C = E(M)$ is the ciphertext, then trying to figure out D by trying to satisfy an M in $E(M) = C$ is unreasonably difficult: the number of messages to test would be impractically large. An E that satisfies (a), (c), and (d) is called a "trap-door one-way function" and is also a "trap-door one-way permutation". It is a trap door because since it's inverse D is easy to compute if certain "trapdoor" information is available, but otherwise hard. It is one-way because it is easy to compute in one direction, but hard in the other. It is a permutation because it satisfies (b), meaning every ciphertext is a potential message, and every message is a ciphertext of some other message. Statement (b) is in fact just needed to provide "signatures". Now we turn to specific keys, and imagine users A and B (Alice and Bob) on a two-user public-key cryptosystem, with their keys: EA, EB, DA, DB.

So far, we expect to make E and D easy to compute through simple arithmetic. We must now represent the message numerically, so that we can perform these arithmetic algorithms on it. Now lets represent M by an integer between 0 and $n - 1$. If the message is too long, sparse it up and encrypt separately. Let e, d, n be positive integers, with (e, n) as the encryption key, (d, n) the decryption key, n = pq. Now, we encrypt the message by raising it to the eth power modulo n to obtain C, the ciphertext. We then decrypt C by raising it to the dth power modulo n to obtain M again. Formally, we obtain these encryption and decryption algorithms for E and D:

$$C \equiv E(M) \equiv M^{e} (\text{mod } n) \qquad (5)$$

$$M \equiv D(C) \equiv C^{d} \ (\text{mod } n) \ .$$

Note that we are preserving the same information size, since M and C are integers between 0 and n − 1, and because of the modular congruence. Also note the simplicity of the fact that the encryption/decryption keys are both just pairs of integers, (e, n) and (d, n). These are different for every user, and should generally be subscripted, but we'll consider just the general case here. Now comes the question of creating the encryption key itself. First, choosing two "random" large primes p and q, we multiply and produce n = pq. Although n is public, it will not reveal p and q since it is essentially impossible to factor them form n, and therefore will assure that d is practically impossible to derive from e. Now we want to obtain the appropriate e and d. We pick d to be a random large integer, which must be coprime to (p − 1) · (q − 1), meaning the following equation has to be satisfied:

$$\gcd(d,(p-1) \cdot (q-1)) = 1 \ . \qquad (6)$$

"gcd" means greatest common divisor. The reason we want d to be coprime to (p − 1) · (q − 1) is peculiar. I will not show the "direct motivation" behind it; rather, it will become clear why that statement is important when l show towards 4 the end of this section that it guarantees (1) and (2). We will want to compute e from d, p, and q, where e is the multiplicative inverse of d. That means we need to satisfy

$$e \cdot d = 1 \ (\text{mod } \varphi(n)) \ . \qquad (7)$$

Here, we introduce the Euler totient function φ(n), whose output is the number of positive integers less than n which are coprime to n. For primes p, this clearly becomes φ(p) = p − 1 . For n, we obtain, by elementary properties of the totient function, that

$$\varphi(n) = \varphi(p) \cdot \varphi(q) = (p-1) \cdot (q-1) \qquad (8)$$

$$= n - (p + q) + 1 \ .$$

From this equation, we can substitute φ(n) into equation (7) and obtain e · d ≡ 1 (mod φ(n)) which is equivalent to e · d = k · φ(n) + 1 for some integer k. By the laws of modular arithmetic, the multiplicative inverse of a modulo m exists if and only if a and m are coprime. Indeed, since d and φ(n) are coprime, d has a multiplicative inverse e in the ring of integers modulo φ(n). So far, we can safely assured the following:

$$D(E(M)) \equiv (E(M))^{d} \equiv (M^{e})^{d} (\text{mod } n) = M^{e\cdot d} \ (\text{mod } n)$$

$$E(D(M)) \equiv (D(M))^{e} \equiv (M^{d}) \ e \ (\text{mod } n) = M^{e\cdot d} \ (\text{mod } n)$$

Also, since e · d = k · φ(n) + 1, we can substitute into the above equations and obtain

$$M^{e\cdot d} \equiv M^{k\cdot\varphi(n)+1} \ (\text{mod } n) \ .$$

Clearly, we want that to equal M. To prove this, will need an important identity due to Euler and Fermat: for any integer M coprime to n, we have $M^{\varphi(n)} \equiv 1$ (mod n) . (9) Since we previously specified that 0 ≤

M < n, we know that M would not be coprime to n if and only if M was either p or q, of the integers in that interval. Therefore, the chances of M happening to be p or 5 q are on the same order of magnitude as 2/n. This means that M is almost definitely relatively prime to n, therefore equation (9) holds and, using it, we evaluate:

$M^{e\text{-}d} \equiv M^{k\text{-}\varphi(n)+1} \equiv (M^{\varphi(n)})^{k} \equiv 1^{k}M \pmod n = M$.

It turns out this works for all M, and in fact we see that (1) and (2) hold for all M, $0 \leq M < n$. Therefore E and D are inverse permutations.

2.2.1 CryptoCop

As individuals utilize and collaborate with an ever increasing number of wearables and IoT-empowered gadgets, their private data is being uncovered with no security insurances. Nonetheless, the restricted capacities of IoT gadgets make executing strong security assurances testing. Accordingly, they show CryptoCoP, a vitality effective, content skeptic security and encryption convention for IoT gadgets. Busybodies can't snoop on information secured by CryptoCoP or track clients by means of their IoT gadgets. We assess CryptoCoP and demonstrate that the execution and vitality overheads are reasonable in a wide assortment of circumstances, and can be adjusted to exchange off forward mystery and vitality utilization against required key stockpiling on the gadget (R. Snader et al. 2016)Authors proposed an encryption protocol for using it on resource-constrained devices.

- That supports energy-efficient symmetric key encryption by offloading expensive cryptographic computations from the wearable device and uploading the keying material to the device while the time of charging.

- "CryptoCoP" uses key rotation so that collected data cannot be recovered by an attacker even if the device is stolen.

2.3 Five Common Encryption Algorithms and the Unbreakables of the Future

1. Triple DES

Triplr DES was designed to replace the original Data Encryption Standard (DES) algorithm, which hackers eventually learned to defeat with relative ease. At one time, Triple DES was the recommended standard and the most widely used symmetric algorithm in the industry.

2. RSA

RSA is a public-key encryption algorithm and the standard for encrypting data sent over the internet.
It also happens to be one of the methods used in our PGP and GPG programs.

3. Blowfish

Blowfish is yet another algorithm designed to replace DES. This symmetric cipher splits messages into blocks of 64 bits and encrypts them individually.

4. Twofish

Computer security expert Bruce Schneier is the mastermind behind Blowfish and its successor Twofish. Keys used in this algorithm may be up to 256 bits in length and as a symmetric technique, only one key is needed.

5. AES

The AES is the algorithm trusted as the standard by the U.S. Government and numerous organizations.

Although it is extremely efficient in 128-bit form, AES also uses keys of 192 and 256 bits for heavy duty encryption purposes. (5 Common Encryption Algorithms and the Unbreakables of the Future. 2017)

CHAPTER 3 METHODOLOGY

3.1 Introduction

This chapter outlines the study area of the research, the research methodology that will be used in the collection and analysis of data, a description of the research design, research tools, instruments of data collection and procedures of data collection and analysis is also given.

3.2 Block diagram of proposed methodology

This is the full steps to solve our problem

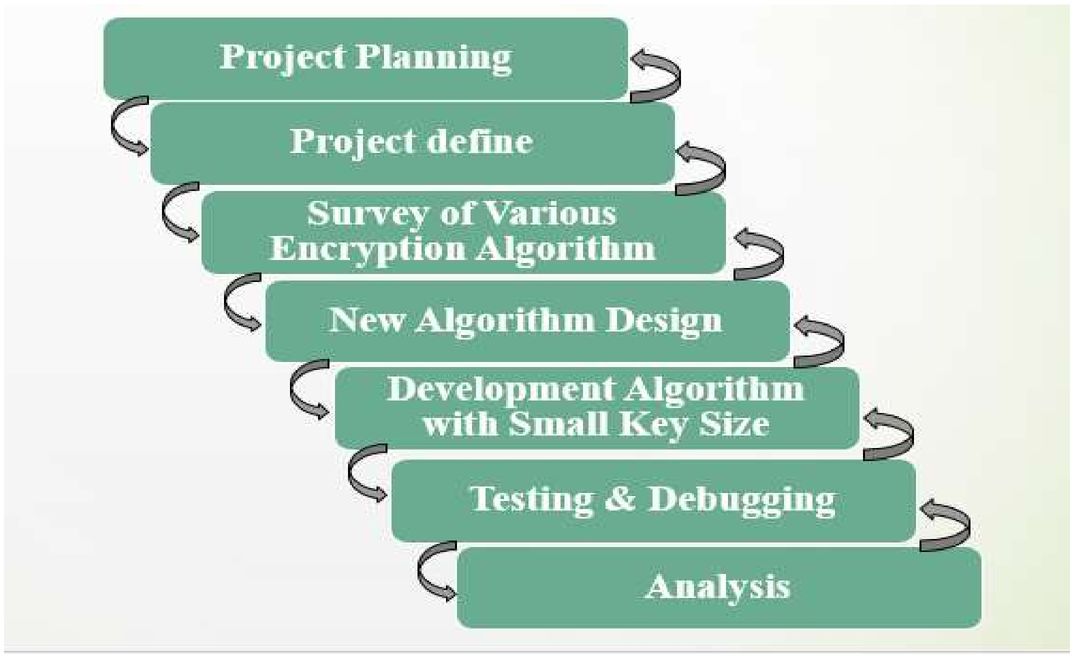

Project Planning
Project define
Survey of Various Encryption Algorithm
New Algorithm Design
Development Algorithm with Small Key Size
Testing & Debugging
Analysis

Figure: 3.1: Block diagram of methodology

3.2.1 Problem Define

It is a process of the identified user needs may be the satisfied problem.

Data security is one of the major concerns while transferring the data through wireless media. Encryption is the one of the best and highly used technology to secure the data. Devices like 'In Body Sensors,' wearable devices, and other real-time sensors may have to work on battery power and have less computational power. So, Encryption Algorithm like AES, DES is not infeasible for this devices. Wearable are able to the photo. But it not provide any security of this data. Most wearable's connected to smartphone or tablets wirelessly, potentially hacker's easy access to devices and a launch pad to an attack on a corporate network. Many wireless communications cannot guard against a determined, forced attack.

There are very few algorithm for Encryption and secure data transferring for this type of devices. But this algorithm are not suitable for less intensive devices. The existing algorithm's that are working on high configured devices not working on this devices. For this reason, devices are less secure than other devices. So it is a big Problem to provide security of data.

3.2.2 Study of Analysis

Study of analysis give the requirement specification, a search is made for analysis to implement the specification. Usually, there is no exact match, and the analysis may be used only provide some of the functionality required.

CryptoCoP

As people use and interact with more and more wearables and IoT-enabled devices, their private information is being exposed without any privacy protections. However, the limited capabilities of IoT devices make implementing robust privacy protections challenging. In response, they present CryptoCoP, an energy-efficient, content agnostic privacy and encryption protocol for IoT devices. Eavesdroppers cannot snoop on data protected by CryptoCoP or track users via their IoT devices.

DES

Triple DES was designed to replace the original Data Encryption Standard (DES) algorithm, which hackers eventually learned to defeat with relative ease. At one time, Triple DES was the recommended standard and the most widely used symmetric algorithm in the industry.

RSA

It works on generating public key and private key pair by selecting two large prime numbers. Find their modulus and choosing at random there encryption key and thus calculating the decryption key public key is published openly whereas private key is made secure. A more secure RSA encryption is proposed in that is used to encrypt and decrypt files for maintaining privacy of user.

AES

AES is used an COAP at application layer. It is a symmetric block cipher standardized by NIST. It uses substitution permutation network and works on 4*4 matrix having block length of 128 bits. Every byte gets affected by subbytes, shiftrows, MixedColumns, AddRoundKey. Key size than can be used is 128, 192, 256 bits. AES is still vulnerable to man-in middle attack.

3.2.3 Design and Development

System design during this phase, the framework of the system is design. The designers take into account the components that are reused and organize the framework to cater to this. Some new software may have to be designed.

Development that cannot be extremely procured is developed, component are integrate to create a new system. System integration, in this model may be part of the development process rather than a separate activity.

∗ We are concerning about secure data/file transferring between source to destination.

∗ Main aim is to reduce the computational complexity, as wearable devices has less computational capacity and memory.

∗ Sending data using Intermediate server to Final server.

3.2.4 Testing

The testing during this stage, software design is realized as a set of programs or program. Unit meets its specification.

Individual Programs testing:

∗ Jumble data part and successfully jumble the collect data.

∗ Sending part test and successfully send the jumble data to the Intermediate server.

∗ Received part test and successfully received the data file to server.

∗ Marge file part test and successfully marge the Splitting file and output the final file. Testing individual program units or programs are integrated and tested as a complete system to ensure that the software requirement have been met. After testing, the software system is delivered to the customer.

3.2.5 Result

Analysis our result we find good response. Findings can only confirm or reject the hypothesis underpinning your study. However, the act of articulating the results helps you to understand the problem from within, to break it into pieces, and to view the research problem from various perspectives. We first collect 128 bit data then jumble this data. All jumble data send to the Intermediate server. Intermediate server create a file and send it to final server. By user requesting final server send this data to user devices. User devices decrypt the data using key (jumble data index). Similar ways we try more than 10 and system provide successfully result.

3.3 Flow chart

This is the full flowchart of our problem

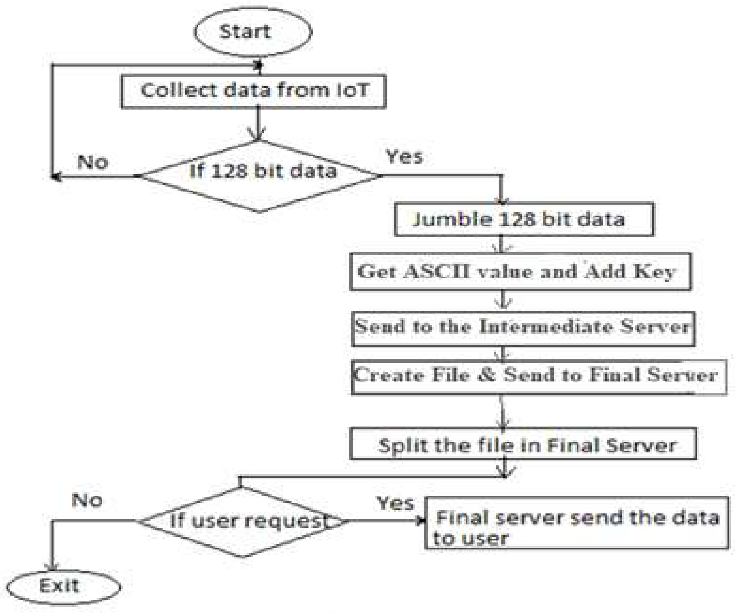

Start
Collect data from IoT
No
If 128 bit data
Yes
Jumble 128 bit data
Get ASCII value and Add Key
Send to the Intermediate Server
Create File & Send to Final Server
Split the file in Final Server
No
If user request
Yes
Final server send the data to user
Exit

Figure: 3.2 Project Flowchart

3.4 Proposed Algorithm

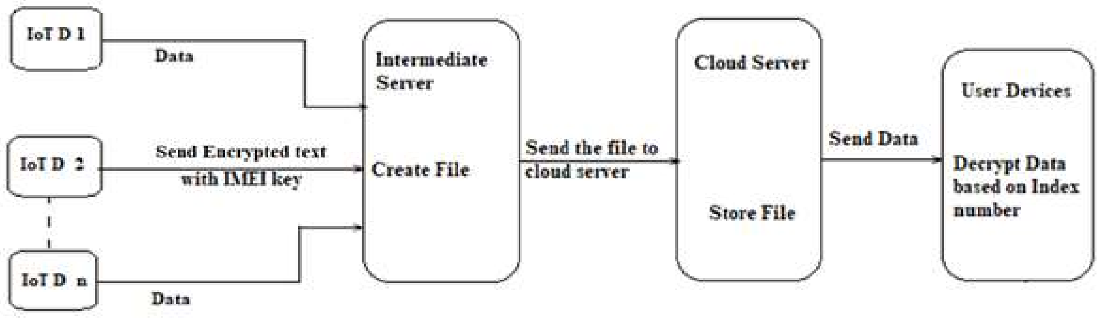


Figure: 3.3 Proposed Model Block Diagram

∗ So we are concerning about secure data/file transferring between source to destination.

∗ Main aim is to reduce the computational complexity, as wearable devices has less computational capacity.

∗ Sending data using server to server via client.

Step-1: Collect 128 bit data from IoT devices.

Step-2: Jumble 128 bit data and get key (index number).

Step-3: Get ASCII value of the jumble data.

Step-4: Add the ASCII value and key.

Step-5: Send the encrypted data to the Intermediate Server.

Step-3: Create a File of the encrypted data in Intermediate Server.

Step-4: Send the File to Final Server.

Step-5: Again, Final Server split the file into small file and send to the receivers.

Step-6: End.

CHAPTER 4 DESIGN, DEVELOPMENT AND ANALYSIS

4.1 Tools and Techniques

4.1.1 Netbeans (IDE)

NetBeans started as a student project (originally called Xelfi) in Czechoslovakia, now the Czech Republic, in 1996. The goal was to write a Delphi-like Java IDE (Integrated Development Environment) in Java. Xelfi was the first Java IDE written in Java, with its first pre-releases in 1997. Xelfi was a fun project to work on, especially since the Java IDE space was uncharted territory at that time.

4.1.2 Android Studio (IDE)

Android Studio most popular java IDE (integrated development environment) provides the fastest tools for building apps on every type of Android device. It was announced on May 16, 2013. It provides the fastest development for Android phone, Wearables, PDA etc.

4.1.3 Gradle and Open Source

Along the way, an interesting thing happened. People began building applications using the NetBeans IDE's platform, together with their own plugins, often creating applications that were not development tools at all. In fact, this turned out to have quite a market. Later, in 2000 and 2001, a lot of work went into stripping out pieces that made the assumption that an application built on NetBeans was an IDE, so that the platform would be a generic desktop application suitable to any purpose. This work turned out to be healthy for codebase of the IDE as well, encouraging a clean API design and a separation of concerns.

4.1.4 JDK 8

The full version string for this update release is 1.8.0_141-b15 (where "b" means "build"). The version number is 8u141.

4.1.5 JRE 8

The JRE expires whenever a new release with security vulnerability fixes becomes available. Critical patch updates, which contain security vulnerability fixes, are announced one year in advance on Critical Patch Updates, Security Alerts and Third Party Bulletin. This JRE (version 8u141) will expire with the release of the next critical patch update scheduled for October 17, 2017.

4.1.6 Java (SE) Language 8.2

Java is one of the high level programming language which is developed by Sun Micro System. For developing Android devices code we have used java code. In the middle portion core java also working for some task.

4.2 Windows OS

An operating system (OS) is software that governs the hardware resources, manages the processes and manages the memory. Similar to Windows 10, MAC OS X, and Linux is a free source operating system. Windows is one of the useable strong computer OS. It does not slow downs and crashes easily.

4.3 Warble Device

4.3.1 Overview of wearable devices:

Wearable devices are also known as wearable technology, wearable gadgets, or simply wearables. Wearable Devices is one kind of Electronics that can be worn on the body or in cloth. The major feature of the wearable technology is to connect Internet and exchange data between end to end user. Wearable technology is going to be popular day by day. Because of it's some major functionality. Similar to Smartphone we can use it. But without taking a hand or don't do any task we can use it. Because it integrated with your hand.

Example:

- Smart watches
- Smart Glasses
- Smart fabrics
- E-textiles etc.

Figure 4.1: Example of wearable devices

4.3.2 History of Wearable devices & there Operating System

The technology of wearable devices is still young at this time. The development of wearable devices is thus bound to encounter various problems such as functional singleness, incompatibility between operating system and software, the convenience of human-computer interaction, data transmission, confidentiality of the information, energy consumption problems brought by continuous running. The operating system is the interface between hardware and software. The main function of the operating system is to manage hardware, software and the resource of this devices. It also enables users to have a good working environment and provide services for the user. The operating system on wearable devices has gone through years of development.

In 2000, IBM collaborated with the Citizen to launch a smart watch named WatchPad which operating system was Linux. Fossil designed a wrist device called Wrist PDA in 2003. It equipped with PalmOS operating system. It was much popular at this time. Also, Microsoft designed in 2004 the SPOT system for smart watches. With integrated Android Operating system Samsung released its first smart watch Galaxy Gear in 2013. After that, Samsung launched the second generation of smart watch running Samsung's independently designed operating system Tizen.

In March 2014, Google launched a smart watch dedicated operating system called Android Wear, whose operation is implemented through Google Now's voice commands. Android Wear is expected to build a uniform and standard operating system platform, accelerating the development of wearable devices.

Google Android Wear

This device paring with Android 4.3+.Without the paired device Android does not work.

- Battery life: Bad, 1-2 days
- Graphics quality: High, including OLED, TF-LCD, and LCD screens
- Fitness tracking: Supported by additional software (Google Fit)
- Supported devices: Android 4.3+
- Chipsets: ARM Cortex-A and potentially x86 and MIPS at present, only Cortex-A processors are used
- Form Factor: Currently limited to smart watches.
- Voice recognition: Yes

Samsung Tizen

This operating system drives from Linux. Like Android, it contained open source code.

- Voice recognition: Yes, with manual activation
- Graphics quality: High, most use OLED screens
- Fitness tracking: Yes, with SAMI (and other Samsung apps)
- Open Source: Yes
- Chipsets: ARM Cortex-M, possibly more
- Supported devices: Select Samsung devices on Tizen and Android.

Linux-derivative

This operating systems dominate smaller and larger wearables. The Linux kernel supports ARM Cortex-M, Cortex-A, MIPS and many more architectures. Its extreme flexibility and open-source code base make it an ideal fit for smart watches.

- Voice recognition: No
- Supported devices: Some offer synchronization services with Android, iOS and Windows products.
- Open Source: Yes
- Battery life: Varies heavily by device
- Fitness tracking: Yes, depending on the device

Efficacy:

1) Convenience. The design of the operating system should be more convenient for users to use wearable devices.

2) Effectiveness. The operating system should be managed more effectively and take advantage of resources like hardware, software and data of wearable devices.

3) Scalability. The operating system should permit new system functions to be developed, tested and included. 4) Openness. The operating system should support integrated and collaborative network work of different manufacturers and devices so that it can achieve the portability and interoperability of applications.

5) Multitasking. The operating system should be able to run multiple applications concurrently. In general, there are several options for developers to develop wearable operating system.[?]

4.3.3 Why wearable devices are popular

The wearable devices are more convenient to the user because of it's functionality and lightweight. This device is easy to use and carry one place to another. You can do many functionalities that can be done to your Smartphone and tablets. But smart watch and other IoT devices are not occupied your hand .

1. It can be used when user are in motion.
2. When both hands are occupied with many tasks you can use it.
3. It should maintain the user control.

4.3.4 Lack of resource on wearable

Lack of resource for software configuration Input system: Its main fact for wearable devices to taking text input. Because of its display size. Wearable devices display size are smaller than other devices. So the use of the keyboard is not possible on this devices. Instead of keyboard, they use voice reorganization. Sometimes it takes input from motion sensors, speed, temperature, etc. This data might be wrong. For pointing, devices wearable devices use different think. Instead of mouse its use trackballs, touch pads. This thing can be used as a pointing device which can select the functionality.

Lack of resource for hardware configuration

This device is Less Computational Intensive devices. There is hardware configuration is very low . See the following:

- 1.2-inch P-OLED,
- 360 x 360-pixel display Ambient light sensor 316L stainless steel Gorilla Glass 3 Italian leather strap
- 1.1GHz Qualcomm Snapdragon Wear 2100 processor
- 256-512MB RAM
- 4-8GB storage
- 150-450mAh battery
- Wi-Fi, Bluetooth
- Using Android Version 4.3+

4.3.5 Potential security concerns for wearables

1. Easily accessible real data:

Wearable devices store its data on local devices. Some time doesn't provide any encryption in the real issue. Sometime doesn't provide PIN, password protection and no biometric security and no user authentication needed for accessing data on wearables. So it is easy to access potential data easily by an unauthorized user.

2. Lack of Encryption:

There is lack of encryption in wearable devices. Wearable are transmitted data or stored on manufacturer or service provider cloud server. So this transition is not secure. Also some third party software neglect security standard and store information without encryption. So its one of the main security concern for this kind of devices.

3. Body area network security:

A new challenge for wearables is the body area network which comprises of the wearable computer, all attached devices, and sensors etc., opens up for a number of security threats sensitive information stored on the wearable device is at risk, privacy and safety issues might arise if the wearer is id entified and the electronic identity stolen.

4. Insecurity of wireless connectivity:

Wearables are always connected with a Smartphone using protocol such as Bluetooth, NFC, etc . But in the same range, many of the devices may can be connected with this. So its insecure data communication.

4.4 System Design

The main aim of this system is to satisfy the requirement of Secure Data Transfer Technique for Less Intensive Devices. This system transferring data of full secure and low configuration device is supported.

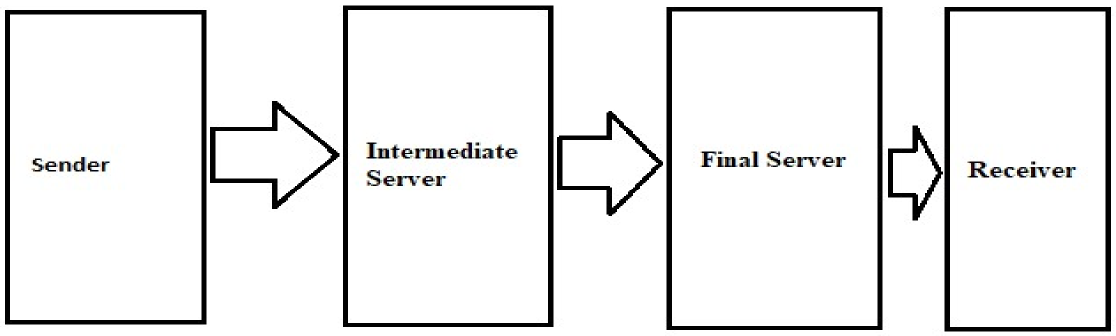


Figure 4.2: Block diagram of system design

4.5 System Development

We are supposing that real-time sensors/thin devices (like wearable devices, in body sensors and others) collecting the data continuously. When these devices collect 128 bits of data, then it will jumble/shuffle this information in a particular pattern or random order and generate the key of shuffle data which is index number of the shuffle data. After shuffle the data, it will connect the intermediate server and intermediate server store these shuffle data. An intermediate server can be connected with many numbers of such type of wearable devices and will get many data from these devices. Then shuffle data coming from different devices will store with IoT devices IMEI and IP address. Then the intermediate server create a file based on IMEI number. For, N number of IMEI number will create N number of files. The index of shuffle data is stored in that IoT devices which are secret for each other (i.e. key used by wearable devices is not known to an intermediate server, and the key used by the intermediate server is not known to input node). In a certain period or after getting the certain number of messages (in our case we use 20 numbers of messages) from different node to intermediate server it will create a file and send these data to the final server for storing purposes (final server split files and finally send to the receiver). The central server may not know the any of key so that without knowing these key it cannot retrieve the 20 message. Whenever a user comes with his both keys and personal identities, Server can decrypt the data. A typical Block diagram of our work is given below:

Mainly there in three node namely

- Input devices (having low computational power, battery power, low hardware configure etc.)
- Intermediate node (higher computational power, better configuration than input devices)
- Main/final server (having high computational power, very good processing power)

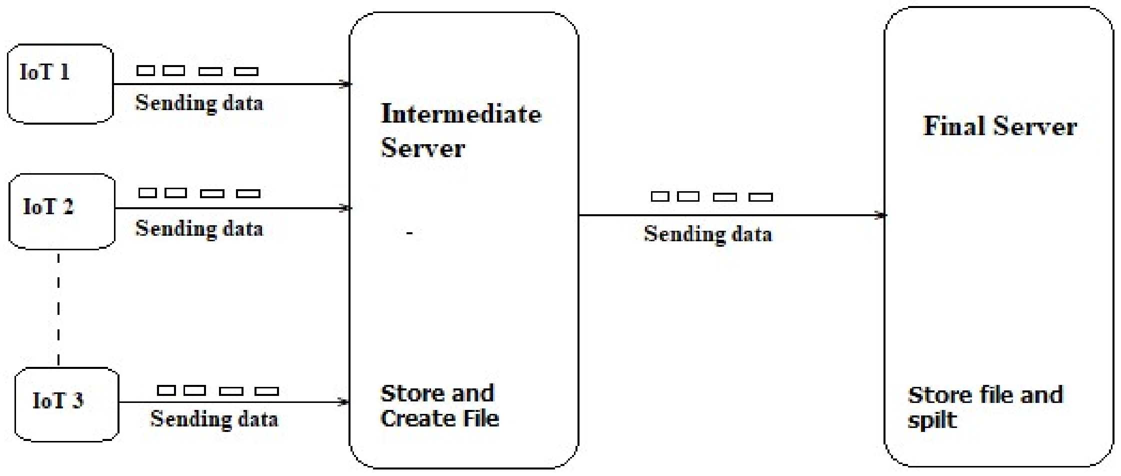


Figure 4.3: Typical Block diagram of proposed model

4.6 Data processing in input Devices

We assume input devices like wearable device, body sensors are generating data in real time basis.

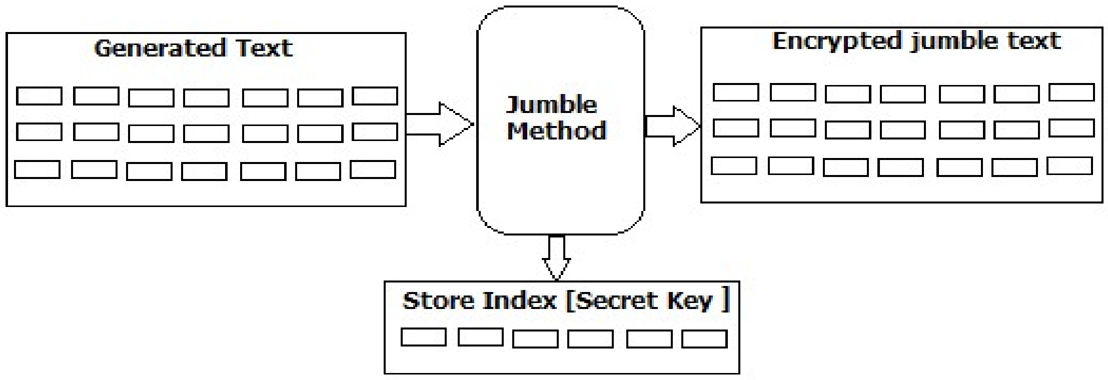


Figure 4.4: Data processing in input devices

As shown in a figure, data are collected by sensors/wearable devices. When the data gathered reach some threshold value (in our case we use 128bit) then using the key of the same length, it will rearrange (jumbled) the bits. Here the key is the index of bits on the original bit position. After jumbling the data, it will send it to the intermediate server along with its identity (in our case we use IMEI no as an identity of the device).

4.7 Data Processing in intermediate server

Intermediate server can be any computational devices with higher processing power and computational power then wearable/sensor devices.

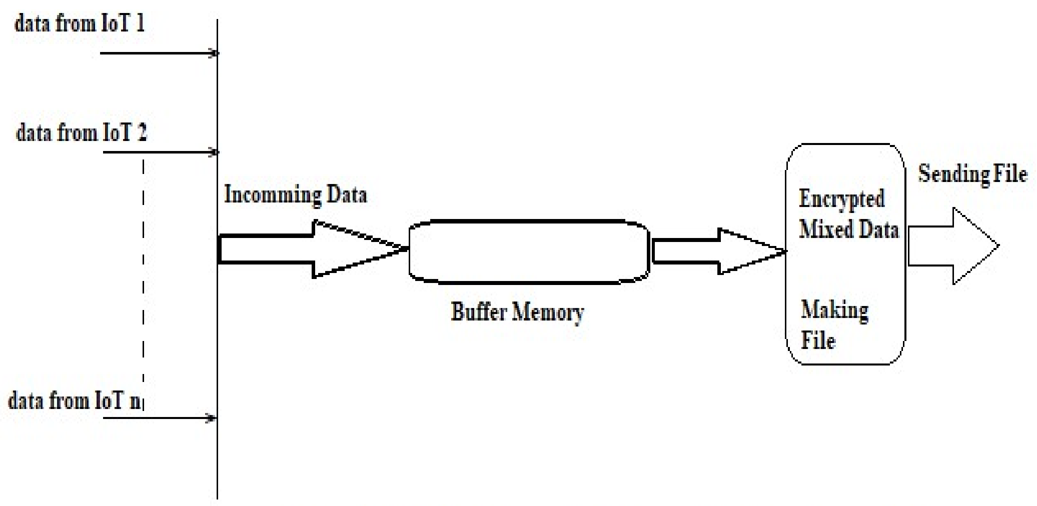


Figure 4.5: block diagram of data processing in intermediate server

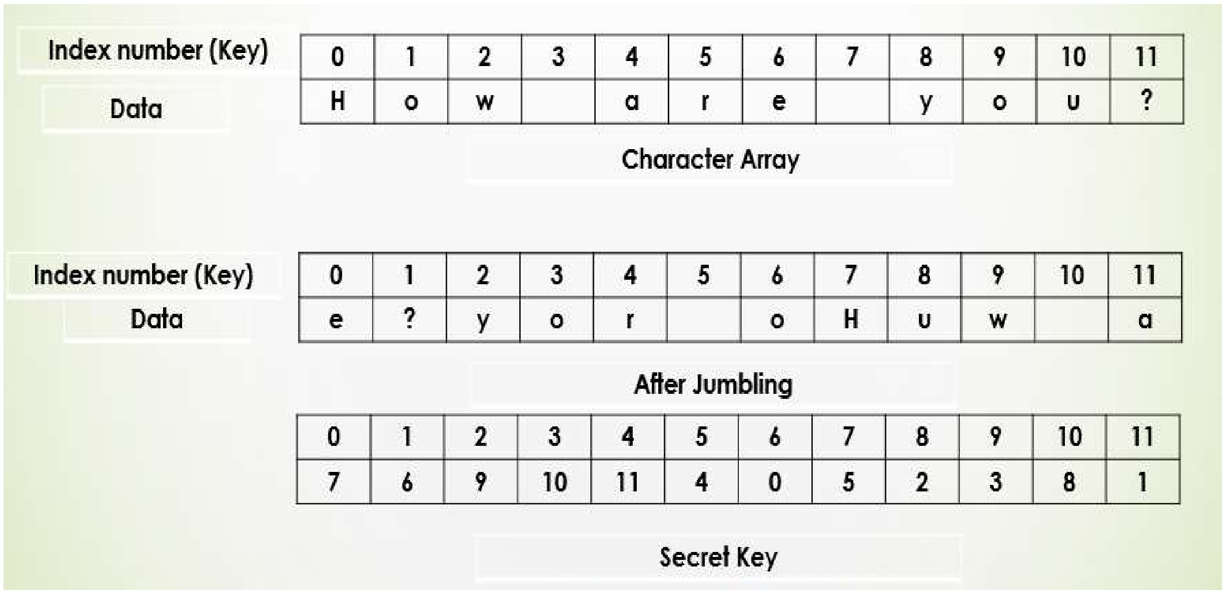
In this stage, an incoming message coming from different devices will store in a buffer memory (in our case buffer memory size should be more than or equal to 2*(20*128 bits+ space require to 22 store IMEI no)). After receiving 20 message from devices, it will count from how many device data are coming. Then it will count no. of received message from each device. The preinstalled key will be there on this server which store the relocation index for 20 incoming message (means in what way data will mix?). According to the number of an incoming message from each device, it will give the location for these messages which are in a random manner. So first, it will relocate these message. After storing it in temporary memory, it will copy these data to a file, and make a file. While making file, it will give filename as a timestamp so that we can search data of based on IMEI number. After making file, it will send this data to the final/main server.

4.8 Final Server

The main server will store the incoming images. If any user wants to see his data, he/she can request to the main server by giving his/her both key and the time range so that main server will extract and decrypt the data according to provided time range, devices address and both key. After retrieving the original message, it will send it to the user.

4.9 Key Generation with Example

Sending devices collect data continuously. When it jumble this data that time it store the index number. When we want to get original text that time using this index number rearrange the text. Here is example-



Index number (Key)	0	1	2	3	4	5	6	7	8	9	10	11
Data	H	o	w		a	r	e		y	o	u	?

Character Array

Index number (Key)	0	1	2	3	4	5	6	7	8	9	10	11
Data	e	?	y	o	r		o	H	u	w		a

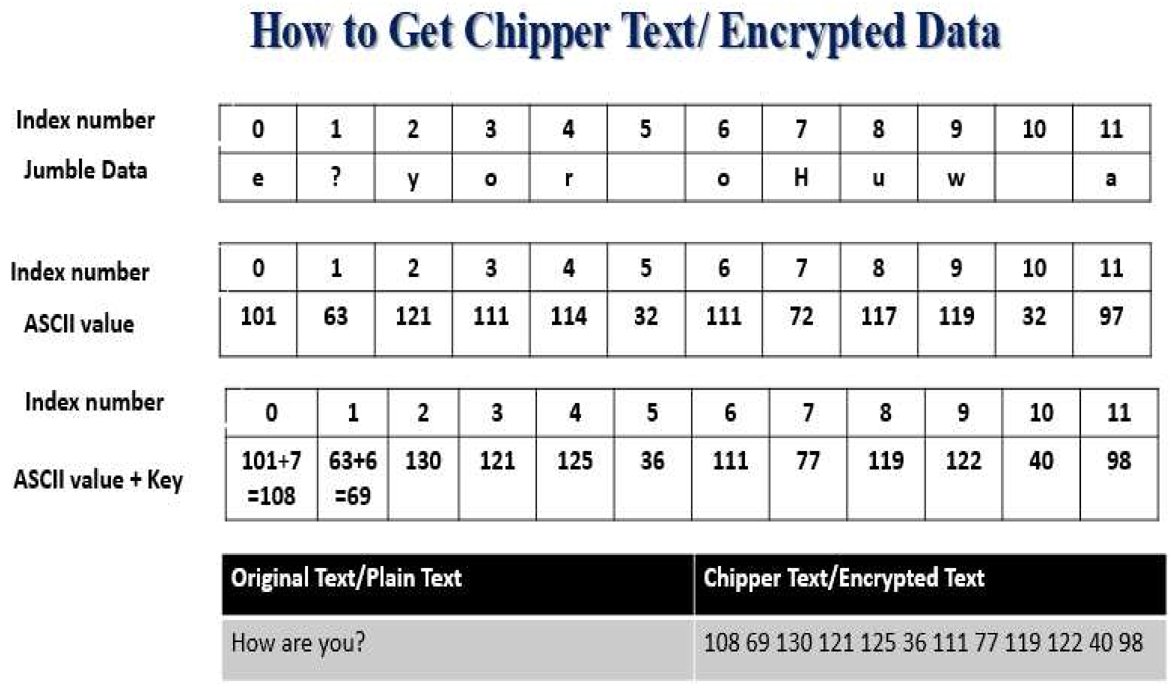
After Jumbling

0	1	2	3	4	5	6	7	8	9	10	11
7	6	9	10	11	4	0	5	2	3	8	1

Secret Key

Figure: 4.6 Key Generation

4.10 Get Chipper Text



How to Get Chipper Text/ Encrypted Data

Index number	0	1	2	3	4	5	6	7	8	9	10	11
Jumble Data	e	?	y	o	r		o	H	u	w		a

Index number	0	1	2	3	4	5	6	7	8	9	10	11
ASCII value	101	63	121	111	114	32	111	72	117	119	32	97

Index number	0	1	2	3	4	5	6	7	8	9	10	11
ASCII value + Key	101+7 =108	63+6 =69	130	121	125	36	111	77	119	122	40	98

Original Text/Plain Text	Chipper Text/Encrypted Text
How are you?	108 69 130 121 125 36 111 77 119 122 40 98

Figure: 4.7 Get Chipper Text

CHAPTER 5 TESTING AND RESULT

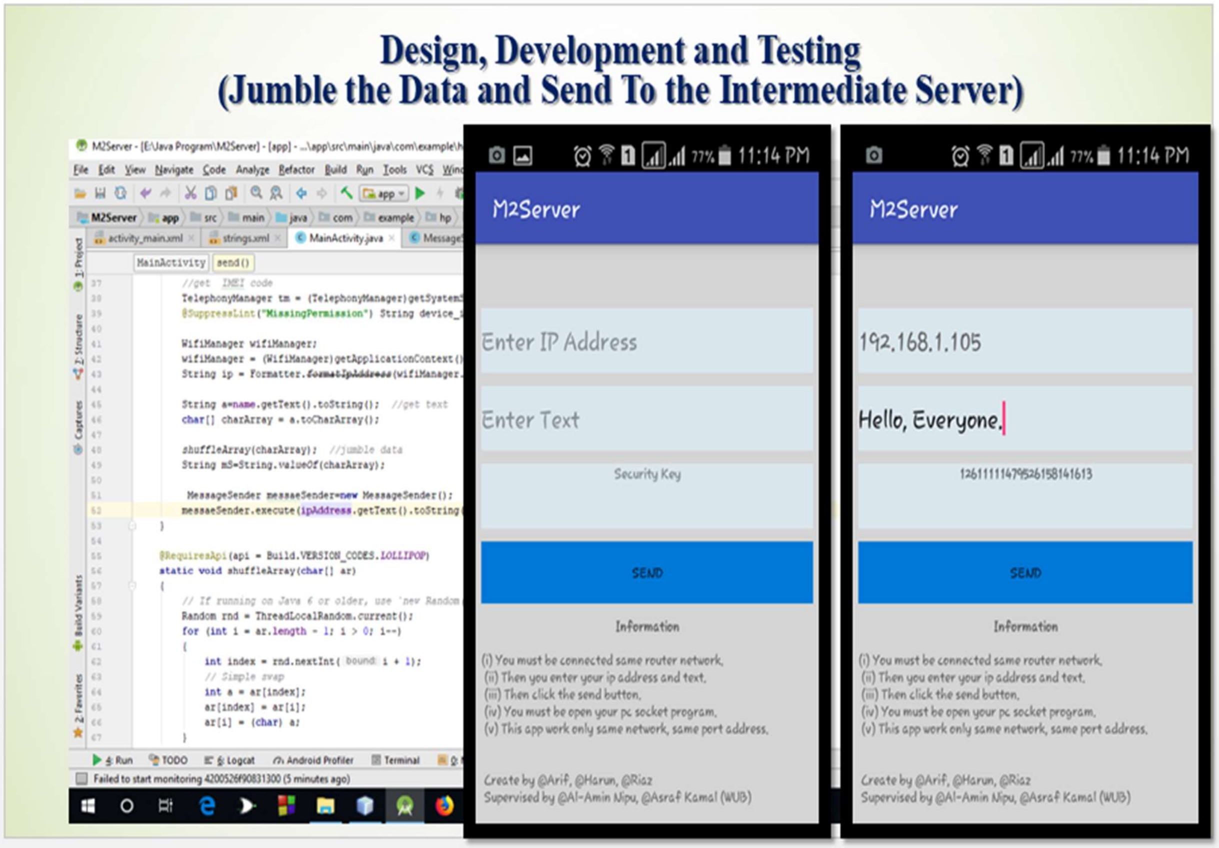
5.1 Testing

5.1.1 Creating Environment

We developed an application which generates random 128bits of data at a time in android by supposing that wearable device is generating data while in use. Then we generate a random 128 bits random non repeated value from 0-127 and stored these data as a key.

5.1.2 Jumble the Data and Send to the Intermediate Server

This application developed for jumble 128 bit data and store the index number. After jumbling the data this interface send the data to the intermediate server.



Design, Development and Testing
(Jumble the Data and Send To the Intermediate Server)

M2Server - [E:\Java Program\M2Server] - [app] - ...\app\src\main\java\com\example\h

```
        //get  IMEI code
        TelephonyManager tm = (TelephonyManager)getSystemS
        @SuppressLint("MissingPermission") String device_i

        WifiManager wifiManager;
        wifiManager = (WifiManager)getApplicationContext()
        String ip = Formatter.formatIpAddress(wifiManager.

        String a=name.getText().toString();  //get text
        char[] charArray = a.toCharArray();

        shuffleArray(charArray);  //jumble data
        String mS=String.valueOf(charArray);

         MessageSender messaeSender=new MessageSender();
        messaeSender.execute(ipAddress.getText().toString(
    }

    @RequiresApi(api = Build.VERSION_CODES.LOLLIPOP)
    static void shuffleArray(char[] ar)
    {
        // If running on Java 6 or older, use `new Random(
        Random rnd = ThreadLocalRandom.current();
        for (int i = ar.length - 1; i > 0; i--)
        {
            int index = rnd.nextInt( bound: i + 1);
            // Simple swap
            int a = ar[index];
            ar[index] = ar[i];
            ar[i] = (char) a;
        }
```

Failed to start monitoring 4200526f90831300 (5 minutes ago)

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Wind
M2Server app src main java com example hp
activity_main.xml strings.xml MainActivity.java Message
MainActivity send()
4: Run TODO 6: Logcat Android Profiler Terminal
11:14 PM
M2Server
SEND

Enter IP Address

Enter Text

Security Key

Information

(i) You must be connected same router network.
(ii) Then you enter your ip address and text.
(iii) Then click the send button.
(iv) You must be open your pc socket program.
(v) This app work only same network, same port address.

Create by @Arif, @Harun, @Riaz
Supervised by @Al-Amin Nipu, @Asraf Kamal (WUB)

192.168.1.105

Hello, Everyone.

126111114795261581416 13

Information

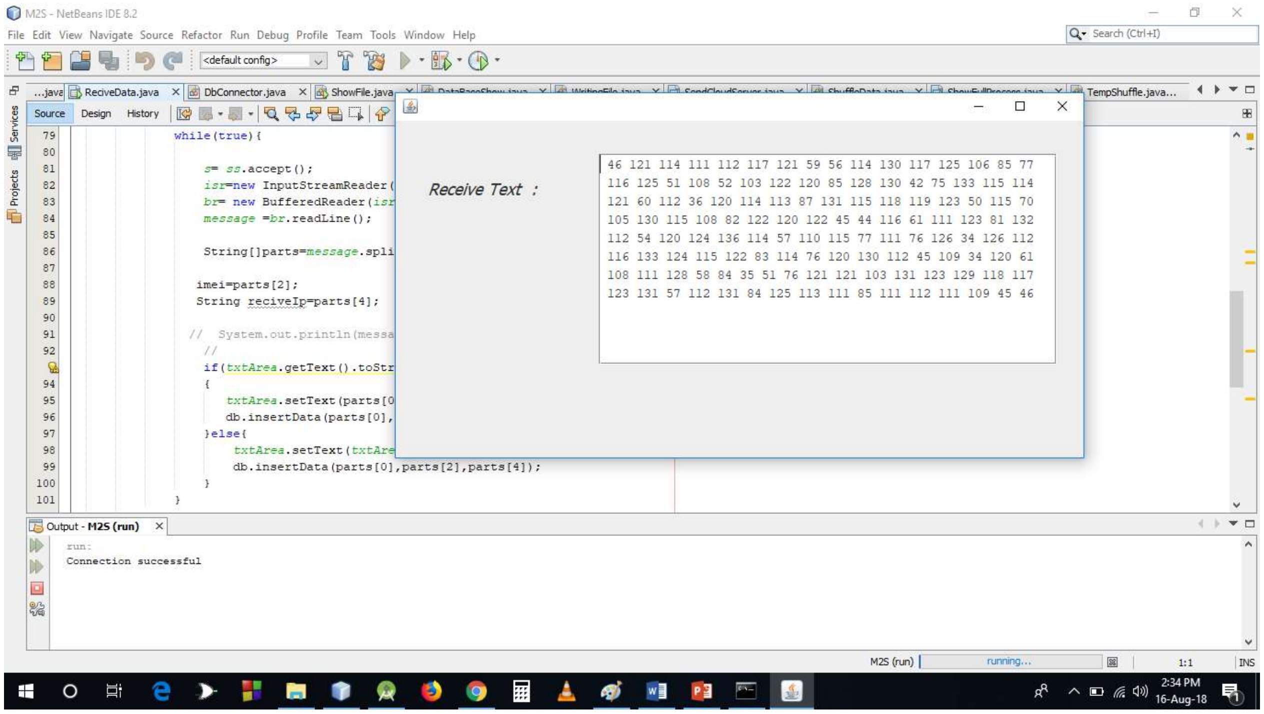
(i) You must be connected same router network.
(ii) Then you enter your ip address and text.
(iii) Then click the send button.
(iv) You must be open your pc socket program.
(v) This app work only same network, same port address.

Create by @Arif, @Harun, @Riaz
Supervised by @Al-Amin Nipu, @Asraf Kamal (WUB)

Figure 5.1: Create Encrypted data and Key

5.1.3 Receiving Text by Intermediate Server

When IoT or sending devices send data to intermediate server then intermediate server receive the text. Using java socket programing we receive the text.



```
        while(true){

            s= ss.accept();
            isr=new InputStreamReader(
            br= new BufferedReader(isr
            message =br.readLine();

            String[]parts=message.spli

           imei=parts[2];
           String reciveIp=parts[4];

         //  System.out.println(messa
            //
            if(txtArea.getText().toStr
            {
               txtArea.setText(parts[0
               db.insertData(parts[0],
            }else{
                txtArea.setText(txtAre
                db.insertData(parts[0],parts[2],parts[4]);
            }
        }
```

Receive Text :

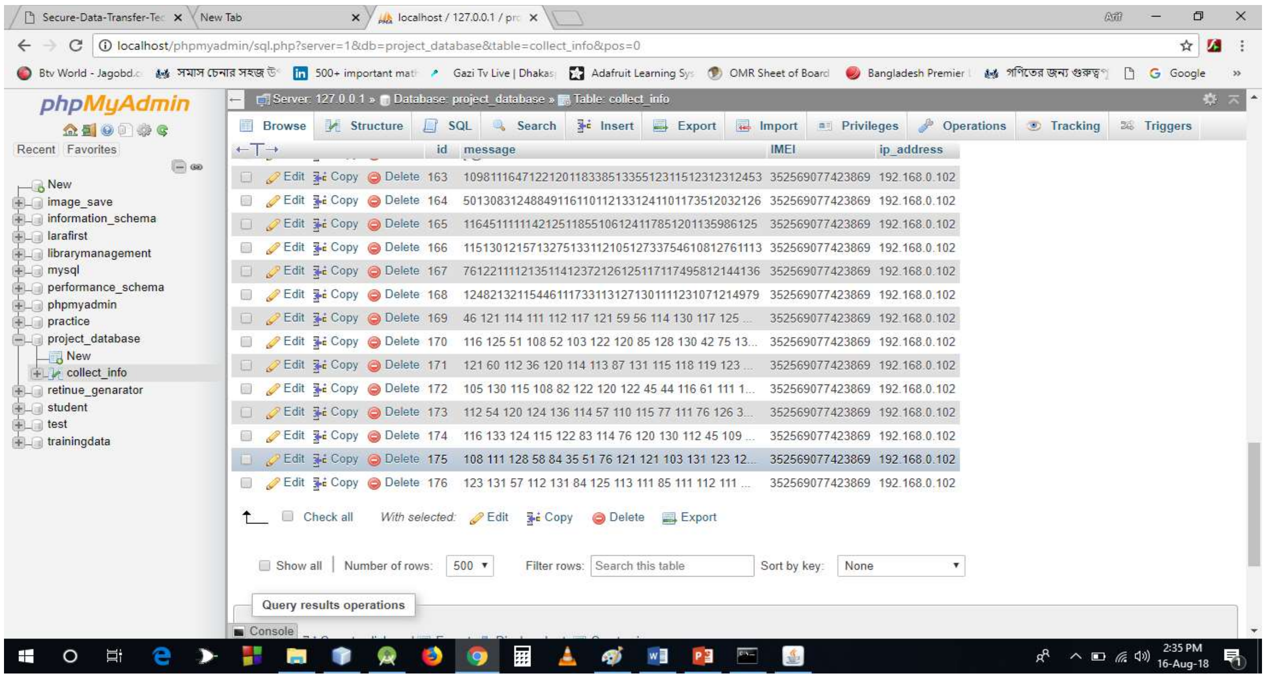
```
46 121 114 111 112 117 121 59 56 114 130 117 125 106 85 77
116 125 51 108 52 103 122 120 85 128 130 42 75 133 115 114
121 60 112 36 120 114 113 87 131 115 118 119 123 50 115 70
105 130 115 108 82 122 120 122 45 44 116 61 111 123 81 132
112 54 120 124 136 114 57 110 115 77 111 76 126 34 126 112
116 133 124 115 122 83 114 76 120 130 112 45 109 34 120 61
108 111 128 58 84 35 51 76 121 121 103 131 123 129 118 117
123 131 57 112 131 84 125 113 111 85 111 112 111 109 45 46
```

```
run:
Connection successful
```

Figure 5.2: Receiving Text by Intermediate Server

5.1.4 Store the data by Intermediate Server

When intermediate server receive the text then it will store the data.



localhost/phpmyadmin/sql.php?server=1&db=project_database&table=collect_info&pos=0

Server: 127.0.0.1 » Database: project_database » Table: collect_info

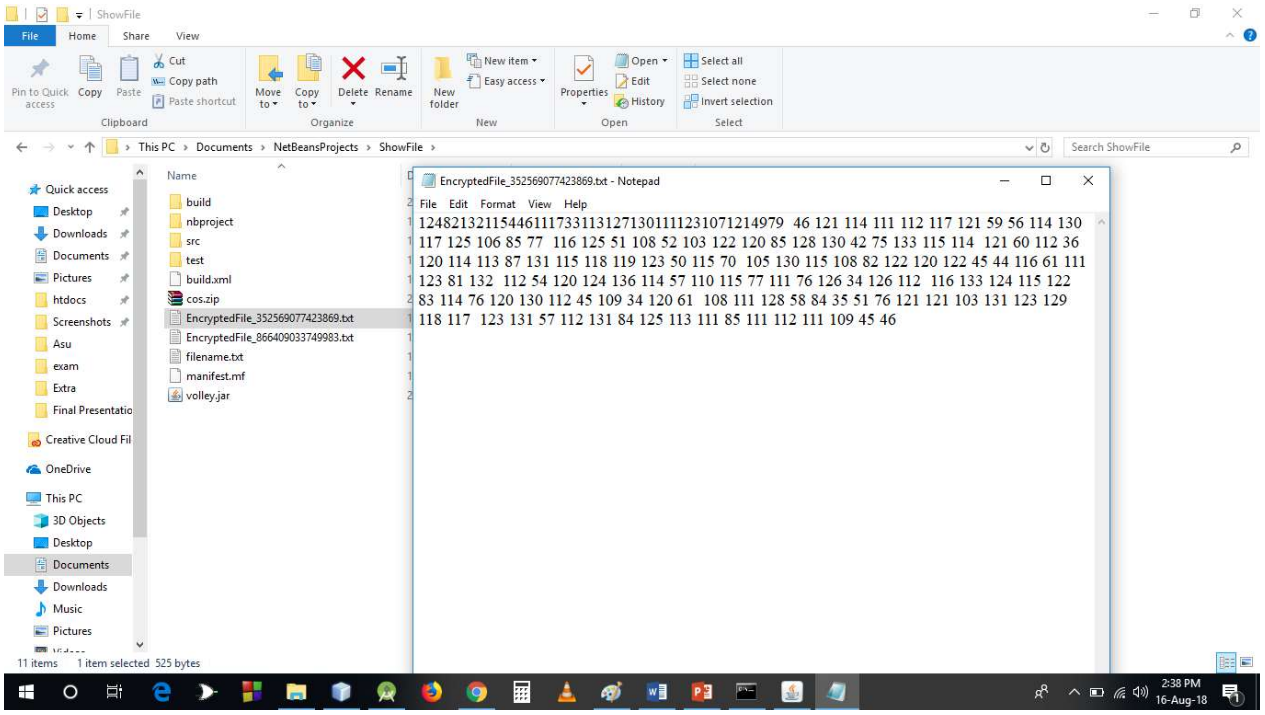
				id	message	IMEI	ip_address
☐	Edit	Copy	Delete	163	10981116471221201183385133551231151231231 2453	352569077423869	192.168.0.102
☐	Edit	Copy	Delete	164	50130831248849116110112133124110117351203 2126	352569077423869	192.168.0.102
☐	Edit	Copy	Delete	165	1164511111421251185510612411785120113598 6125	352569077423869	192.168.0.102
☐	Edit	Copy	Delete	166	11513012157132751331121051273375461081276 1113	352569077423869	192.168.0.102
☐	Edit	Copy	Delete	167	76122111121351141237212612511711749581214 4136	352569077423869	192.168.0.102
☐	Edit	Copy	Delete	168	12482132115446111733113127130111231071214 979	352569077423869	192.168.0.102
☐	Edit	Copy	Delete	169	46 121 114 111 112 117 121 59 56 114 130 117 125 ...	352569077423869	192.168.0.102
☐	Edit	Copy	Delete	170	116 125 51 108 52 103 122 120 85 128 130 42 75 13...	352569077423869	192.168.0.102
☐	Edit	Copy	Delete	171	121 60 112 36 120 114 113 87 131 115 118 119 123 ...	352569077423869	192.168.0.102
☐	Edit	Copy	Delete	172	105 130 115 108 82 122 120 122 45 44 116 61 111 1...	352569077423869	192.168.0.102
☐	Edit	Copy	Delete	173	112 54 120 124 136 114 57 110 115 77 111 76 126 3...	352569077423869	192.168.0.102
☐	Edit	Copy	Delete	174	116 133 124 115 122 83 114 76 120 130 112 45 109 ...	352569077423869	192.168.0.102
☐	Edit	Copy	Delete	175	108 111 128 58 84 35 51 76 121 121 103 131 123 12...	352569077423869	192.168.0.102
☐	Edit	Copy	Delete	176	123 131 57 112 131 84 125 113 111 85 111 112 111 ...	352569077423869	192.168.0.102

Number of rows: 500 Filter rows: Search this table Sort by key: None

Figure 5.3: Store the data by intermediate server

5.1.5 Create File by Intermediate Server

After storing intermediate server create a file based on sending device IMEI number.



```
EncryptedFile_352569077423869.txt - Notepad
12482132115446111733113127130111123107121497 9  46 121 114 111 112 117 121 59 56 114 130
117 125 106 85 77  116 125 51 108 52 103 122 120 85 128 130 42 75 133 115 114  121 60 112 36
120 114 113 87 131 115 118 119 123 50 115 70  105 130 115 108 82 122 120 122 45 44 116 61 111
123 81 132  112 54 120 124 136 114 57 110 115 77 111 76 126 34 126 112  116 133 124 115 122
83 114 76 120 130 112 45 109 34 120 61  108 111 128 58 84 35 51 76 121 121 103 131 123 129
118 117  123 131 57 112 131 84 125 113 111 85 111 112 111 109 45 46
```

Figure 5.4 Create file by intermediate server

5.2.1 Send File to Final Server

After creating file intermediate server send this file to final server which is cloud server. This file contain encrypted jumble data.

5.2.2 Final Server Store the File and Spilt the File

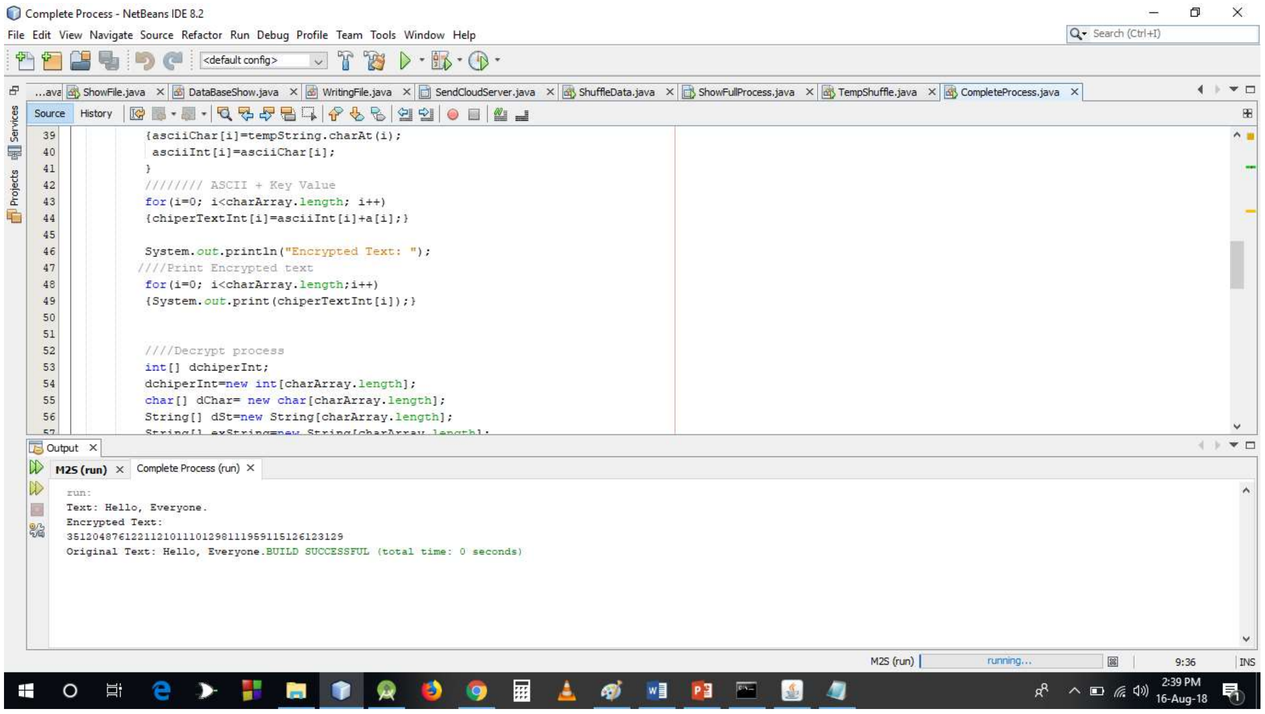
When intermediate server send the file then final server receive the file and store the file. Final server split the file.

5.2.3 Final Server Send the Split File and Send to User

When any user device request for any file then final server send to the user. By final server request sender device send the key to the user device.

5.3 Receive the File by User Device and Get Final Result or Decrypted data

When user device receive the file and key then it decrypt the file data. Then user device get original massage.



```
{asciiChar[i]=tempString.charAt(i);
 asciiInt[i]=asciiChar[i];
}
//////// ASCII + Key Value
for(i=0; i<charArray.length; i++)
{chiperTextInt[i]=asciiInt[i]+a[i];}

System.out.println("Encrypted Text: ");
////Print Encrypted text
for(i=0; i<charArray.length;i++)
{System.out.print(chiperTextInt[i]);}


////Decrypt process
int[] dchiperInt;
dchiperInt=new int[charArray.length];
char[] dChar= new char[charArray.length];
String[] dSt=new String[charArray.length];
```

```
run:
Text: Hello, Everyone.
Encrypted Text:
3512048761221121011101298111959115126123129
Original Text: Hello, Everyone.BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure: 5.5 Decrypt process

5.4 Testing Result of Various Encryption Algorithm

DES	AES	Proposed Algorithm
Low configuration device not supported.	Low configuration device not supported.	Our proposed algorithm is fully feasible for intensive devices.
Need to generate private and public key.	Need to generate private and public key.	No need to generate private and public key.
Secure for high configure devices.	Secure for high configure devices.	High secure in high and low configure devices
Time complexity more than AES.	Time complexity less than DES.	Time complexity more than AES.

CHAPTER 6 CONCLUSION

6.1 Conclusion

IoT based devices security are a big challenge. We have focused on encryption and existing algorithms are not suitable for these devices. In this paper, we have proposed a secure transmission technique between wearable devices. It removes all the difficulties of an Encryption algorithm. Our proposed algorithm is fast and light weight. Our proposed algorithm high secure in high and low configure devices. It is feasible for lot devices. We have implemented the algorithm and collects data for different cases. We compare or data with the existing algorithms and found it performs well. In future, we will add some more security features which will make the algorithm more immune to the security threats.

6.2 Future Work

Although our algorithm provides security to low configure smart devices, there is always a way to do better. We use simple bit shuffling technology in these devices and sending data to the intermediate server, but there is no trust has been made between intermediate server and sender and as well intermediate server to server. So, if we can use some methodology to verify input devices and intermediate server before sending and receiving the message, it will be much difficult to even access these shuffled data i.e. channel security.

We have implemented this algorithm only on the text file and jumble into a character array basis. In Future, we will implement this algorithm on all types of file or data.

To make it stronger, we can use multiple keys which can be used in rotation in time fashion. So that even hacker is able to get key, they can only extract some time data.

To make data more unpredictable, we can add some extra random bits in generated data then encrypt it, and make reverse operation in decryption part.

REFERENCES

C. Doukas, I. Maglogiannis, V. Koufi, F. Malamateniou, and G. Vassilacopoulos, “Enabling data protection through pki encryption in iot m-health devices,” in Bioinformatics & Bioengineering (BIBE), 2012 IEEE 12th International Conference on, pp. 25–29, IEEE, 2012. 16

C. Gehrmann and L. Barriga, “Indirect public-key encryption,” Aug. 17 2004. US Patent 6,779,111. 2, 15

D. L. Thompson, “Variable encryption scheme for data transfer between medical devices and related data management systems,” Sept. 16 2003. US Patent 6,622,050. 16

H. Wang, H. Zheng, B. Hu, and H. Tang, “Improved lightweight encryption algorithm based on optimized s-box,” in Computational and Information Sciences (ICCIS), 2013 Fifth International Conference on, pp. 734–737, IEEE, 2013. 1, 14

O. B. Sahoo, D. K. Kole, and H. Rahaman, “An optimized s-box for advanced encryption standard (aes) design,” in Advances in Computing and Communications (ICACC), 2012 International Conference on, pp. 154–157, IEEE, 2012. 1, 15

R. Snader, R. Kravets, and A. F. Harris III, “Cryptocop: Lightweight, energyefficient encryption and privacy for wearable devices,” in Proceedings of the 2016 Workshop on Wearable Systems and Applications, pp. 7–12, ACM, 2016. 17

S. T. Kofuji et al., “Performance analysis of encryption algorithms on mobile devices,” in Security Technology (ICCST), 2013 47th International Carnahan Conference on, pp. 1–6, IEEE, 2013. 15

T. Mills, M. Burnside, J. Ankcorn, and S. Devadas, “A proxy-based architecture for secure networked wearable devices,” 2001. 16

Y. Yan and T. Shu, “Energy-efficient in-network encryption/decryption for wireless body area sensor networks,” in Global Communications Conference (GLOBECOM), 2014 IEEE, pp. 2442–2447, IEEE, 2014. 16

5 Common Encryption Algorithms and the Unbreakable of the Future. www.storagecraft.com/blog/5-common-encryption-algorithms, 31 Jun 2017