

# Securing IoT Devices with In-Hop Encryption



By

**Arslan Riaz**

FALL-2018-MS-IS 00000275246 SEECS

Supervisor

**Dr. Syed Taha Ali**


A thesis submitted in partial fulfillment of the requirements for the  
degree of Masters of Science in Information Security (MS IS)

School of Electrical Engineering and Computer Science,  
National University of Sciences and Technology (NUST), Islamabad,  
Pakistan.

(July 2022)

## THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS/MPhil thesis entitled "Securing IoT devices with In-hop encryption" written by Arslan Riaz, (Registration No 00000275246), of SEecs has been vetted by the undersigned, found complete in all respects as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS/M Phil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

Signature: \_\_\_\_\_  \_\_\_\_\_

Name of Advisor: Dr. Syed Taha Ali \_\_\_\_\_

Date: 28-Jun-2022 \_\_\_\_\_

HoD/Associate Dean: \_\_\_\_\_

Date: \_\_\_\_\_

Signature (Dean/Principal): \_\_\_\_\_

Date: \_\_\_\_\_

## Approval

It is certified that the contents and form of the thesis entitled "Securing IoT devices with In-hop encryption" submitted by Arslan Riaz have been found satisfactory for the requirement of the degree

Advisor : Dr. Syed Taha Ali

Signature:  \_\_\_\_\_


Date: 28-Jun-2022

Committee Member 1: Dr. Wajahat Hussain

Signature: 

28-Jun-2022

Committee Member 2: Mr. Muhammad Imran  
Abeel

Signature:  \_\_\_\_\_

Date: 28-Jun-2022

Signature: \_\_\_\_\_

Date: \_\_\_\_\_


# Dedication

To my parents, my wife, daughters, my friends and my supervisor who has always been supportive.

## Certificate of Originality

I hereby declare that this submission titled "Securing IoT devices with In-hop encryption" is my own work. To the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEecs or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEecs or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics, which has been acknowledged. I also verified the originality of contents through plagiarism software.

Student Name: Arslan Riaz

Student Signature:  \_\_\_\_\_

# Acknowledgements

I am extremely thankful to Allah Almighty and my family who supported me and encouraged me. I would like to express immense gratitude to my supervisor Dr. Syed Taha Ali whose guidance helped me in all the time of research and writing the thesis. I am also very much grateful to my GEC members. I was very lucky that I could work on my thesis at SEECS-NUST. I want to thank Dr. Syed Taha Ali on his generosity in sharing precious thoughts and knowledge with me.

# Contents

<b>1. Introduction.....</b>	<b>1</b>
1.1. Motivation.....	2
1.2. Problem Statement.....	2
1.3. Objectives and Research Goals .....	3
1.4. Thesis Organization.....	4
<b>2. Background Information .....</b>	<b>6</b>
2.1. Architecture of IoT .....	6
2.2. IoT Communication Protocols .....	7
2.3. Security Importance of Internet/Network Layer in IoT .....	9
2.3.1. Information Eavesdropping .....	9
2.3.2. Man-in-the-Middle.....	10
2.3.3. Denial of Service (DoS).....	10
2.4. Achieving Network Security through Encryption.....	10
2.4.1. Encryption Types.....	11
2.4.2. Symmetric key Encryption Algorithms .....	11
2.5. OpenWrt .....	12
2.6. Iptables .....	12
<b>3. Literature Review .....</b>	<b>13</b>
3.1. Network Security Solutions for IoT .....	13
3.2. Proposed Encryption Solutions for IoT .....	16
<b>4. Research Methodology .....</b>	<b>22</b>
4.1. Problem Overview .....	22

4.2.	Proposed Solution.....	22
4.3.	Considerations .....	23
4.4.	Infrastructure .....	23
4.5.	Design and Architecture .....	23
4.5.1.	Policy Management.....	24
4.6.	User Space .....	25
4.6.1.	Custom Module.....	26
4.6.2.	Encryption Algorithms .....	30
4.6.3.	Kernel Space.....	31
4.6.4.	Interaction Between Kernel and User Space.....	32
4.7.	Packet Flow .....	33
<b>5.</b>	<b>Development and Implementation .....</b>	<b>35</b>
5.1.	Development.....	35
5.2.	Packages .....	35
5.2.1.	Cross Compiling Custom Module for OpenWrt .....	36
5.2.2.	Web Development Interface for Policy Management.....	37
5.3.	Implementation for Online Chat Application .....	38
5.4.	Implementation Verification.....	40
<b>6.</b>	<b>Evaluation of Research Work.....</b>	<b>43</b>
6.1.	Performance Analysis.....	43
6.2.	Throughput.....	43
6.2.1.1.	Plain Mode Throughput (without encryption).....	44
6.2.1.2.	Encrypted Mode Throughput .....	45
6.2.1.3.	Latency .....	47



6.3.	Comparison with Existing Solution .....	48
<b>7.</b>	<b>Conclusion and Future Work .....</b>	<b>50</b>
7.1.	Conclusion .....	50
7.2.	Future Work .....	50
<b>8.</b>	<b>Bibliography .....</b>	<b>52</b>

# List of Abbreviations and Symbols

## Abbreviations

IoT	Internet of Things
IP	Internet Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
Wifi	Wireless Fidelity
OpenWrt	Open Wireless Router
AES	Advanced Encryption Standard
VPN	Virtual Private Network

# List of Tables

Table 5-1 Network Configuration.....	39
Table 6-1 Routers Specifications .....	43
Table 6-2 PCs Specifications .....	44
Table 6-3 PCs Performance Comparison Table.....	49

# List of Figures

Figure 1-1 Research model .....	4
Figure 1-2 Thesis Organization .....	5
Figure 2-1 Basic IoT Layered Architecture .....	6
Figure 2-2 IoT Communication Protocols .....	8
Figure 2-3 Network layer communication .....	9
Figure 2-4 Encryption and Decryption Process .....	10
Figure 3-1 Architecture Diagram [28] .....	14
Figure 3-2 Overview of in-network security [29].....	15
Figure 3-3 IOT-NETSEC Workflow [31].....	16
Figure 3-4 HAN encryption algorithm usage in IOT [32].....	17
Figure 3-5 Proposed Model [33].....	18
Figure 3-6 Encryption and Decryption time Comparison [34].....	19
Figure 3-7 A VPN Network [39] .....	20
Figure 3-8 A Concept of interconnecting two VPN Gateways using Raspberry Pi devices [39].....	21
Figure 4-1 In-Hop Encryption .....	23
Figure 4-2 In-Hop Encryption Architecture .....	24
Figure 4-3 In-Hop Web Management.....	25
Figure 4-4 Iptables Rule.....	26
Figure 4-5 Obtaining Netfilter Queue Connection Handler .....	27
Figure 4-6 Binding Newly Created Queue .....	27
Figure 4-7 Handling of Incoming Data.....	28

Figure 4-8 Allocating Raw Data to Buffer .....	28
Figure 4-9 Payload Data extraction .....	29
Figure 4-10 Packet Data.....	29
Figure 4-11 Packet Data.....	30
Figure 4-12 Encrypt Payload .....	30
Figure 4-13 AES CTR Encryption Function .....	31
Figure 4-10 Interaction Between Kernel and User Space.....	33
Figure 4-11 Packet Flow.....	34
Figure 5-1 OpenWrt version.....	35
Figure 5-2 Packages.....	36
Figure 5-3 Build System Configuration.....	37
Figure 5-4 Cross Compiled Program .....	37
Figure 5-5 Status Web Page.....	38
Figure 5-6 Chat Application Scenario .....	39
Figure 5-7 NC server .....	40
Figure 5-8 Plain Mode .....	41
Figure 5-9 Encrypted Mode .....	41
Figure 5-10 Encrypted Mode.....	42
Figure 6-1 Throughput in Plain Mode .....	45
Figure 6-2 Throughput in Encrypted Mode (RC4).....	46
Figure 6-3 Throughput in Encrypted Mode (AES).....	46
Figure 6-4 Latency in plain Mode .....	47
Figure 6-5 Latency in Encrypted Mode (AES).....	48

Figure 6-6 Latency in Encrypted Mode (RC4) .....48

Figure 6-7 Throughput for IPsec VPN.....49

# Abstract

In this modern age, Internet of Things (IoT) is becoming part of our lives incautiously. The ease of monitoring and performing our tasks remotely has made this technology inescapable. Accessing IoT devices over the insecure network possess confidentiality threats to the private as well sensitive information. Communication can be intercepted in the absence of any proper access control and security mechanisms. Plain data traffic at the public network remains enticement for the hackers and bad actors. The problem with the conventional security methods is that they are not well suited for IoT environment. There are multitudinous factors involves which causes the security of IoT technology unattainable. So, there is need to introduce and implement a confidentiality control framework reconcilable for the IoT devices in the network. The solution is to scramble the information coming out the private IoT network, no plain traffic will pass through the public network. This solution is implemented to provide data confidentiality over the insecure media. In the implementation, custom build encryption module is deployed at the local router which encrypts the IoT network data traffic when rules are applied. The network traffic is segregated by the rules. At the client end, there is decryption module with the corresponding rules. The encrypted communication between client and server with the help of standard traffic capturing tool has been verified. The solution can be deployed at any standard OpenWrt firmware's router which is mostly used in the wireless routers. Some practical application test cases have also been demonstrated. The performance evaluation of the solution shows the minimum delay and lags with the sufficient network throughput in encryption mode.

# Chapter 1

## 1 Introduction

The Internet of Things (IoT) is a system of interconnected devices that can transfer data over the network. These devices communicate with each other in a small proximity or sometimes over a large area to achieve the desired goal. With the evolution of IoT devices over the past few years, the applications of IoT has also increased. Smart cities, smart homes, smart wearables, smart health system and smart energy meters are some promising examples of IoTs.

Devices connected to the internet have numerous threat vectors, moreover plain data communication is a high security risk. Because any intruder can easily intercept or manipulate it over the network. There are several vulnerabilities and incidents were reported in the past. According to 2020 Unit 42 IoT Threat Report 98% of IoT traffic is unencrypted [1]. This exposes confidential and personal data on the network.

The data remains trusted as long as it is in the private network but the moment it travels over the public network its confidentiality becomes a challenge. The IP cameras connected in the houses or offices provide live video feed over the internet, but the hacker can also get the access to attain private and sensitive information. According to “Bloomberg”, around 150000 IP cameras were hacked inside police departments, schools, hospitals and companies like Tesla and Equinox exposing data and their intellectual property [2]. Similarly, at the event of president Trump inauguration speech it was reported that about 70% of Washington DC IP surveillance cameras were attacked by the hackers [3].

The solution is required in similar cases where network traffic is in plaintext where hackers and attackers can get sensitive information.

There are different levels of security and also numerous challenges are involved when it comes to the IoT [4]. At the communication system (OSI model) each layer has its own protocols and security should be implemented at each level. The network layer is



responsible for connecting nodes and networks for transferring packets. The nature of IoT devices always make them to transfer data over the networks for transferring information. There is no standard in-built network security mechanism in the IoT networks. Researchers has proposed some communication security measures [5], [6], but there is need to protect the confidentiality at network level.

In this thesis, we will talk about achieving data confidentiality at network level with In-hop encryption. So, it is implemented at the network layer of communication system. Confidentiality can be achieved by encrypting the useful information so that the third party should be unable to understand the communication between legitimate users. In cryptography, encryption is a process which involves some mathematical operations to alter the intended information and later on decrypting it by the authorized party only to get the original information.

### **1.1. Motivation**

The internet connected things remain vulnerable to many types of attack. Many critical applications (i.e. surveillance IP cameras monitoring, alarm systems, household private devices, medical devices) where remote access and control by an intruder can cause confidentiality breach as well serious damage to the system. In the absence of suitable security controls this technology is taking place in our daily lives through home devices, office devices and our personal use devices. Hackers and attackers are taking advantage of vulnerabilities and gaining access to private and sensitive information. Moreover, devices been hacked can be used as botnets to launch greater attacks. So, there should be some solution for IoT to protect against eavesdropping and exposing private data to the public.

### **1.2. Problem Statement**

With the advancement of IoT technology the security challenges related to it are also increasing. Implementing IoT security becomes problematic due to some inherit characteristics of IoT networks and devices. IoT devices are diverse in nature for example they might be actuators, sensors, IP cameras, smart home devices or industrial nodes [7]. The protocols and standards with they are communicating with each other,

are also very versatile. The major hurdle in implementing elaborate security protocols or existing cryptographic algorithms is having constrained computational and memory resources in these devices. Further customized cryptographic solutions developed for one type of devices might not be suitable for other kind or at least there is no practical implementation available which is applicable to all kind of IoTs [8]. Adding extra hardware for security purpose is also not an economical solution. There should be some mechanism which should cater these problems in more efficient way.

### **1.3. Objectives and Research Goals**

Research goal of this thesis is to propose and implement the framework for the network security of IoT devices where confidentiality of private data can be ensured. The communication and sensitive information shared between users or client and server should be protected from attackers and malicious actors.

The In-hop encryption solution should be implemented in IoT network for real time applications having maximum throughput and minimum delay. Solution should not add any extra hardware instead it should be very flexible using existing infrastructure. Solution should also be user friendly to implement so it can be used most widely.

These following main objectives of this research are shown in figure 1-1:

- Goal-1: Implement In-hop encryption module
- Goal-2: Encrypt intended traffic with rules
- Goal-3: Encrypted traffic analysis
- Goal-4: Performance evaluation of proposed solutions

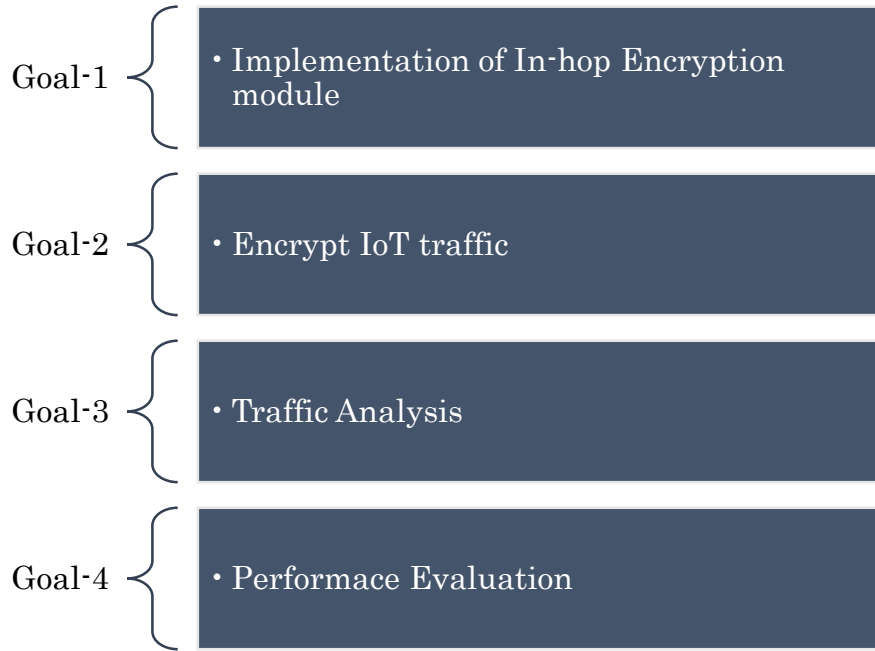


Figure 1-1 Research model

#### 1.4. Thesis Organization

In chapter 2, discussion and explanation related to research work has been done. This is helpful in understanding of In-hop Encryption design. In Chapter 3, literature review of existing and related solutions to this research problem has been discussed. Chapter 4 includes methodology and purposed solution used to solve the research problem. Architecture and complete design are explained. In chapter 5, proposed solutions is implemented with use cases of some practical applications. Chapter 6 is related to performance evaluation of the solution with respect to throughput and latency. Conclusion and future work are described in chapter 7. Thesis organization as described above is also shown in figure 1-2.

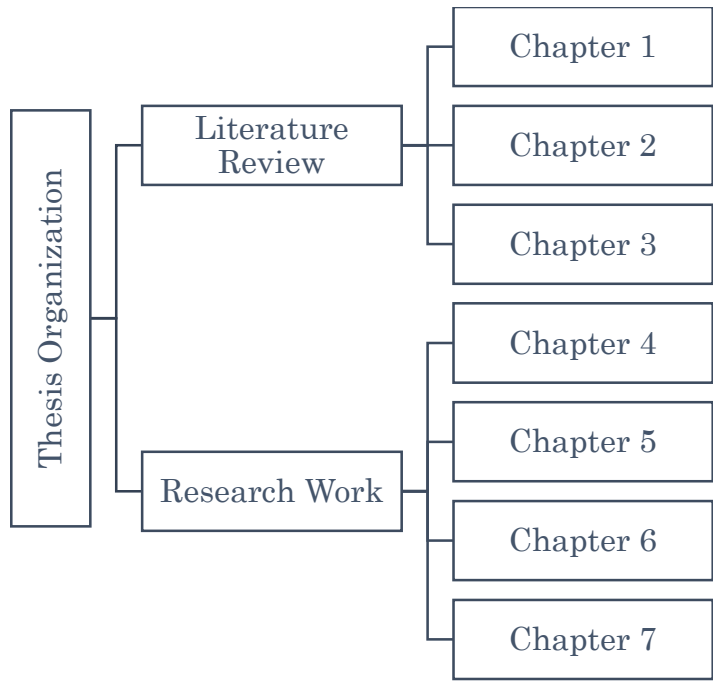


Figure 1-2 Thesis Organization

# Chapter 2

## 2 Background Information

For the understanding of this thesis, required background information is given in this chapter. Architecture of IoT, Communication protocols in IoT, Network layer importance in IoT, achieving network security through encryption is included.

### 2.1. Architecture of IoT

There is no standard architecture design for IoT. Many researchers have proposed layered approach like three-layer, five-layer and seven-layer architecture [9], [10]. Figure 2-1 represents the most basic model that is used widely:

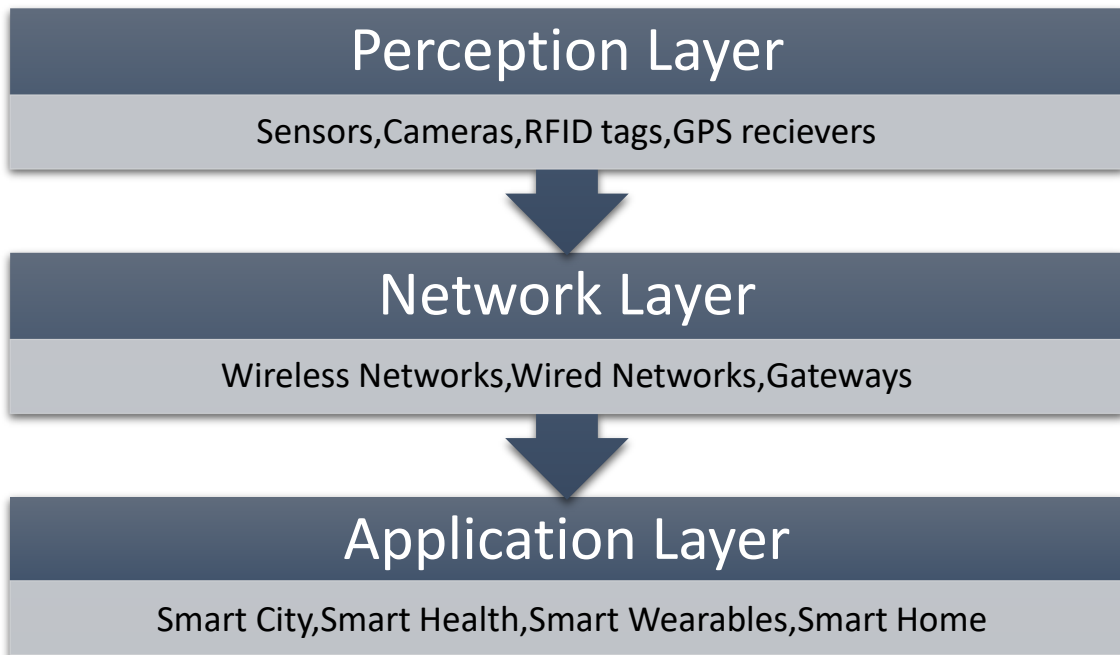


Figure 2-1 Basic IoT Layered Architecture

There are majorly three layers:

1. Perception Layer
2. Network Layer
3. Application Layer

The perception layer is where sensors, actuators, thermostats and other monitoring devices physically lies in the environment to sense and gather information. It basically identifies and intercept different parameters around the surroundings.

Network layer is an interface between application and perception layer. The connectivity and all the information collected by the IoT devices is shared and transferred under this layer. Network layer is also responsible for packet routing for internet connectivity.

Application layer provides application environment and services to the users. The data collected from IoT devices is formatted and presented by the application layer. The applications can be smart cities, smart health, smart homes and the communication protocols handling of these applications is also the responsibility of this layer [11] [12].

## **2.2. IoT Communication Protocols**

Communication is the most important part of IoT and protocols defines the rules which enable the connectivity and exchange of data between two or more nodes. Open System Interconnection is the standard model which allows implementation of interoperable networks communication at seven layers. And each layer has its own protocols to work with same layer of other devices and also handle the interfacing between upper and lower layers. TCP/IP model is the more concise version of OSI with four layers. Most of the applications of IoT are working on the principle of conventional communication layer models and also some new standards, protocols and models are in development phase that would be specific for IoT technology. IBM, CISCO and some other researchers have mapped IoT communication protocols with the TCP/IP model [13], [14]. Figure 2-2 shows the protocols of IoT against the TCP/IP layers.

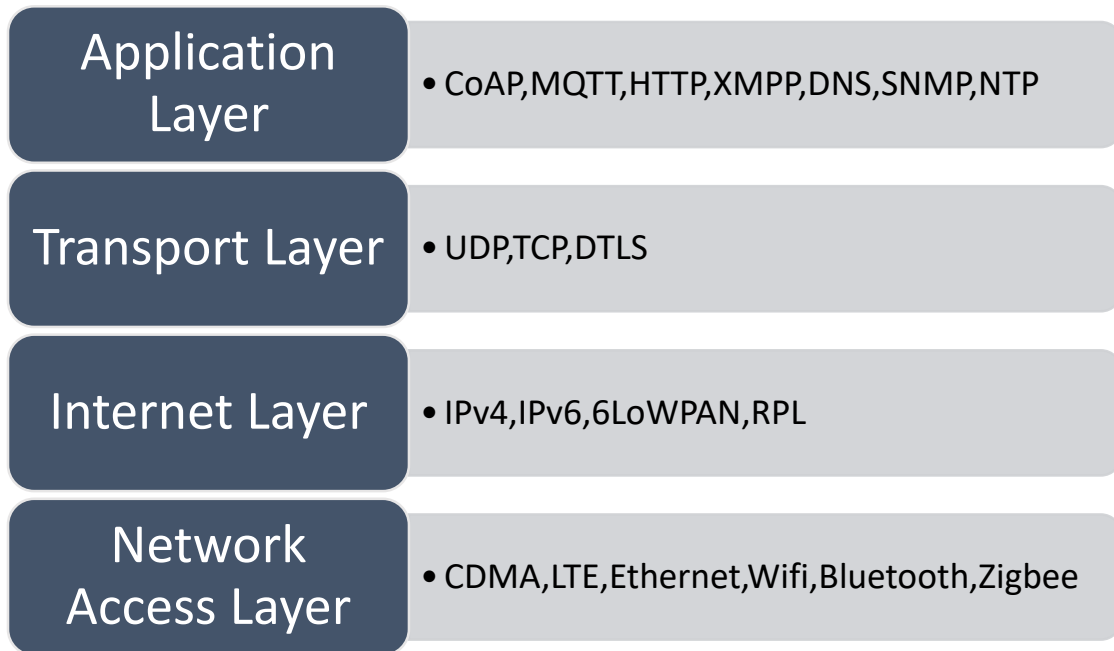


Figure 2-2 IoT Communication Protocols

Network access layer is responsible for the physical connectivity of IoT devices. Media could be wired (Ethernet) or wireless (WiFi, Cellular, Bluetooth, WiMax) depending on the environment.

The routing and addressing of data packets comes under the internet layer. Although other protocols have also been used in specific applications but Internet Protocol (IP) is the standard protocol at this layer and most of the devices are connected to the network via this protocol. IP addressing is the logical addressing at this layer and each device is assigned with an IP address to interact with each other. IPv4 and IPv6 are the Internet Protocol versions and later was introduced to have greater capacity of IP addresses. Other protocols like 6LoWPAN, 6Lo and IPv6 over Bluetooth Low Energy are also based on standard IPv6 protocol.

Despite the limited number of IP addresses (4.3 billion) in IPv4, this protocol has managed the issue with the private addresses and Network Address Translation (NAT). Range of private addresses has been defined that can be used without any conflict and in order to connect to the outer world router performed as a gateway to translate the private IP addresses into public IP address.

Transport layer protocols ensures the reliable end-end communication and session maintenance.

Application layer protocols provide interface for the users to monitor and control IoT devices. IoT uses some conventional protocols like HTTP and HTTPS. Furthermore, protocols like Representational State Transfer, Extensible Message and Presence Protocol and Constrained Application Protocol have also been developed for IoT devices.

### 2.3. Security Importance of Internet/Network Layer in IoT

Several security issues and challenges have been pointed out by the researchers at each level [15], [16], [17]. Although security should be a primary concern at each layer of IoT architecture, the network security problems can cause serious damage to private and sensitive information of the users. The communication at this layer is not only limited to the internal networks but connectivity among different networks and also across the internet, is the responsibility of this layer as shown in the Figure 2-3.

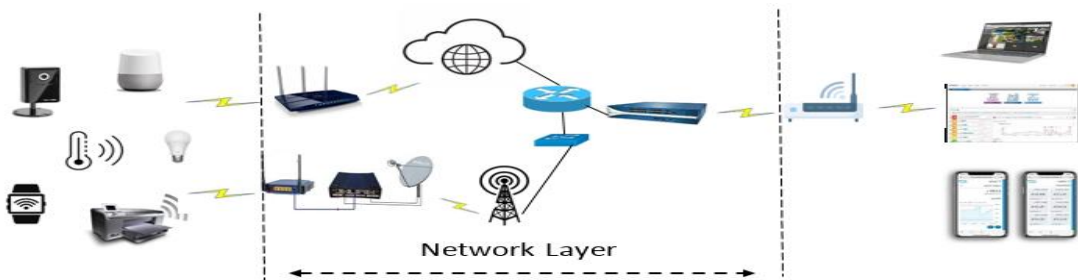


Figure 2-3 Network layer communication

When IoT data exchange and connectivity happens at public network it is vulnerable to many security threats. The most common security attacks at this layer are:

#### 2.3.1. Information Eavesdropping

Information eavesdropping is a common attack at network layer to compromise the data confidentiality and privacy. The main reason for this attack is due to remote access of IoT devices over the internet and external networks. Intruders can intercept the



exchanged information and can be used for criminal purposes. Protection of data is as important as the physical security of devices.

### 2.3.2. Man-in-the-Middle

Intruders and bad actors can perform man-in-the-middle attack by becoming part of the network from the middle. Sitting in the middle of the network and pretending to be a legitimate user, attacker can simply eavesdrop or manipulate the communication and compromise the integrity of the data.

### 2.3.3. Denial of Service (DoS)

IoT nodes have limited resources and less processing capabilities. Attackers from the outside perform flooding of application requests like DNS requests to consume the resources. Multiple connection requests handling may crash down the systems. It also involves network choking to make the services unavailable to the users or jamming wireless networks [18]. This attack is the compromise of availability in the information security triad.

## 2.4. Achieving Network Security through Encryption

Encryption falls under the domain of cryptography and it is a fundamental building block of network security [18]. Encryption is a procedure of converting plain text information into scrambled (unintelligible) form that nobody can understand. The encryption process involves some mathematical operations and a secret key. For reverting back, the information into its original form the process done is called decryption [19]. Figure 2-4 illustrates the encryption and decryption process.

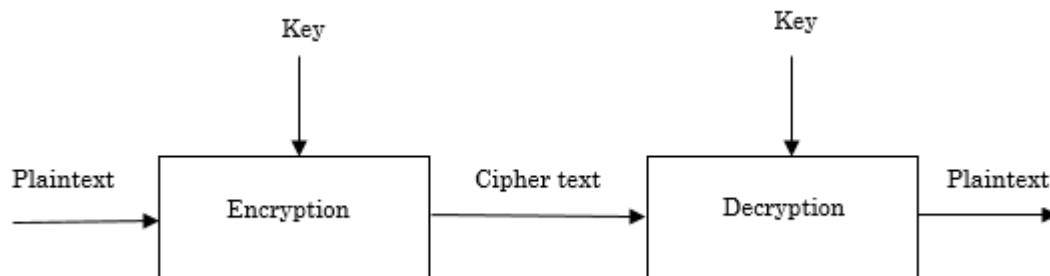


Figure 2-4 Encryption and Decryption Process

Encryption applies at both levels, data at rest (file encryption, disk encryption) and data in transit. At network layer data travels over the media and encryption can be applied to secure the communication. Encrypted traffic remains unreadable for the intruders and malicious actors. Number of attacks like eavesdropping at network layer can be avoided by applying this mechanism. By applying encryption, the most important security component confidentiality can be achieved to protect the privacy and sensitive information. Integrity can also be achieved by applying some other techniques like Message Authentication Code (MAC) along with the encryption. Network standard encryption protocols like IPsec also ensures the availability component among confidentiality and integrity of data.

#### **2.4.1. Encryption Types**

Various encryption techniques have been used in the past to achieve the secret communication [20], [21]. The contemporary encryption types are symmetric and asymmetric. Symmetric key encryption technique uses same key for encryption and decryption operations while asymmetric key encryption type uses different keys. Asymmetric key encryption type is also called public key encryption scheme where anybody can encrypt and only owner of private key can decrypt. Asymmetric key encryption is used where small amount of data is to be encrypted because its process is very slow whereas symmetric key encryption is used to encrypt the large amount of data very efficiently. Symmetric key encryption is always suitable at network layer where data is in bulk amount and this scheme uses less processing power and memory.

#### **2.4.2. Symmetric key Encryption Algorithms**

There are two variants of symmetric encryption [22];

1. Block Cipher
2. Stream Cipher

Block cipher encrypts the data in form of fixed size blocks operating on the fixed size blocks of input data. Block cipher uses multiple modes of operation to produce randomization of same input blocks [23]. Out of the many common encryption algorithms, Advanced Encryption System (AES) is known to be most invulnerable

against almost all types of attacks and also a standard algorithm approved by NIST in 2001.

Stream cipher encrypts the data bit by bit instead of blocks. Randomly generated key is XORed with the same length of plain text to produce ciphertext and similarly ciphertext is XORed with the key to produce original plain text [24]. RC4 is a popular symmetric key stream cipher.

## **2.5. OpenWrt**

Open Wireless Router (OpenWrt) is an open source Linux based operating system for embedded devices [25]. It was initially developed by Linksys and later on base source code was released to work on for different variants under open source community [26]. OpenWrt is an appropriate operating system for memory and space constrained routers like embedded devices. Web interface or command line interface (shell) can be used for configuration. OpenWrt OS comes with some basic routing and firewall features, to enhance the functionality there are lot of programs are available which can be installed. OpenWrt also provides toolchain and software development kit (SDK) for the development of custom modules. These custom modules can be easily installed on the routers using package manager (opkg).

## **2.6. Iptables**

Iptables is a Linux program that allows to configure rules for filtering IP (IPv4) packets thus acting as a firewall [27]. For handling of packets rules are defined under the chains and chains exists under the tables. There are five predefined tables (Raw, Mangle, NAT, Filter, Security), five chains (INPUT, OUTPUT, FORWARD, PREROUTING, POSTROUTING) and many targets (ACCEPT, DROP, REJECT etc.). Additional chains can also be created that exists under some table. Iptables rules are defined to perform specified actions on the packets mentioned in the target.

# Chapter 3

## 3 Literature Review

In this chapter, we have done review of research work related to IoT security at network layer and also cryptographic (encryption) solutions proposed to secure IoT. Many researchers have discussed IoT security challenges and solutions. Existing shortcomings to achieve required goals have been also discussed in this chapter.

### 3.1. Network Security Solutions for IoT

IoT is all about connectivity and communication, and network layer is responsible for this task. Compromise at network security may collapse the whole IoT system. While designing the network in any IoT environment its security should also be given a great importance. IoT is a diverse technology and no standardization has been done yet regarding its protocols and security controls. Researchers have identified many challenges and also specified different solutions to implement network level security for IoT.

Anna Kornfeld, Franziska Roesner and Tadyoshi Kohno presented the solution for security of vulnerable home IoT devices with a security manager as shown in Figure 3-1 [28]. Their proposed solution is developed at gateway router and will intercept all incoming and outgoing traffic. This solution gives status of all IoT devices and can report their vulnerabilities. At the time of need security manager could take action to diminish security risks. The solution has four modules that identify vulnerable devices, acquire the patch from internet, apply the patch and limit the traffic rate by finding any anomalies. The solution has also the feature of authentication of devices.

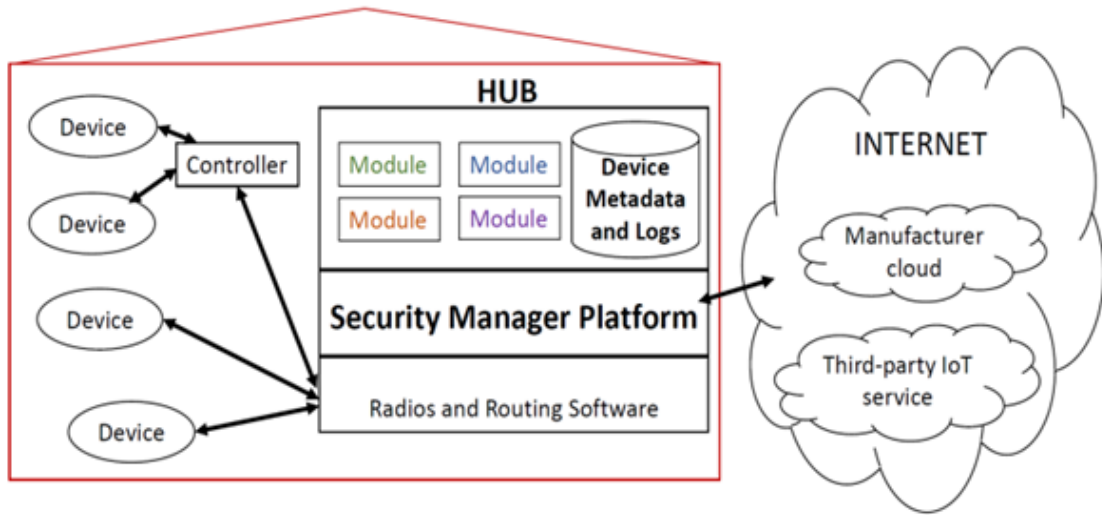


Figure 3-1 Architecture Diagram [28]

Martin Serror, Martin Henze, Sacha Hack and Marko Schuba [29] proposed a solution in which router acts as a central device where traffic rules work with the communication filtering and anomaly detection techniques to alleviate the security concerns. Figure 3-2 [29] shows that the solution has three components that are specification of network compliant behavior, IoT traffic filtering and anomaly detection. Specification of network compliant behavior is defining the rules of IP address, port number and communication direction based on the known legitimate network traffic. IoT traffic filtering is enforcement of defined rules through Software-Defined Networking approach using static and dynamic matching that are basically two techniques used in the SDN deployments. The above components are based on the pre-defined rules, to handle the anomalies in the network two techniques were proposed. First is to observe the unknown data traffic flow to monitor and compare the traffic pattern against malicious behavior. Second approach proposed for anomaly detection is the use of machine learning technique.

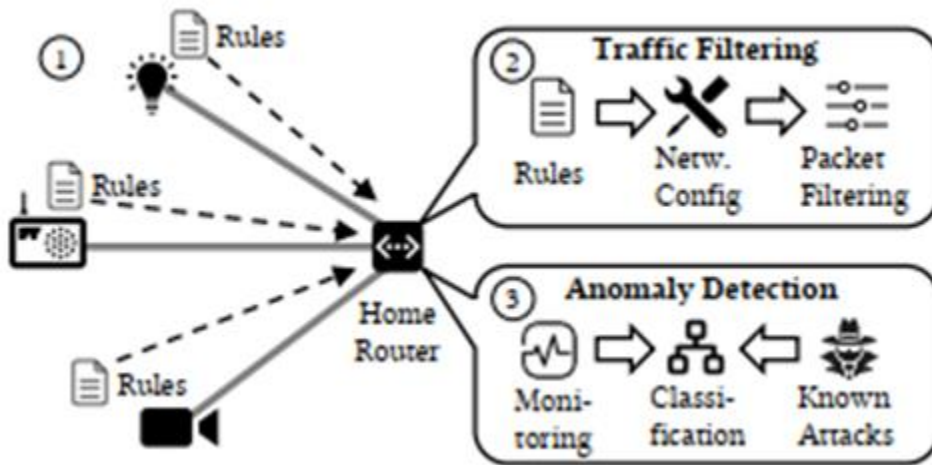


Figure 3-2 Overview of in-network security [29]

Aneesh Dua, Vibhor Tyagi, ND Patel and BM Mehtre [30] proposed IDRBT-IoT Secure Router (IISR) to protect against the number of cyber-attacks. In their demonstration they conducted attacks and out of ten seven were detected and mitigated successfully. They used Raspberry Pi as a gateway router and install snort on it to detect and prevent intrusions.

Mahdi Nobakht, Craig Russell, Wen Hu, Aruna Seneviratne [31] considered the existing general purpose SDN security solutions impractical due to diversity and bulk amount of network traffic therefore proposed their IOT-NETSEC framework. They integrated their prototype with SDN controller and demonstrated three network attacks to show the feasibility of the IOT-NETSEC solution. They adopted policy-based approach to monitor all the network traffic and limit traffic rate up to the threshold that is defined in the security policy. Users register their IoT devices and define security policies that specifies the threshold level of traffic flow associated to the devices. IOT-NETSEC monitors both incoming and outgoing traffic flows. The software has also the capability to block the access to the IoT device that violates the security policy. To define the policies, the users should have good understanding of traffic flows related to the IoT devices. Figure 3-3 [31] shows the workflow of IOT-NETSEC.

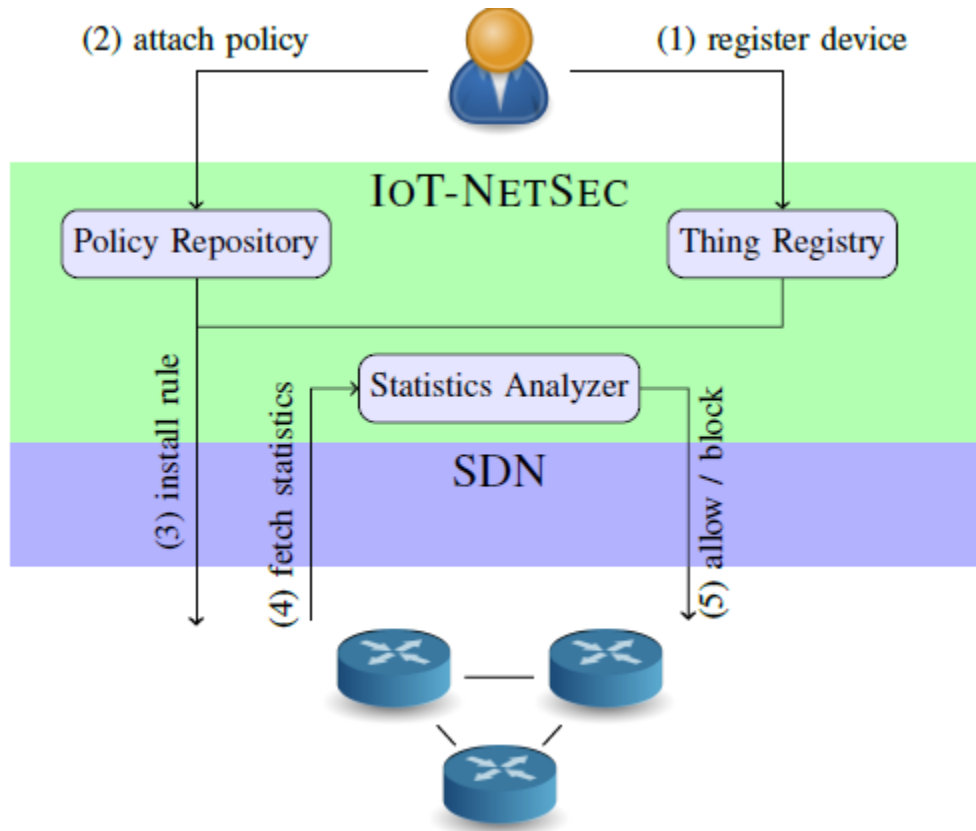


Figure 3-3 IOT-NETSEC Workflow [31]

The researchers in their above-mentioned works have proposed different techniques and solution to mitigate and reduce network attacks and threats. Like in [28] patching mechanism to remove vulnerable devices in the IoT network along with network traffic rate limitation module is proposed. The researchers proposed anomaly detection framework in the network [29]. The research work of [30] presented a router that can detect and prevent some network attacks. The above all proposals somehow reduce security risks but they all lack in achieving the most important network security element that is confidentiality. Implementing security controls without confidentiality exposes private and sensitive information to the public networks. Attackers not only attempts Denial of Service attacks to disrupt the services but they also intercept communication for the use of their interests.

### 3.2. Proposed Encryption Solutions for IoT

Most of the encryption proposals and solutions for the security of IoT are related to the improvement, analysis, suggestion or new design of encryption algorithms.

Amirhossein Safi [32] suggested the use of hybrid encryption algorithm for IoT which gives strong security with low computation. A suggested hybrid encryption algorithm HAN was evaluated in MATLAB and compared with the standard algorithms like AES and RSA. This technique guarantees integrity, confidentiality and non-repudiation of the information exchanged in the IoT. HAN algorithm is the combination of symmetric and asymmetric algorithm in which public key is produced by symmetric encryption and messages are encrypted by asymmetric encryption. Recipient owns the private key and decrypts the messages encrypted by the sender. Figure 3-4 shows the HAN encryption algorithm usage in IOT. The use of digital signature is also recommended by the author to improve the security. The results show that suggested algorithm speed is much better than AES and RSA.

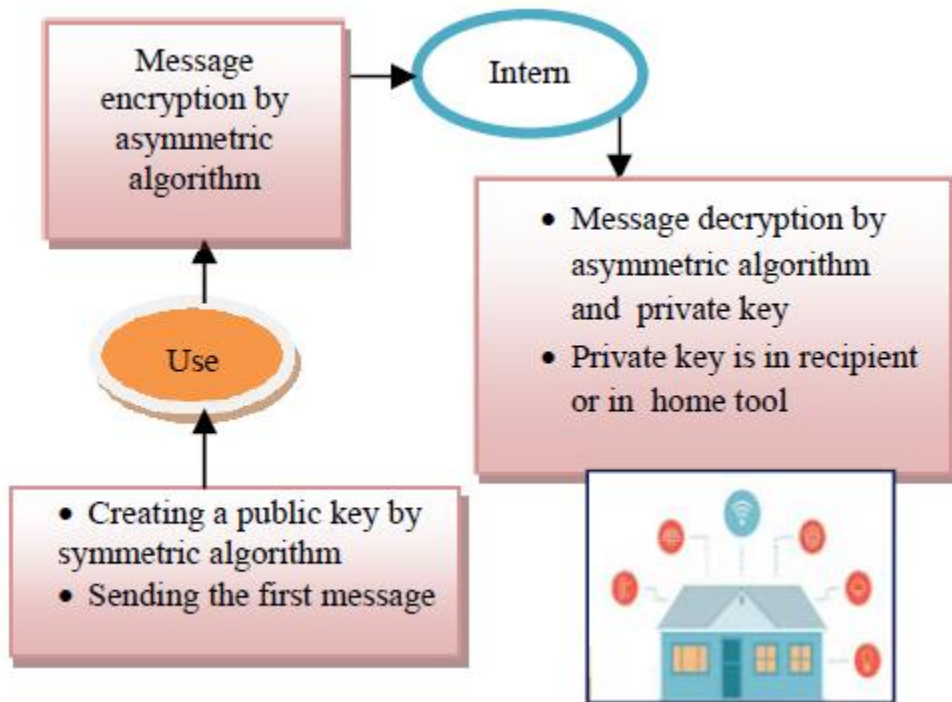


Figure 3-4 HAN encryption algorithm usage in IOT [32]

Rupesh Bhandari, Kirubanand V B [33] proposed an enhanced cryptographic solution for data transmission. The main goal of the proposed model is to establish a secure channel between IoT device and user device to transfer data ensuring the confidentiality, authenticity and integrity. This solution combines the characteristics of symmetric, asymmetric and public key server. Public key server conserve the public key database.



Proposed model shown in the Figure 3-5 depicts the whole process of keys generation, key exchange, encryption and decryption between two parties (IoT device and user device). The first step involves in registration of IoT devices and user devices by sending their MAC addresses and public keys to the public key server generated by elliptic curve cryptography. After that both devices will share their public keys over HTTPS along with their MAC addresses. Next both devices will verify themselves from the public key server. From the public keys both devices will agree upon the common secret key that will be used for the encryption and decryption using AES.

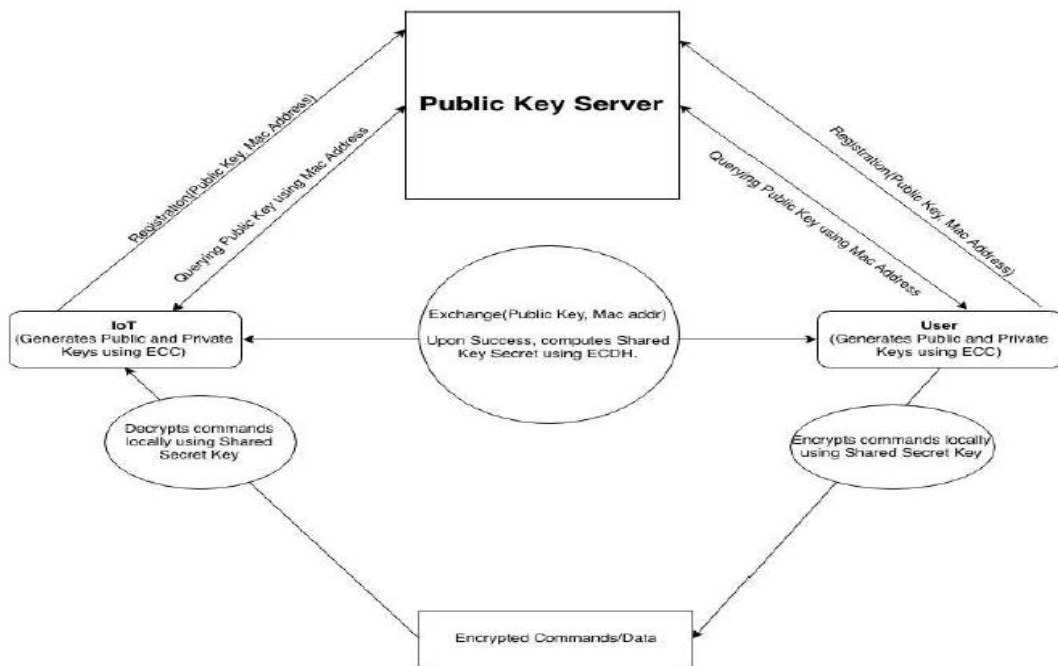


Figure 3-5 Proposed Model [33]

Iqra Hussain, Nitin Pandey and Mukesh Chandra Negi [34] proposed a symmetric key encryption algorithm based on binary bit sequence and XOR operation. The algorithm basically works on the binary bit sequence of plaintext and common secret key for encryption and decryption provided by the user. The process involves in;

- Converting the plaintext with corresponding ASCII values.
- XOR operation of values and the key.

- Converting numerical values and replacing it with the  $n/2$ -bit sequence.
- Converting obtained  $n/2$ -bit sequence with the decimal form.
- Cipher text is obtained in the last by changing decimal values with their character from the character table defined by the user.

Decryption algorithm works by replacing the cipher text by their numerical values, maintaining the binary bit sequence, doing XOR operation and converting the characters from character table values. Figure 3-6 [34] represents the comparison made between the proposed algorithm and the standard algorithms AES and RSA.

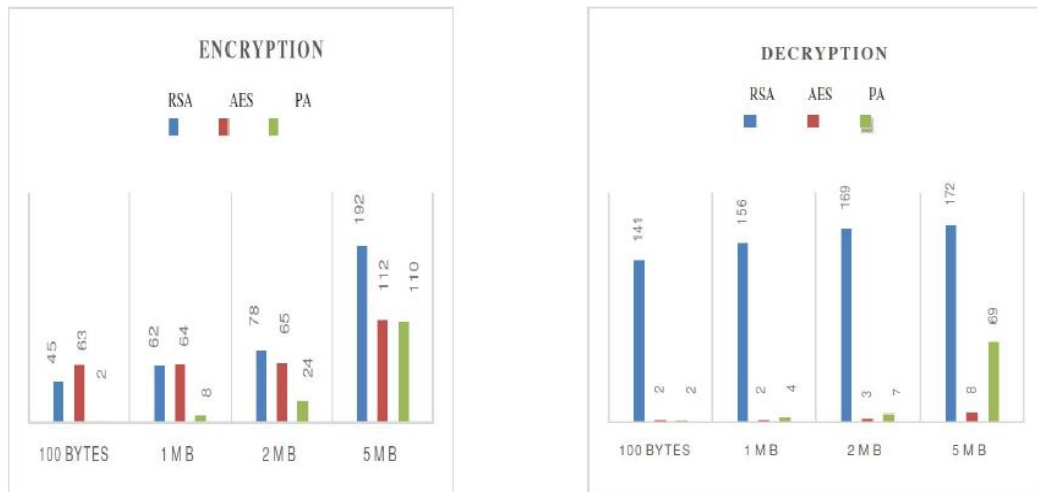


Figure 3-6 Encryption and Decryption time Comparison [34]

Considering resource constraint IoT technology many researchers have proposed customized and lightweight encryption algorithms [35], [36], [37]. Effy Raja Naru, Dr. Hemraj Saini and Mukesh Sharma [38] had reviewed many lightweight encryption techniques for the secure IoT communication. They stated that some encryption techniques require more memory with less computation and vice versa. They have also compared the pros and cons of the different lightweight encryption techniques.

Bogdan Jeliskoski, Biljana Stojcevska and Adrijan Bozinovski [39] presented a solution for securing the network of home IoT internet connected devices. Raspberry Pi device is used as a router and acts as a virtual private network (VPN) gateway for all the IoT devices in the home. VPN is a technology that provides connectivity between two or

more devices among different networks in such a manner that they virtually become the part of same private network. VPN provides a point to point tunnel like channel between two nodes that is transparent to other networks or nodes and encryption can also be applied to pass this tunnel encrypted traffic only.

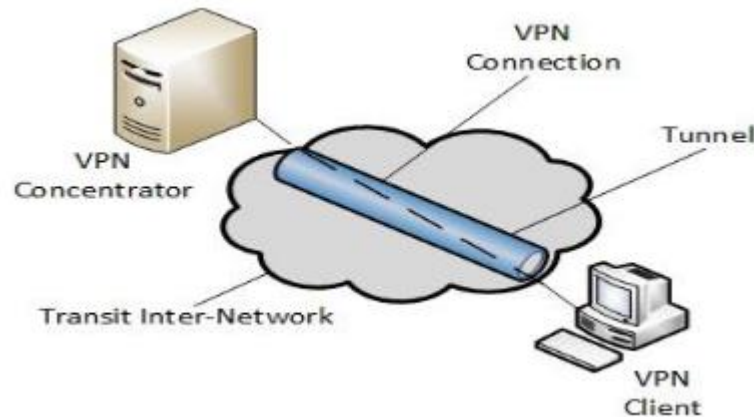


Figure 3-7 A VPN Network [39]

In this way VPN provides confidentiality through encryption along with authentication. In this research work OpenVPN system on raspberry Pi device (acting as a gateway) is implemented. OpenVPN implements client and server application. For encryption RSA-2048 algorithm is used and mutual authentication is achieved using PKI (public key infrastructure) that verify both server's and client's certificate. They have also recommended some other features that can strengthen the security like utilizing CRL (certificate revocation list), implementing DNS security, using 'dnsmasq' that enforces the DNS traffic to pass through the VPN, using IPTABLES to block all incoming and outgoing traffic in case of no VPN. And they suggested that it would be better to use site to site VPN, which establish VPN connection between gateways at both ends like shown in the figure 3-8 [39].

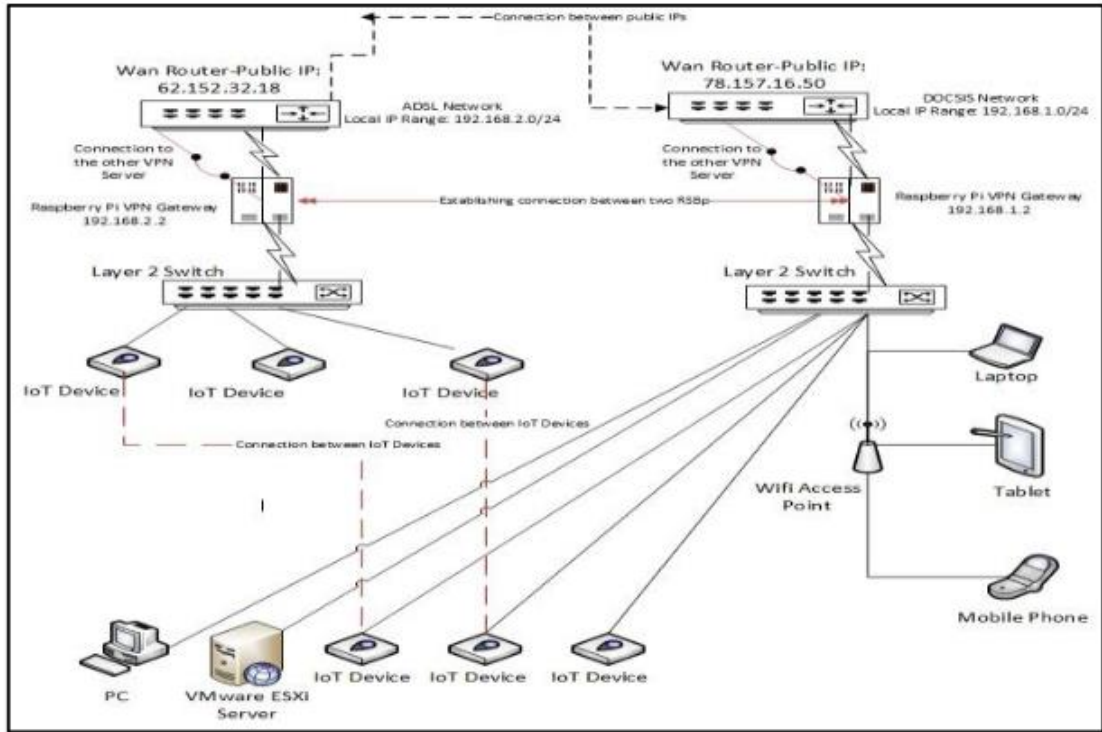


Figure 3-8 A Concept of interconnecting two VPN Gateways using Raspberry Pi devices [39]

# Chapter 4

## 4 Research Methodology

In this chapter, we have discussed the methodology of In-Hop encryption solution for securing the network of IoT devices.

### 4.1. Problem Overview

Network data confidentiality is an important security component in IoT network that protects the communication. Once communication channel is compromised, IoT devices and networks are prone to multiple attacks. Privacy and sensitive information leakage can have serious impact on the users. Plain data traffic at public networks leverage attackers to steal passwords and other credentials that can give access to the devices. Further devices can be accessed remotely to launch DDOS like attacks. Data confidentiality is exploited widely and served many purposes for intruders. So, there should be a mechanism to protect the confidentiality at network level.

### 4.2. Proposed Solution

In our solution, we have used symmetric key encryption mechanism at the gateway router where all IoT devices are connected. In Chapter 2, section 2.4 encryption and its type symmetric key encryption is explained in detailed. When user wants to access IoT service and send request, the response from the device reaches at the gateway router where encryption is applied to that traffic and encrypted traffic passes out from the router. The encrypted response reaches at the user end where decryption is applied first at the gateway router and then plaintext response is presented to the user. Figure 4-1 represents the proposed scenario where IoT devices are connected to the gateway router.

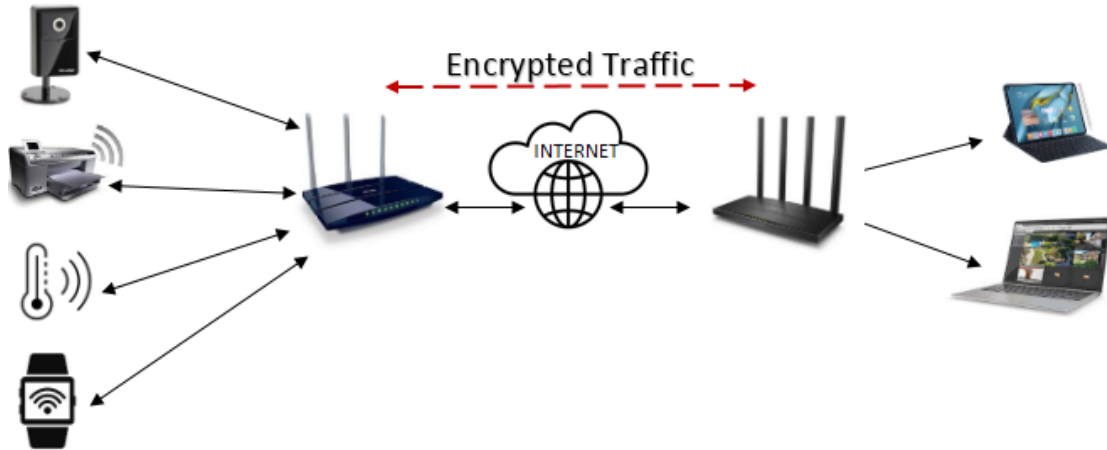


Figure 4-1 In-Hop Encryption

### 4.3. Considerations

As explained earlier (Chapter 1, section 1.3), the prime goal of our research work is to propose and implement network security solution for IoT environment. And design should have following considerations:

- Ensures the data confidentiality.
- Practical implementation of the design.
- Use of existing infrastructure instead of expensive or additional hardware.
- Lightweight package or program that can be installed by the users.
- User friendly solution.

### 4.4. Infrastructure

IoT three-layer architecture (Chapter 2, section 2.1) shows that IoT devices at the perception layer, when send responses to the outer world, they communicate through the gateway device. That's why In-Hop encryption solution is implemented at the gateway devices (routers) using existing infrastructure. The In-Hop encryption solution requires no extra hardware or resources.

### 4.5. Design and Architecture

Figure 4-2 shows the architecture of In-Hop Encryption solution in OpenWrt router.

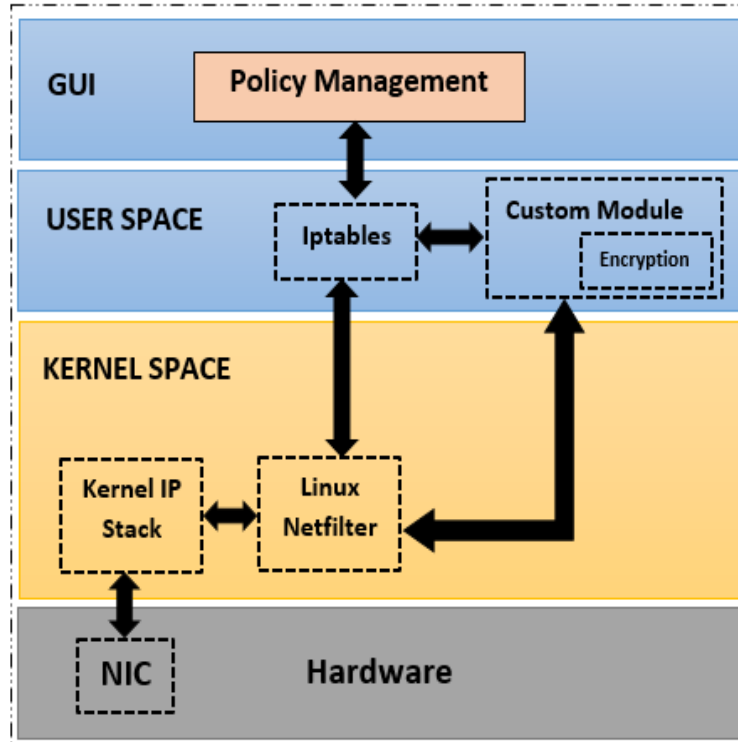


Figure 4-2 In-Hop Encryption Architecture

Each component is discussed below.

#### 4.5.1. Policy Management

As mentioned earlier in Section 2.5, OpenWrt can be accessed and managed through command-line interface and web browsers with the management IP address. Command-line interface provides facility to perform development related tasks. OpenWrt custom installation provides web user interface (LuCI WebUI) for the administrative purposes and it uses uHTTPd web server by default. uHTTPd web server nicely integrates with the OpenWrt's configuration framework [40].

User level policies of In-Hop encryption solution is managed through default web interface of the router. For this purpose, we have developed management application in existing web framework of OpenWrt's router. User can perform following tasks;

- User can view status of the encryption/decryption module.
- User can configure, view and delete traffic policies to encrypt IoT networks.
- User can view live logs of traffic matching rules.

Figure 4-3 shows web interface application for the policy management of In-Hop encryption module.

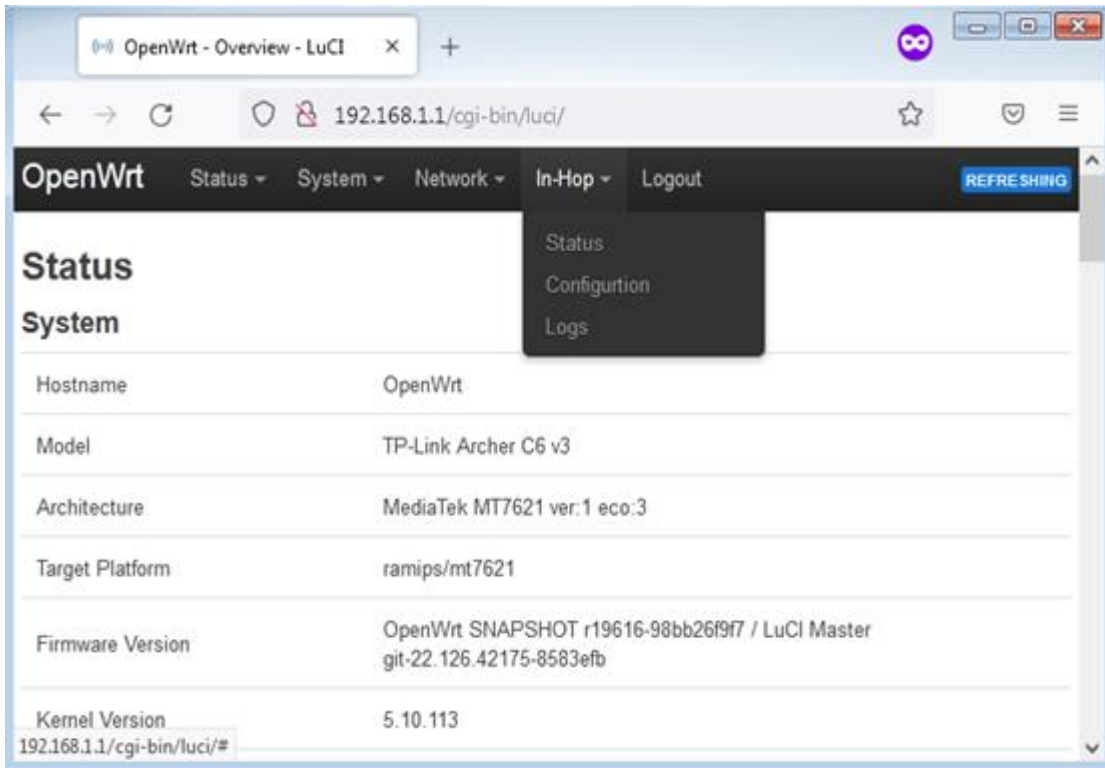


Figure 4-3 In-Hop Web Management

#### 4.5.2. User Space

User space is a logical memory separation in Linux based and today's modern systems (OpenWrt) that is dedicated for running user applications [41]. Most of the programs, tools, utilities and programming languages come pre-packaged with the operating system under the user space and can be utilized for developing and running custom programs. As mentioned earlier (Chapter 2, Section 2.6), Iptables is a user space program to handle network packets in Linux based systems. This is an administrative tool that provides firewalling capability to filter network traffic. Defined rules match the concerned traffic to filter and policies perform the required action on that packets. The action on the packets in Iptables is called target and actions can be called directly (ACCEPT, DROP etc.) or action can be delegated to a user space program. NFQUEUE is one of the actions (target) of Iptables that provides the capability of manipulating or



taking some decision on the network packets through user space program. The figure 4-4 shows below is an Iptables rule that queues the routed HTTP traffic towards user space program.

```
iptables -I FORWARD 1 -p tcp -dport 80 -j NFQUEUE --queue-num 1
```

Figure 4-4 Iptables Rule

Queue is a collection of packets indexed by an integer (packet id) and linked together in a linear sequence. Once packets are enqueued to the user program for some decision, the verdict on each packet by its index number get them released from the queue. The delayed decision on the packets can cause the queue full and packets can be dropped [42].

### 4.5.3. Custom Module

Custom module is a user space program written in ‘C’ language for the implementation of In-Hop encryption. Our custom module performs two tasks that are handling of packets (receiving and returning back) and applying encryption/decryption function to the application data.

Packet handling involves receiving queued packets and issuing verdict on each packet. Library named “libnetfilter\_queue” is used in the module that acts as a programming interface for the queued packets [43]. To handle the packets pushed to the queue, connection handler is opened (just like creating a socket). This queue connection handle is bound to the AF\_INET to process only IPv4 addresses. Figure 4-5 shows below the library functions used in C program for creating a new connection handler to process IP packets.

```

/* Pointer to Structure */
struct nfq_handle *nfqHandler;

/* Obtain Netfilter Queue Connection Handler */
nfqHandler = nfq_open ();

/* Bind Handler to Process IPv4 */
nfq_bind_pf (nfqHandler, AF_INET);

```

Figure 4-5 Obtaining Netfilter Queue Connection Handler

To bind the program with the queue, new queue handle is created with the parameters of corresponding queue number (mentioned in the Iptables rules), connection handler (opened earlier), the function to be called for each packet (CbEncrypt) and any data to be passed into the function at packet interception. Size of the packet data to get into the callback function is also specified. Figure 4-6 shows the functions used to bind the queue with our program.

```

/* Create New Queue with Parameters */
nfQueue = nfq_create_queue (nfqHandler, 1, &cbEncrypt, NULL);

/* Copy Full Packet in the function */
nfq_set_mode (nfQueue, NFQNL_COPY_PACKET, 0xffff);

```

Figure 4-6 Binding Newly Created Queue

After opening handler and creating a queue handle, next step is to retrieve the network packet and trigger the call back (cbEncrypt) function for the received packet. Figure 4-7 shows that this is done by the handle function. From the file descriptor of opened handler data is read continuously to the buffer.

```

/* Get the File Descriptor */
int fd = nfq_fd(nfqHandler);
/* Read the Incoming data to the buffer */
while ((int len = recv (fd, buff, sizeof(buff), 0)) && res >= 0)
{
/* Retrieve the network packet */
nfq_handle_packet (nfqHandler, buff, len);
}

```

Figure 4-7 Handling of Incoming Data

The encryption mechanism is called inside the callback function which triggers on receiving every packet. First of all, payload of packet data with its length is obtained from the queue and then a data structure containing all information regarding packet is created. The raw data obtained is then copied to that data structure. The reason for copying data into the buffer is to keep the original data untouched. Figure 4-8 shows the code of retrieving payload data length and copying raw queued data.

```

/* Retrieving payload and length */
int PayloadLen = nfq_get_payload (pkt, &PayloadData);
/* Pointer to the packet buffer structure */
struct pkt_buff *PktBuff;
/* Allocating queued packet data of IPv4 family */
PktBuff = pktdb_alloc (AF_INET, PayloadData, PayloadLen, 4096);

```

Figure 4-8 Allocating Raw Data to Buffer

After retrieving and copying packet data, payload data is extracted from the complete IP packet. Figure 4-9 is showing the payload extraction for the TCP data packets.

```
/* Getting TCP payload and its length */
if (ip -> protocol == IPPROTO_TCP)
{
    tcp = nfq_tcp_get_hdr (pkBuff);
    payload = nfq_tcp_get_payload (tcp, pkBuff);
    payloadLen = nfq_tcp_get_payload_len (tcp, pkBuff);
}
```

Figure 4-9 Payload Data extraction

As shown in the figure 4-9 the received IP packet at the network layer is encapsulated with the transport and network layer headers. Each of the IP and TCP headers are of 20 bytes in size (in case of UDP traffic, the header size is 8 byte) and payload actual data's maximum segment size (MSS) is 1460 bytes maximum [44].



Figure 4-10 Packet Data

The payload data is the actual application data generated by the corresponding application and only that private data is presented to the encryption or decryption function (according to the traffic policy rule).

```

/* Initializing the OpenSSL Library */
ERR_load_crypto_strings ();
OpenSSL_add_all_algorithms ();
OPENSSL_config (NULL);
/* Applying encryption to the payload data */
ciphertext_len = encrypt (payload, payloadLen, key, iv, ciphertext);

```

Figure 4-11 Packet Data

Encryption and decryption functions utilizes the “libcrypto” API within the OpenSSL library that provides symmetric (Chapter 2, Section 2.4) encryption and decryption operations using different algorithms and modes [45]. And we have utilized the “EVP” interface of OpenSSL to call the desired encryption algorithms. The length of keys and IVs (initialization vector) are converted according to the encryption algorithm being selected by the user.

```

/* Initializing OpenSSL library */
ERR_load_crypto_strings ();
OpenSSL_add_all_algorithms ();
OPENSSL_config (NULL);
/* Applying encryption to the payload */
ciphertext_len = encrypt (payload, payloadLen, key, iv, ciphertext);

```

Figure 4-12 Encrypt Payload

#### 4.5.4. Encryption Algorithms

The security of encryption solution relies on the algorithm being chosen. As encryption function is applied to each packet individually in our solution, stream ciphers would be fit in here encrypting one byte at a time. There are many stream ciphers available and among them the most common and popular is RC4 which we have implemented in our solution. RC4 is known for its simplicity and fast processing that’s why remained

industry standard for so many years. Wireless routers also use RC4 in Wired Equivalent Privacy (WEP) and Wi-Fi Protected Access (WPA) security protocols [46].

We have implemented another common encryption algorithm Advanced Encryption Standard (AES) in our solution. It is a block cipher but it can be operated as a stream cipher by using recommended modes of operation [47].

```
EVP_CIPHER_CTX *ctx;
ctx = EVP_CIPHER_CTX_new();
EVP_EncryptInit_ex (ctx, EVP_aes_256_ctr (), NULL, key, iv);
EVP_EncryptUpdate (ctx, ciphertext, &len, plaintext, plaintext_len);
EVP_EncryptFinal_ex (ctx, ciphertext + len, &len);
ciphertext_len += len;
EVP_CIPHER_CTX_set_padding (ctx,0);
EVP_CIPHER_CTX_free (ctx);
```

Figure 4-13 AES CTR Encryption Function

Packet handling also involves in returning back each packet after applying the encryption function.

#### 4.5.5. Kernel Space

Kernel is the core part of the operating system and it links the user space applications to the hardware of the system [48]. Applications make I/O requests via system calls and kernel facilitates them in most appropriate manner. Kernel is also responsible for memory management and resource utilization among multiple applications. Like user space, kernel space is also the logical separation of memory that is restricted for kernel related tasks only.

IP stack is also implemented in the kernel space that receives the packets from the network interface card, checks packets for any error, removes the IP header and perform defragmentation of fragmented packets. Another packet filtering framework “Netfilter”

[49] is also the part of kernel that interacts with IP stack by defining five different hooks (well defined points) in the packet traversal path.

1. NF\_IP\_PRE\_ROUTING
2. NF\_IP\_LOCAL\_IN
3. NF\_IP\_FORWARD
4. NF\_IP\_LOCAL\_OUT
5. NF\_IP\_POST\_ROUTING

These hooks are registered with the functions through kernel modules that are called when packets passing through the stack triggers the corresponding hook. These callback functions (hook functions) are basically the decision on the packets to either accept, drop or do something else with the packets. One of the hook functions provided by the framework is ‘NF\_QUEUE’ that insert the packets in the queue from where our user space custom module access those packets and apply encryption function on them and later on release back to the kernel. iptables (Chapter 2, Section 2.6 & Chapter 4, Section 4.7) is a user space administration tool of underlying Netfilter framework. Netfilter have many kernel modules to register with the hooks placed in the IP protocol stack.

#### **4.5.6. Interaction Between Kernel and User Space**

There is a segregation between both spaces so that important programs remain protected from user space applications and operating system necessary applications are always available. But for the communication between them different IPC (inter process communication) methods exist and Netfilter uses “netlink” sockets (nfnetlink) for this purpose [49]. Netfilter sockets provides standard protocol for the user and kernel spaces modules to talk to each other. It is a both way communication channel that transmit and receive “nfnetlink” formatted data packets. Figure 4-10 shows the interaction between user and kernel modules through “netfilter netlink” sockets. At user space corresponding library receive the queued packets pushed from the kernel.

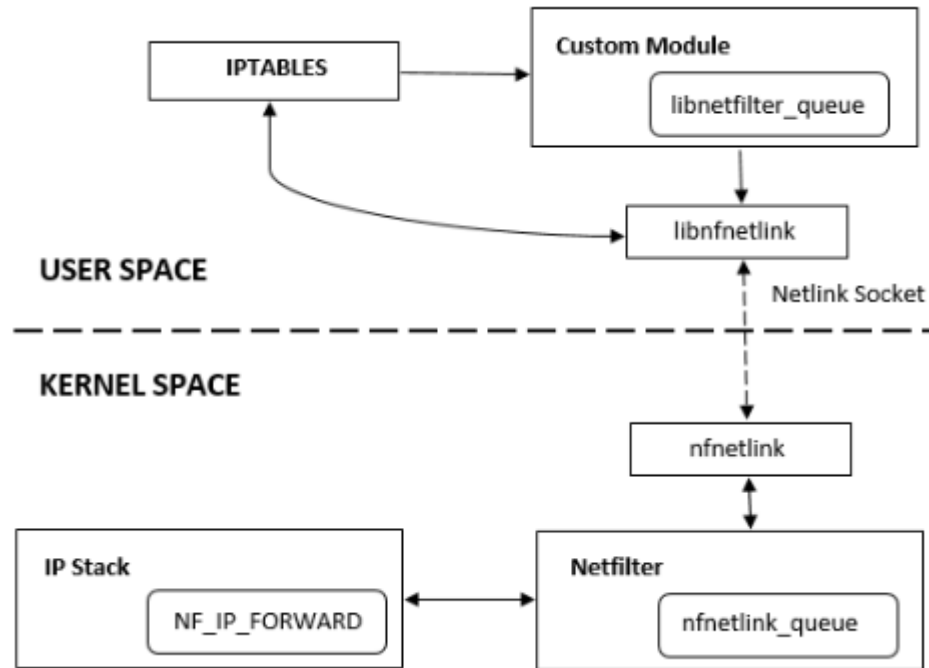


Figure 4-14 Interaction between Kernel and User Space

#### 4.6. Packet Flow

Points described below explains the packet flow path in detail.

- When packet enters the IP layer, it is passed on to the routing function.
- Routing function decides to send the packet towards forwarding function on the basis of destination IP address.
- Forwarding function delegates, the decision to the user space custom module (defined action in iptables rule).
- Connection handler for receiving IPV4 packet is opened at the user space program.
- New queue handler is created with indexed number and program is bind to that queue handler.
- Packet handler function receive the packet and send it to the callback function.
- IP and TCP headers are retrieved in the callback function.
- Encryption function is now applied to the payload data packet.



- TCP checksum is recalculated for the packet.
- Packet ID from the Netlink message header is retrieved and verdict is set to pass the encrypted packet back to kernel.
- Netfilter framework receives the packets from user space and forward them to the post routing module.
- Encrypted packet is passed on to the outer interface.

Figure 4-11 below depicts the complete packet flow path in proposed design within Netfilter framework.

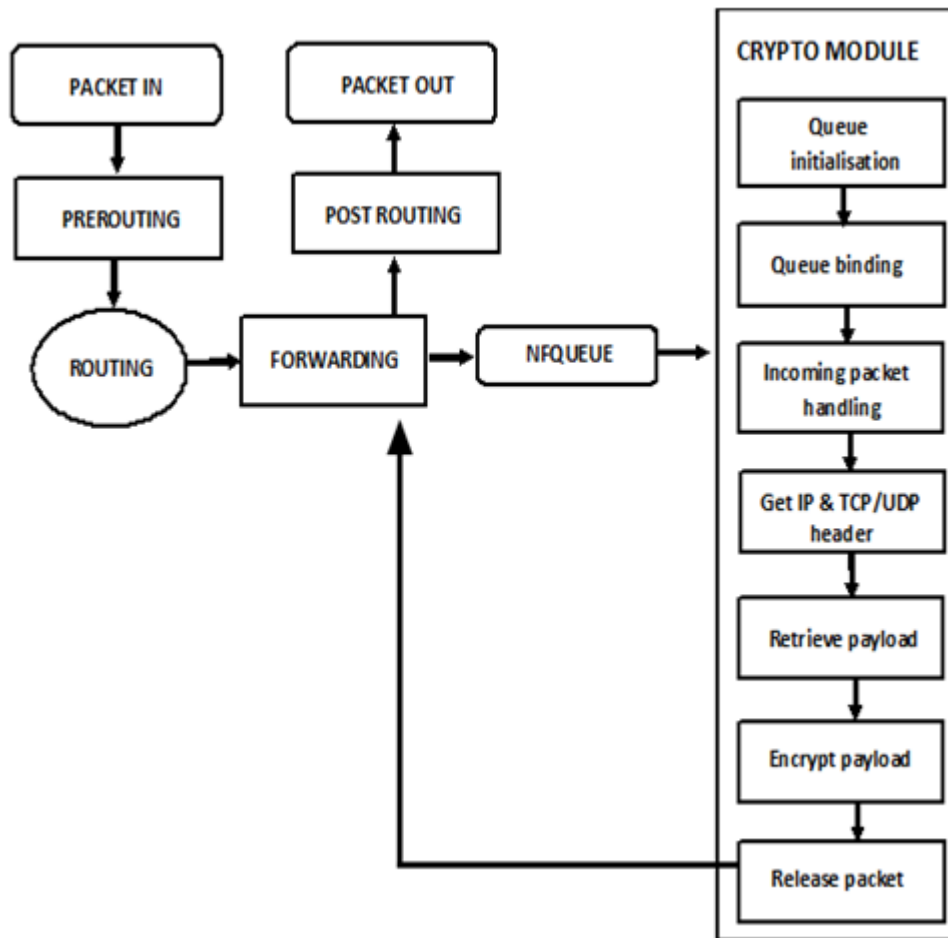


Figure 4-15 Packet Flow

# Chapter 5

## 5 Development and Implementation

In this chapter, we have discussed the development and implementation of our proposed solution along with the traffic policies to encrypt and decrypt the data of applications being used over the network.

### 5.1. Development

TP-LINK's router 'TL-WR1043ND' and 'Archer C6 | AC 1200' are chosen to work as gateway routers for development and deployment purpose in our work. The base firmware of these routers is OpenWrt (Chapter 2, section 2.4) and later on customized by router vendors. We flashed routers with OpenWrt operating system downloaded from the official website [50]. OpenWrt version of TP-LINK router has been shown in the figure 5-1.

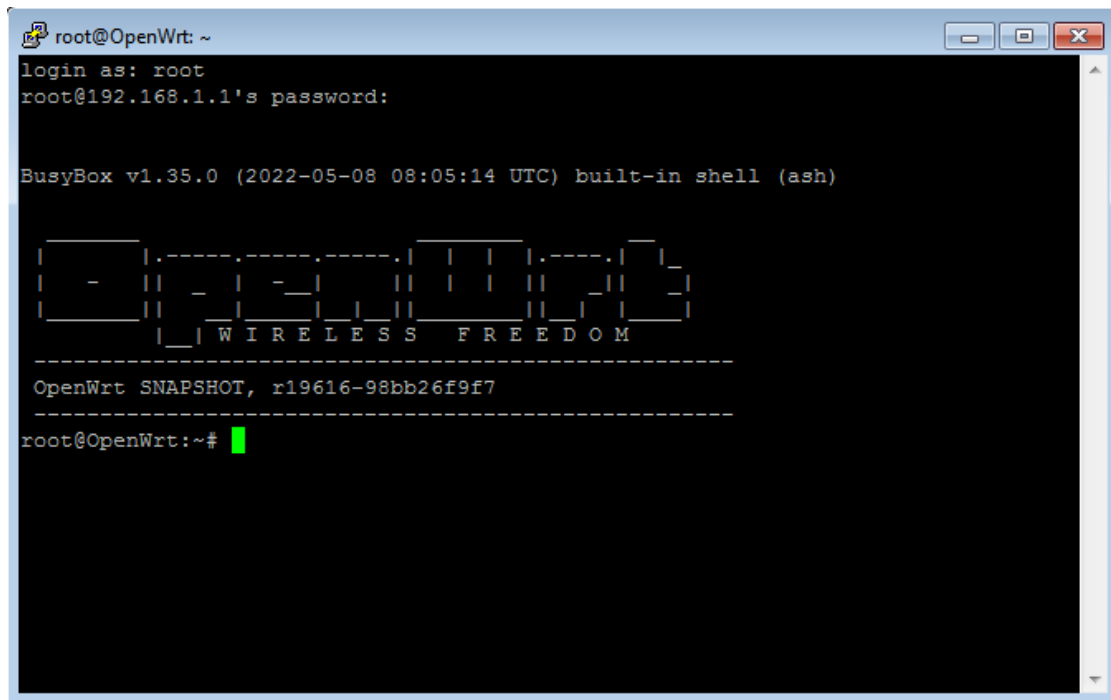


Figure 5-1 OpenWrt version

#### 5.1.1. Packages

Packages required for In-Hop encryption modules are installed after flashing routers with OpenWrt firmware. “iptables-mod-nfqueue” as discussed in (Chapter 4, Section 5-2) and (Chapter 2, Section 6) is installed for the queuing functionality of network traffic. Library “libnetfilter\_queue” (Chapter 4, Section 5-2-1) is installed as an API for handling queued packets. For encryption operations and functions in custom module library named “libopenssl” is installed.

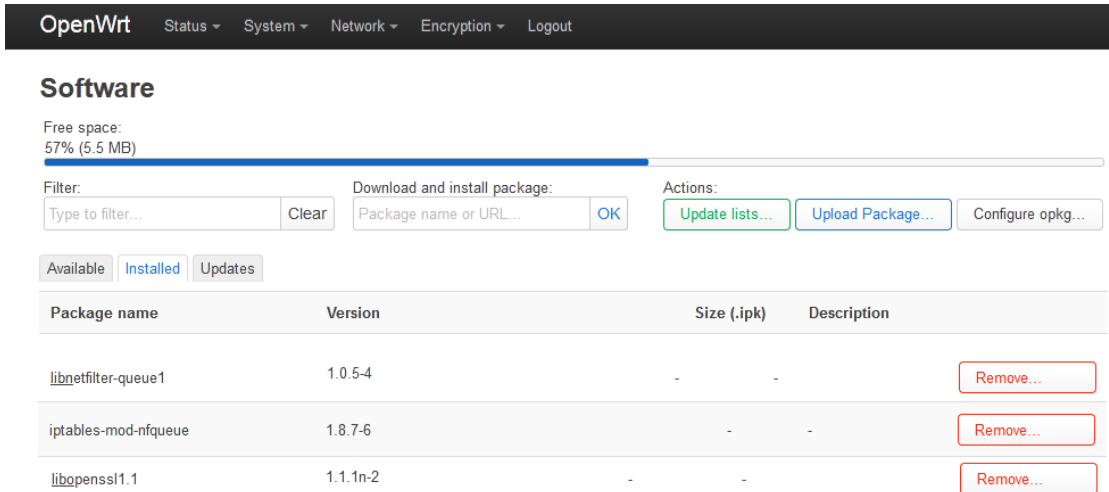


Figure 5-2 Packages

### 5.1.2. Cross Compiling Custom Module for OpenWrt

Home wireless routers have limited resources i.e. ram, rom, processing power etc. so it is not possible to install a compiler and compile the programs on it. However, there are many ways to develop custom modules or packages for OpenWrt routers [51].

For developing and compiling our custom In-Hop encryption module we build an OpenWrt image for routers (Chapter 5, Section 5-1) on the UBUNTU (Linux) PC.

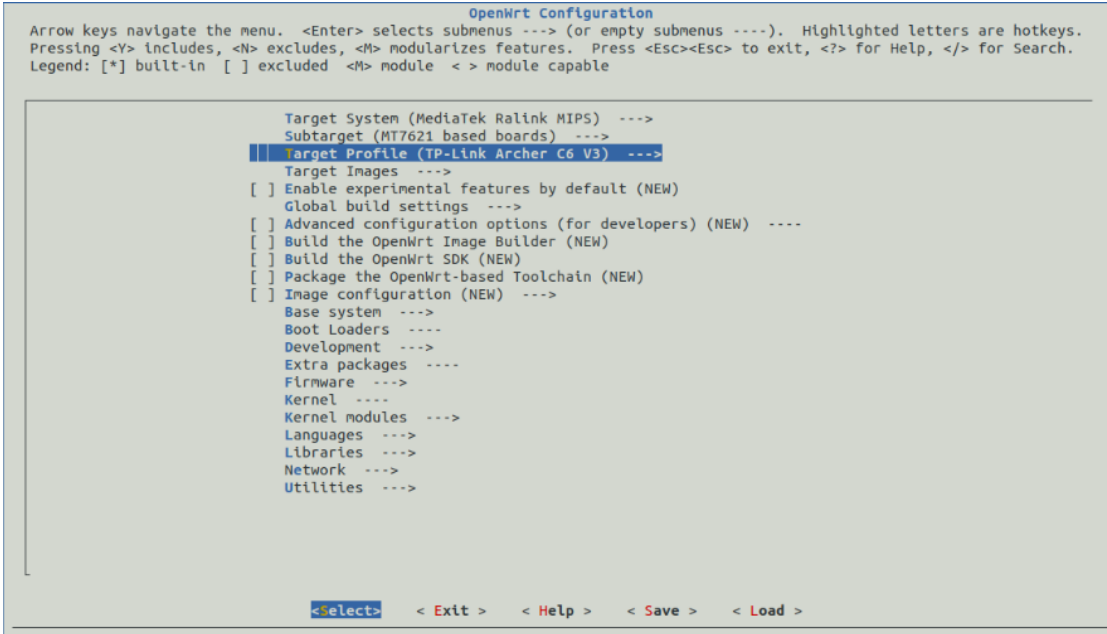


Figure 5-3 Build System Configuration

After building an image for target router we set the path environment variables of Linux PC according to the toolchain binary directories of OpenWrt build image. The same packages (Chapter 5, Section 1-1) installed in the router are compiled from sources under the toolchain [52] directory of OpenWrt build system in Linux PC. Toolchain is a tool that makes possible to execute a program on one platform (router) that has been compiled on another (Linux PC). Finally, program is cross compiled including all the required header files and shared objects. The compiled program is then copied to the router and executed there.

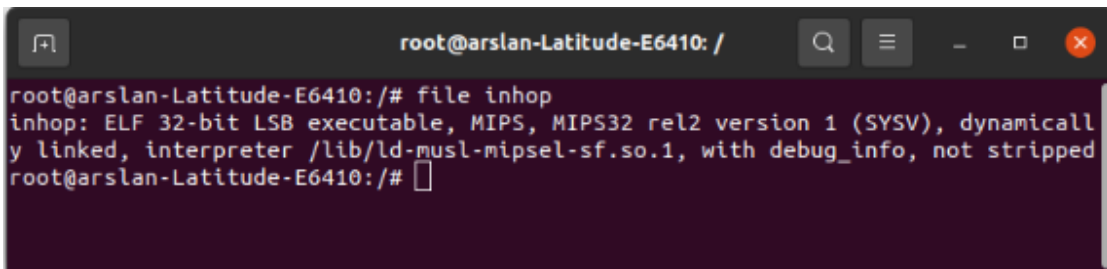


Figure 5-4 Cross Compiled Program

### 5.1.3. Web Development Interface for Policy Management

As discussed earlier (Chapter 4, Section 1.6), OpenWrt uses LuCI framework for its web user interface. We developed In-Hop encryption web management interface (luci-app-inhop) using the same framework. LuCI is basically the combination of Lu (lua programming language) and Ci (Unified Configuration Interface) that divides the interface into logical parts like views, models and also uses templating and object-oriented libraries. Figure 5-5 shown below is the status page of the application. It gives control to start, stop, restart, enable and disable the module.

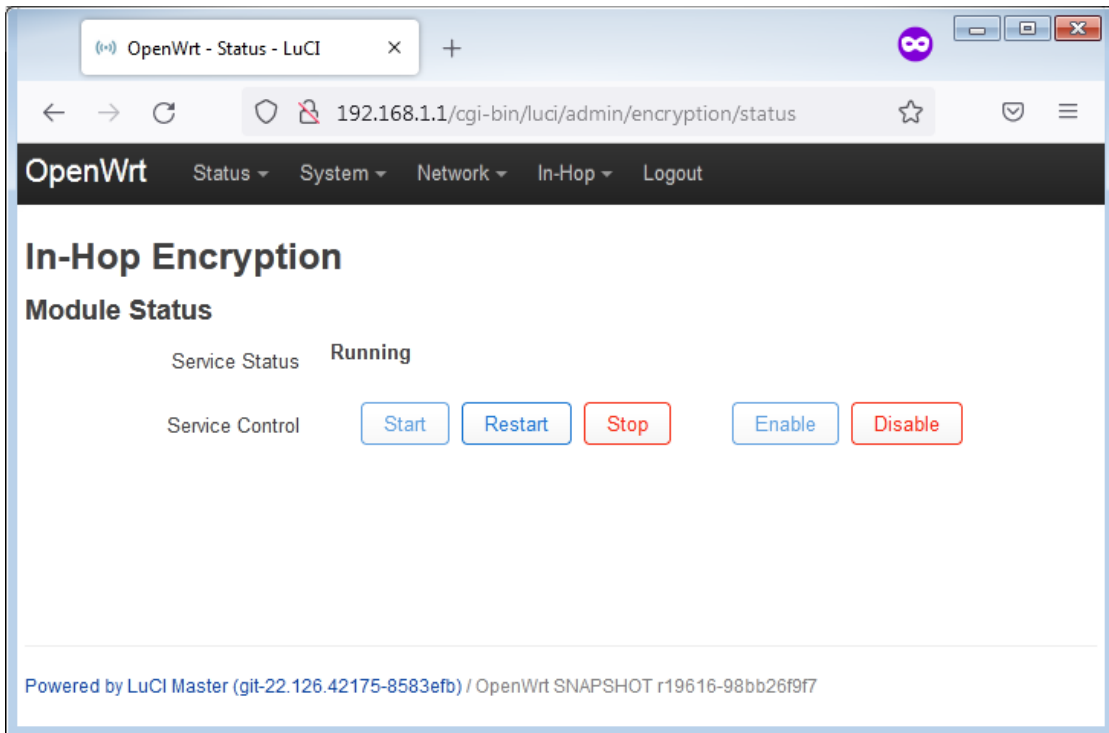


Figure 5-5 Status Web Page

In-Hop web management application is integrated with the backend In-Hop encryption module developed and cross-compiled for the router. Service control buttons starts or stops the executable binary program and gives the results. Traffic policies web page under the In-Hop insert or delete the iptables rules at the backend. Live logs can also be viewed for matching traffic policies.

## 5.2. Implementation for Online Chat Application

NC (netcat) is an multipurpose network utility used for reading and writing data to the TCP or UDP connections [53]. We have utilized this tool for creating an online chat application. It is a client and server model where server listens at specific port and clients connects to that port. Figure 5-6 is a network representation of NC client server chat application. Two OpenWrt flashed TP-LINK routers with our In-Hop encryption solution are connected point-to-point.

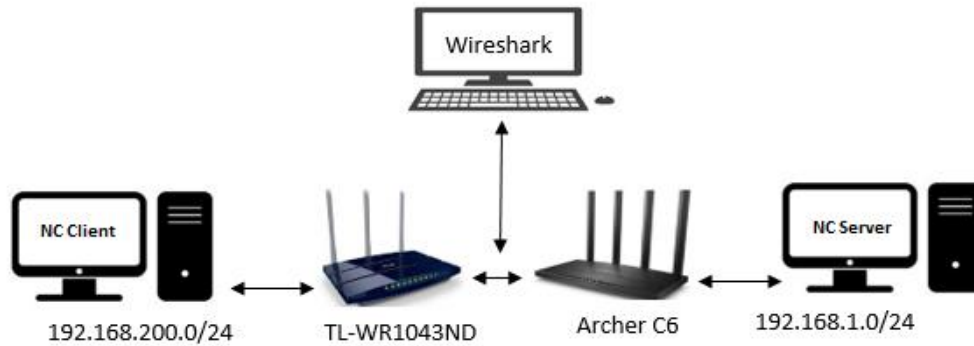


Figure 5-6 Chat Application Scenario

Network is configured at both routers for end-to-end connectivity as shown in the table 5-1.

Table 5-1 Network Configuration

ROUTER	LAN IP Address	WAN IP Address	ROUTESs
Archer C6	192.168.1.1	10.1.1.1	192.168.200.0
WR1043	192.168.200.1	10.1.1.2	192.168.1.0

At LAN (local area network) side of both routers' PCs are installed that are running NC client and server applications. Server-side PC is listening at TCP port '5001' and client PC connects to that port. After connection is established between both PCs, they can send chat messages to each other. Figure 5-7 is showing the server-side PC listening at port '5001' and receiving chat messages from the client PC.

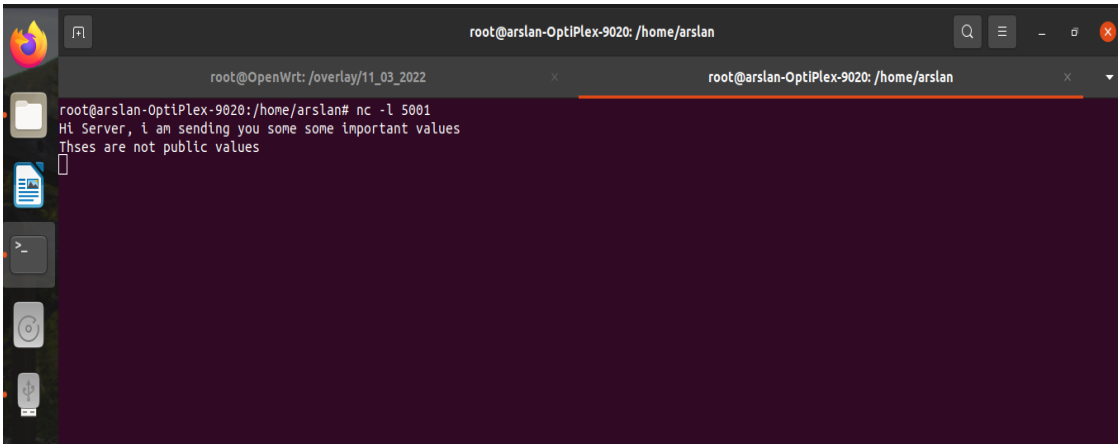


Figure 5-7 NC server

### 5.3. Implementation Verification

We have installed Wireshark [54] an open source packet analyzer in between both routers to capture the live traffic passing at the network. Traffic is captured before the encryption policy and also after the encryption policy being configured. This depicts the man-in-the-middle attack scenario where an intruder can somehow get access of the network and intercept the communication between two parties. Attacker can also intercept sensitive or private attributes like passwords. This setup is also an implementation verification of our In-Hop encryption solution where a third-party tool is being used to analyze the network traffic. Fig 5-8 shows the captured traffic from the middle of the both routers in the plain mode without encryption being applied to the chat traffic. Chat messages can be clearly shown from the IP addresses of client and server PCs.

```

> Internet Protocol Version 4, Src: 192.168.200.141, Dst: 192.168.1.184
> Transmission Control Protocol, Src Port: 37788, Dst Port: 5001, Seq: 1,
Data (55 bytes)
  Data: 48 69 20 53 65 72 76 65 72 2c 20 69 20 61 6d 20 ...
  [Length: 55]
0000  60 a4 b7 56 fa f6 c0 4a 00 2d 39 a0 08 00 45 00  `..V...J .-9...E.
0010  00 6b d8 ef 40 00 3f 06 1f 07 c0 a8 c8 8d c0 a8  .k..@.?. .....
0020  01 b8 93 9c 13 89 04 63 f4 62 fa a7 16 38 80 18  .....c .b...8..
0030  00 a6 24 61 00 00 01 01 08 0a a4 24 64 01 e3 ef  ..$a.....$d...
0040  19 6b 48 69 20 53 65 72 76 65 72 2c 20 69 20 61  .kHi Ser ver, i a
0050  6d 20 73 65 6e 64 69 6e 67 20 79 6f 75 20 73 6f  m sendin g you so
0060  6d 65 20 73 6f 6d 65 20 69 6d 70 6f 72 74 61 6e  me some importan
0070  74 20 76 61 6c 75 65 73 0a                          t values .

```

Figure 5-8 Plain Mode

After that at client-side router encryption policy is added to route all TCP traffic of port '5001' towards encryption module. And at server-side decryption policy is added to pass that traffic towards the decryption module. Decryption policy at server-side router is shown in the figure 5-9.

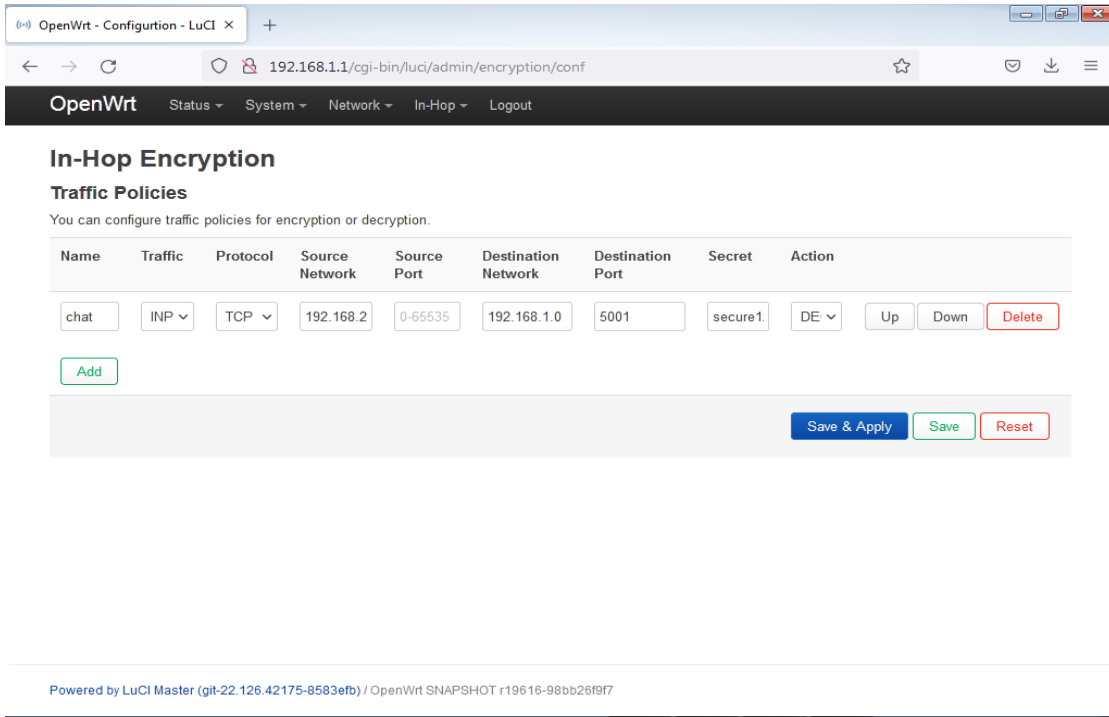


Figure 5-9 Encrypted Mode

By adding policies at both ends, no one can intercept plain chat messages from the network between two routers. All the traffic passing over the network is in encrypted form. Figure 5-10 is showing the captured traffic when encryption policy is being configured. It can be clearly shown that chat messages are not in plain text now.



```

> Internet Protocol Version 4, Src: 192.168.200.141, Dst: 192.168.1.184
> Transmission Control Protocol, Src Port: 37792, Dst Port: 5001, Seq: 1,
  Data (55 bytes)
    Data: 99 3c c0 e1 10 b6 ec 5a a7 28 6b 97 34 80 6c 8e ...
    [Length: 55]

```

---

0000	60 a4 b7 56 fa f6 c0 4a	00 2d 39 a0 08 00 45 00	`..V...J .-9...E.
0010	00 6b 6d 9b 40 00 3f 06	82 5b c0 a8 c8 8d c0 a8	.km.@.?. .[.....
0020	01 b8 93 a0 13 89 9f fa	5a 3e 40 01 b3 90 00 18	..... :>@.....
0030	00 a6 31 75 00 00 01 01	08 0a a4 2a 7e 3d e3 f5	..lu.... ..*cm...
0040	39 a7 99 3c c0 e1 10 b6	ec 5a a7 28 6b 97 34 80	9..<.... .Z.(k.4.
0050	6c 8e a1 f2 d3 bf 9f a1	a6 3b 15 3a 2e 70 dc 70	l..... ;:;X.V
0060	b3 08 9c 5a 5f 5b 9b 69	23 25 61 a6 00 45 b9 0a	...Z_[.i #%a...E..
0070	c0 16 02 da 01 20 4b be	bd	....(K. .

Figure 5-10 Encrypted Mode

# Chapter 6

## 6 Evaluation of Research Work

In this chapter, we have discussed the performance evaluation of implemented In-Hop encryption solution. Performance comparison is also done with some other solutions. In performance analysis we have measured performance of OpenWrt routers with and without encryption.

### 6.1. Performance Analysis

There are several metrics that measures performance of the router. We have performed some parameter tests to evaluate our purposed solution. Figure 6-1 shows the setup where two end nodes are configured to run the different tools and measure the performance of OpenWrt routers with and without encryption mechanism running on it. The specifications of the OpenWrt routers used to implement our solution are given in the table 6-1.

Table 6-1 Routers Specifications

BRAND	MODEL	CPU	FLASH	RAM	NICs
TP-Link	TL-WR1043ND, v2	Single Core, Qualcomm Atheros, QCA9558, 720 MHz	8 MB	64 MB	1 Gbit
TP-Link	Archer C6 AC1200, v3	Dual Core, MediaTek, MT7621DAT, 880 MHz	16 MB	128 MB	1 Gbit

#### 6.1.1. Throughput

Throughput [55] determines how much data can be transferred successfully at destination within a given time frame and indicates problems like packet loss, delay and jitter. Throughput varies with the specification of routers, network configurations (NAT, routing etc.) and encryption application along with the algorithm used.

We have used JPERF [56] to measure the throughput in the above given scenario at figure 5-6. JPERF is a graphical interface of IPERF that is an open source tool based on the client and server model. It performs different tests to measure the network performance over the network. We have installed the tool at both end PCs; one is configured as server and other as client. IPERF works by sending bytes of TCP or UDP data to the server node. As mentioned earlier throughput can be affected by many factors, as it also depends upon the client and server capabilities to generate, send and receive the traffic.

Throughput is measured in plain and encrypted modes with the following specifications of the PCs in table 6-2 below.

Table 6-2 PCs Specifications

PC	MODEL	CPU	RAM	NICs
Client	Dell Latitude E6410	Intel(R) Core (TM) i5-M560 @ 2.67 GHzx2	8 GB	1 Gbit
Server	Dell Latitude E6410	Intel Core i7-7500 CPU @ 2.70GHzx4	8 GB	1 Gbit

IPERF provides different parameters to set like payload size, time, protocols and flags. For our tests, we have chosen to send 500 Mbytes of TCP data with the default window size of 0.08 Mbytes and server is configured to listen at TCP port 5001.

#### 6.1.1.1. Plain Mode Throughput (without encryption)

Figure 6-1 is showing the results measured at the server side using the network setup in the figure 5-6. It shows that 500 Mbytes of data has been received at the server with the throughput of 277 Mbits/sec when no encryption is involved at the routers.

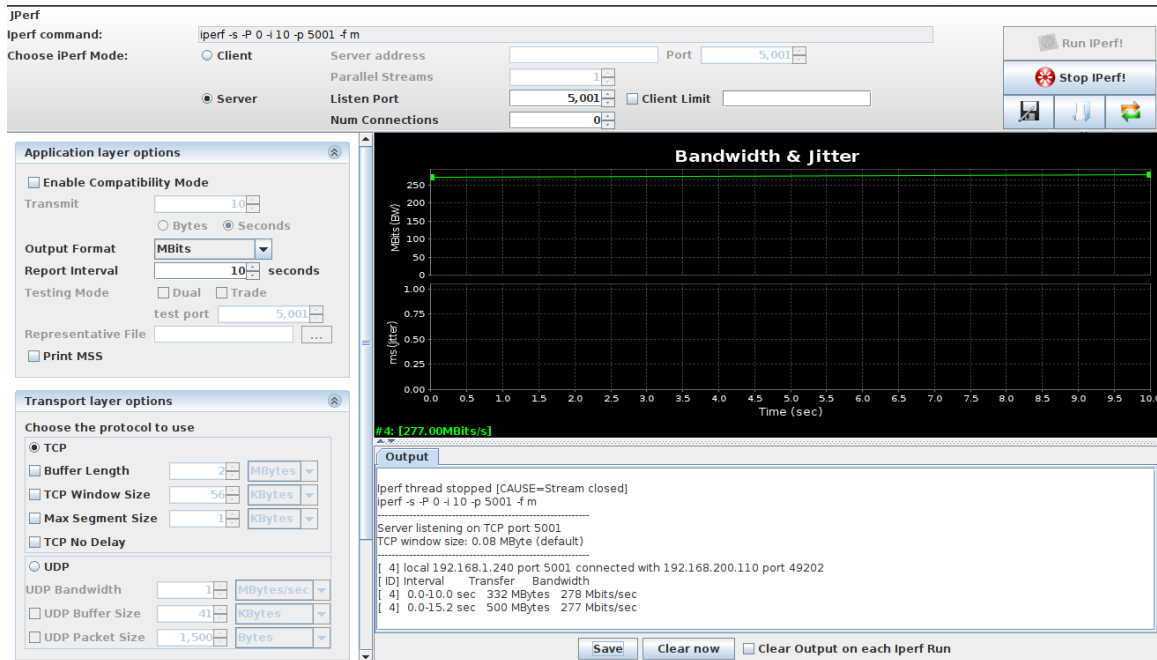


Figure 6-1 Throughput in Plain Mode

### 6.1.1.2. Encrypted Mode Throughput

As mentioned earlier (Chapter 4, Section 4-2) Two encryption algorithms AES and RC4 have been implemented in our In-Hop encryption solution. We have tested the performance analysis of our solution with both algorithms. The network scenario and configurations were same as in plain mode. Rules were added to pass TCP traffic of port 5001 through encryption module at client end router and decryption module at server end router. JPERF client sending all the traffic was encrypted by the OpenWrt router, passed to the other router, decrypted and then sent to the JPERF client.

In figure 6-2 it is shown that throughput of **39.4 Mbits/sec** has been achieved in the encrypted mode with the **RC4 algorithm**. Moreover all 500 Mbytes of data has been received with no loss.

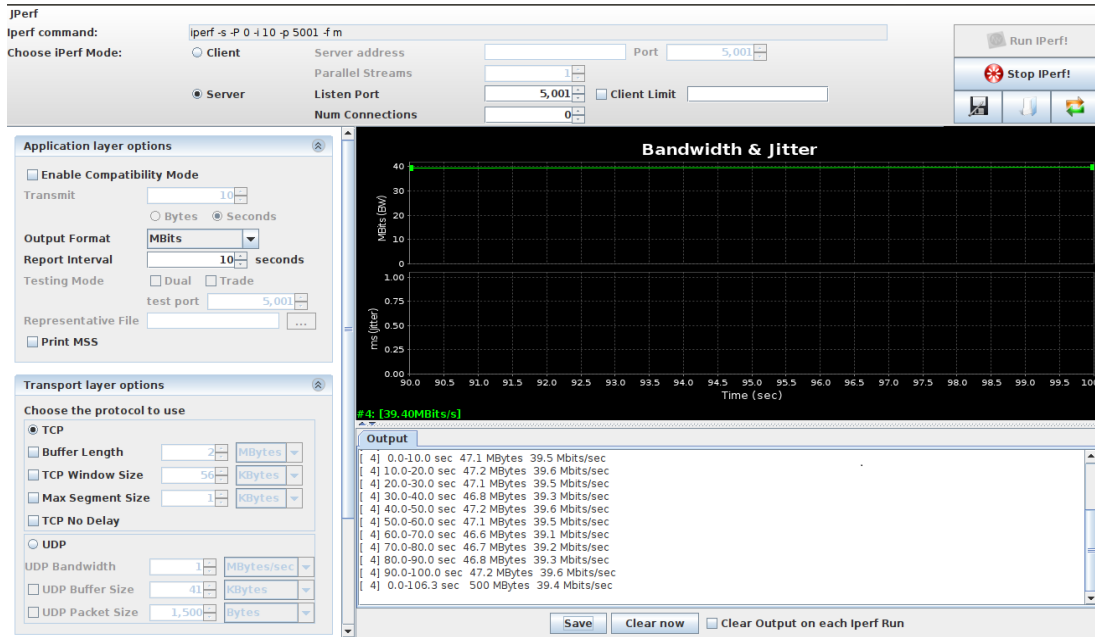


Figure 6-2 Throughput in Encrypted Mode (RC4)

Implementation of AES algorithm in our In-Hop encryption solution have produced throughput of **24.6 Mbits/sec** while receiving 500 Mbytes of TCP data as shown in the figure 6-3 below.

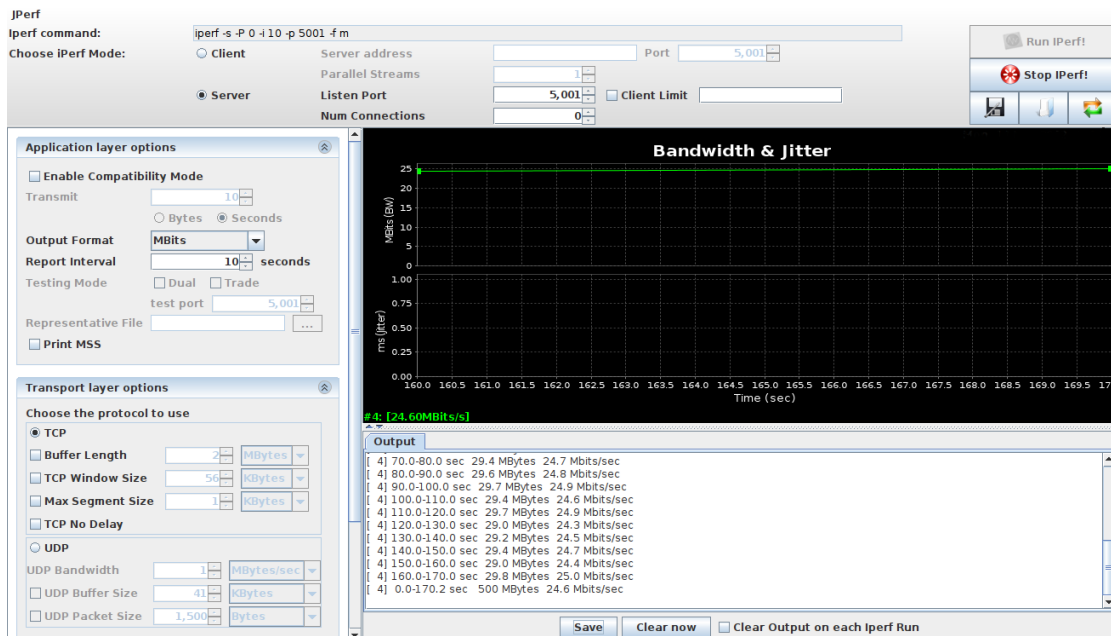
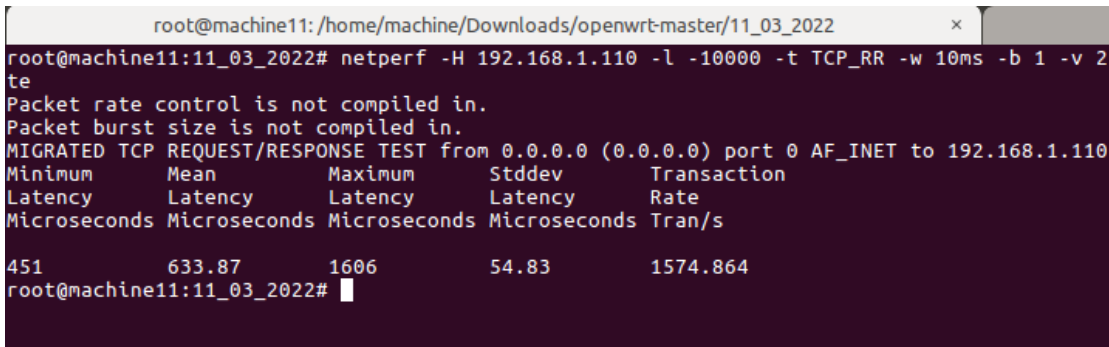


Figure 6-3 Throughput in Encrypted Mode (AES)

### 6.1.2. Latency

Latency [57] is another network performance metric that measures the time delay for packets to reach the server from client. In network terms it is called round trip time (RTT), that basically determines how fast the request and response can be exchanged between two nodes. There are lot of tools that can be used for this purpose like PING, IPERF and many others but according to Google and researchers at Southern Methodist University’s AT&T Center for Virtualization [58] “netperf” provides more reliable results. To measure the latency, we used “netperf” tool [59]. It basically measures the round-trip time (RTT) of transactions using TCP or UDP packets. Figure 6-4 shows the latency results without encryption using the setup shown in figure 5-6. The output results are showing minimum, mean and maximum latencies in micro seconds. Mean latency in case of plain mode where no encryption is applied is **0.63387 ms**.



```
root@machine11: /home/machine/Downloads/openwrt-master/11_03_2022
root@machine11:11_03_2022# netperf -H 192.168.1.110 -l -10000 -t TCP_RR -w 10ms -b 1 -v 2
te
Packet rate control is not compiled in.
Packet burst size is not compiled in.
MIGRATED TCP REQUEST/RESPONSE TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 192.168.1.110
Minimum      Mean          Maximum      Stddev      Transaction
Latency      Latency      Latency      Latency      Rate
Microseconds Microseconds Microseconds Microseconds Tran/s
451          633.87       1606         54.83       1574.864
root@machine11:11_03_2022#
```

Figure 6-4 Latency in plain Mode

Latency results are again produced in same network scenario with In-Hop encryption solution being applied. Output results where AES is used are shown in figure 6-5. The minimum latency in this case is 0.8 milli seconds, mean latency is **1.05 ms** and maximum latency is 3.3 milli seconds. Two other parameters standard deviation latency and transition rate are also shown in the output results.

```

root@machine11: /home/machine/Downloads/openwrt-master/11_03_2022
root@machine11:11_03_2022# netperf -H 192.168.1.110 -l -10000 -t TCP_RR -w 10ms -b 1 -v 2
te
Packet rate control is not compiled in.
Packet burst size is not compiled in.
MIGRATED TCP REQUEST/RESPONSE TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 192.168.1.110
Minimum      Mean          Maximum      Stddev      Transaction
Latency      Latency      Latency      Latency      Rate
Microseconds Microseconds Microseconds Microseconds Tran/s
801          1055.17      3348         102.20       946.610
root@machine11:11_03_2022# █

```

Figure 6-5 Latency in Encrypted Mode (AES)

Latency has also been observed in figure 6-6 where RC4 encryption is used. The mean latency is **0.931 ms** with RC4 encryption.

```

root@machine11: /home/machine/Downloads/openwrt-master/11_03_2022
root@machine11:11_03_2022# netperf -H 192.168.1.110 -l -10000 -t TCP_RR -w 10ms -b 1 -v 2
te
Packet rate control is not compiled in.
Packet burst size is not compiled in.
MIGRATED TCP REQUEST/RESPONSE TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 192.168.1.110
Minimum      Mean          Maximum      Stddev      Transaction
Latency      Latency      Latency      Latency      Rate
Microseconds Microseconds Microseconds Microseconds Tran/s
729          931.51       2538         60.44        1072.189
root@machine11:11_03_2022# █

```

Figure 6-6 Latency in Encrypted Mode (RC4)

## 6.2. Comparison with Existing Solution

IPsec [60] is a network layer security protocol that provides secure communication through encryption and authentication. Many commercial frameworks provide IPsec implementation for different platforms. We have installed strongSwan [61] on both routers and configured the network according to scenario (Figure 5-6). We established the IPsec VPN between two routers and compared the performance metrics. We used the same tools for measurement as used in In-Hop encryption solution.

Throughput is measured in tunnel mode with the encryption of AES 256-bit algorithm. Figure 6-7 shows the throughput of **14.2 Mbits/sec**

```

root@OpenWrt: ~ x root@C
root@machine11:/# iperf -c 192.168.1.110 -n 500M
-----
Client connecting to 192.168.1.110, TCP port 5001
TCP window size: 85.3 KByte (default)
-----
[ 3] local 192.168.200.149 port 37146 connected with 192.168.1.110 port 5001
[ ID] Interval      Transfer      Bandwidth
[ 3] 0.0-295.8 sec  500 MBytes   14.2 Mbits/sec
root@machine11:/# █

```

Figure 6-7 Throughput for IPsec VPN

Latency is also measured for IPsec VPN and figure 6-8 shows the minimum latency of **3.35 ms**.

Moreover, table 6-3 shows the comparison between In-Hop encryption and IPsec VPN in terms of throughput and latency with AES algorithm.

Table 6-3 PCs Performance Comparison Table

SOLUTION	THROUGHPUT (Mbits/s)	LATENCY (ms)
In-Hop Encryption	24.6	1.05
IPsec VPN	14.2	3.35

Comparison table clearly shows that our solution is more efficient in terms of throughput as well latency. As these two parameters are major network performance metrics.



# Chapter 7

## 7 Conclusion and Future Work

### 7.1. Conclusion

Network Security is very important in all those scenarios where attacker can gain access to the network and further there is no other layer security for private data. This is the most common case in internet of things (IoT) where each node is connected to network and exchanges information without any data payload protection. The main reason is the complexity and overhead of security solution implementations. The proposed solution provides very user-friendly interface in existing infrastructure to protect the data. Implementation and performance analysis show that it can be used in real time applications with minimum delay. Our solution can be implemented for applications running in existing networks

In-Hop encryption solution enables the end user to communicate and exchange personal or private information securely. Users can select and verify security policies easily. The encryption algorithms being implemented are standard and known to be most secure. In case of interception or gain of data over the network by some attacker, the encrypted data is of no use to someone.

### 7.2. Future Work

Our research in network security domain especially for IoTs, opens many directions towards latest technology research. Our research gives idea for the development of network security solutions for resource constrained devices. The security aspect of IoT is already neglected but security of data over the network should be the prime focus for researchers today. With the consideration of all the limitations of this technology we tried to develop a simple and efficient security solution that can be integrated seamlessly. This is a prototype implementation at lab level and tested at available devices (routers and PCs) with available network.

Network layer security demands more importance, there should be in-built security protocols for IoT networks and also suitable standard encryption algorithms should be developed. The performance can be increased by using lightweight encryption protocols. This research gives direction for open source community to contribute in designing more robust and secure frameworks. This will help the end users to use security as an integral part and would not make them choose performance over security.

The routers manufacturers should also look at integrating hardware security chips at these low-cost devices so that cryptographic operations would be fast. In term of performance, specialized hardware modules have better performance than software base modules. Vendors should provide network security solutions as a basic feature in their routers.

As our In-Hop encryption solution would also require keys management when implementing at industry level, this solution can be integrated with Amirhossein Safi [32]. It can also utilize the key generation and distribution idea of Rupesh Bhandari [33].

## 8 Bibliography

- [1] Zingbox Team, "IoT Threat Report" Unit 42, Palo Alto Networks, 2020. [online]. Available:- <https://iotbusinessnews.com/download/white-papers/UNIT42-IoT-Threat-Report.pdf>
- [2] Check Point, "Hackers Gained Access to 150,000 IP Cameras." [Online]. Available: <https://blog.checkpoint.com/2021/03/26/hackers-gained-access-to-150000-ip-cameras-inside-hospitals-police-departments-prisons-schools-and-companies-like-tesla-equinox/>
- [3] , Clarence Williams, "Hackers hit D.C. police closed-circuit camera network" [Online]. Available: [https://www.washingtonpost.com/local/public-safety/hackers-hit-dc-police-closed-circuit-camera-network-city-officials-disclose/2017/01/27/d285a4a4-e4f5-11e6-ba11-63c4b4fb5a63\\_story.html](https://www.washingtonpost.com/local/public-safety/hackers-hit-dc-police-closed-circuit-camera-network-city-officials-disclose/2017/01/27/d285a4a4-e4f5-11e6-ba11-63c4b4fb5a63_story.html)
- [4] P. C. van Oorschot and S. W. Smith, "The Internet of Things: Security Challenges," in IEEE Security & Privacy, vol. 17, no. 5, pp. 7-9, Sept.-Oct. 2019, doi: 10.1109/MSEC.2019.2925918.
- [5] I. Hussain, N. Pandey, A. V. Singh, M. C. Negi and A. Rana, "Presenting IoT Security based on Cryptographic Practices in Data Link Layer in Power Generation Sector," 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2020, pp. 1085-1088, doi: 10.1109/ICRITO48877.2020.9197905.
- [6] Zhang, C. & Green, R.C, "Communication security in internet of thing: Preventive measure and avoid DDoS attack over IoT network. Simulation Series" in Proceedings of the 18<sup>th</sup> Symposium on Communications & Networking, April 2015, pp 8-15.
- [7] H. R. Ghorbani and M. H. Ahmadzadegan, "Security challenges in internet of things: survey," 2017 IEEE Conference on Wireless Sensors (ICWiSe), 2017, pp. 1-6, doi: 10.1109/ICWISE.2017.8267153.

- [8] J. M. Carracedo et al., "Cryptography for Security in IoT," 2018 Fifth International Conference on Internet of Things: Systems, Management and Security, 2018, pp. 23-30, doi: 10.1109/IoTSMS.2018.8554634.
- [9] Pallavi Sethi and Smruti R. Sarangi, "Internet of Things: Architectures, Protocols, and Applications" in Journal of Electrical and Computer Engineering, Jan 2017, Article ID 9324035.
- [10] D. Navani, S. Jain and M. S. Nehra, "The Internet of Things (IoT): A Study of Architectural Elements," 2017 13th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), 2017, pp. 473-478, doi: 10.1109/SITIS.2017.83.
- [11] S. N. Swamy, D. Jadhav and N. Kulkarni, "Security threats in the application layer in IOT applications," 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2017, pp. 477-480, doi: 10.1109/I-SMAC.2017.8058395.
- [12] M. B. Yassein, M. Q. Shatnawi and D. Al-zoubi, "Application layer protocols for the Internet of Things: A survey," 2016 International Conference on Engineering & MIS (ICEMIS), 2016, pp. 1-4, doi: 10.1109/ICEMIS.2016.7745303.
- [13] Anna Gerber, Jim Romeo, "Connecting all the things in the Internet of Things", [Online]. Available:- <https://developer.ibm.com/articles/iot-lp101-connectivity-network-protocols>.
- [14] Rik Irons-Mclean, Anthony Sabella, Marcelo Yannuzzi, "IoT and Security Standards and Best Practices", [Online]. Available:  
<https://www.ciscopress.com/articles/article.asp?p=2923211&seqNum=6>
- [15] Panchiwala, Shivani; Shah, Manan, "A Comprehensive Study on Critical Security Issues and Challenges of the IoT World" in Journal of Data, Information and Management, 2020, doi:10.1007/s42488-020-00030-2
- [16] Dhuha Khalid Alferidah and NZ Jhanjhi, "A Review on Security and Privacy Issues and Challenges in Internet of Things", in IJCSNS International Journal of

Computer Science and Network Security, April 2020, VOL.20 No.4.

- [17] Ahmad, M., Younis, T., Habib, M.A., Ashraf, R., Ahmed, S.H. (2019). A Review of Current Security Issues in Internet of Things. In: Jan, M., Khan, F., Alam, M. (eds) Recent Trends and Advances in Wireless and IoT-enabled Networks. EAI/Springer Innovations in Communication and Computing. Springer, Cham. [https://doi.org/10.1007/978-3-319-99966-1\\_2](https://doi.org/10.1007/978-3-319-99966-1_2)
- [18] Kevin Curran, Niall Smyth and Bryan McGrory, "Cryptography", in Cyber Warfare and Cyber Terrorism. [Online]. Available:- <https://www.igi-global.com/chapter/cryptography/7440>
- [19] Charles Edge & Daniel O'Donnell , "Introduction to Cryptography", in Enterprise Mac Security Apress, Berkeley, CA. [Online]. Available:- [https://doi.org/10.1007/978-1-4842-1712-2\\_21](https://doi.org/10.1007/978-1-4842-1712-2_21).
- [20] Donald Davies, "A brief history of cryptography", in Information Security Technical Report, Volume 2, Issue 2, 1997, Pages 14-17
- [21] William August Kotas, "A Brief History of Cryptography", University of Tennessee - Knoxville.
- [22] Faiqa Maqsood, Muhammad Mumtaz Al, Muhammad Ahmed, Munam Ali Shah, "Cryptography: A Comparative Analysis for Modern Techniques", (*IJACSA*) *International Journal of Advanced Computer Sciences and Applications*, Vol.8, No.6, 2017.
- [23] Khurana, Mehak & Kumari, Meena, "Security Primitives: Block And Stream Ciphers", in JNU Conference, April 2015.
- [24] Andreas Klein, "Stream Ciphers", Springer London, [Online] Available <https://doi.org/10.1007/978-1-4471-5079-4>
- [25] Openwrt, "Openwrt Wireless Freedom", [Online] Available <https://openwrt.org>
- [26] Weiss, Aaron, "*The Open Source WRT54G Story*". Wi-Fi Planet. Retrieved July 5, 2018.

- [27] Andreasson, Oskar. "Iptables Tutorial 1.2. 2." *Copyright© 2001–2006 Oskar Andreasson, GNU Free Documentation License* (2001).
- [28] A. K. Simpson, F. Roesner and T. Kohno, "Securing vulnerable home IoT devices with an in-hub security manager," 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), 2017, pp. 551-556, doi: 10.1109/PERCOMW.2017.7917622.
- [29] Martin Serror, Martin Henze, Sacha Hack, Marko Schuba, Klaus Wehrle, "Towards In-Network Security for Smart Homes", in Proceedings of the 13th International Conference on Availability, Reliability and Security, August 2018, Article No.18, Pages 1–8
- [30] A. Dua, V. Tyagi, N. Patel , "IISR: A Secure Router for IoT Networks," 2019 4th International Conference on Information Systems and Computer Networks (ISCON), 2019, pp. 636-643, doi: 10.1109/ISCON47742.2019.9036313..
- [31] M. Nobakht, C. Russell, W. Hu and A. Seneviratne, "IoT-NetSec: Policy-Based IoT Network Security Using OpenFlow," 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), 2019, pp. 955-960, doi: 10.1109/PERCOMW.2019.8730724..
- [32] Safi, Amirhossein. "Improving the Security of Internet of Things Using Encryption Algorithms." *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering* 11 (2017): 558-561.
- [33] Rupesh Bhandari, Kirubanand V B, "Enhanced encryption technique for secure iot data transmission", in proceeding of International Journal of Electrical and Computer Engineering (IJECE), p-ISSN 2088-8708, e-ISSN 2722-2578.
- [34] I. Hussain, M. C. Negi and N. Pandey, "Proposing an Encryption/ Decryption Scheme for IoT Communications using Binary-bit Sequence and Multistage Encryption," 2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2018,

pp. 709-713, doi: 10.1109/ICRITO.2018.8748293.

- [35] Muhammad Usman, Irfan Ahmed, M. Imran Aslam, Shujaat Khan, Usman Ali Shah, "SIT: A Lightweight Encryption Algorithm for secure Internet of Things", in proceedings of (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 8, No. 1, 2017
- [36] E. R. Naru, H. Saini and M. Sharma, "A recent review on lightweight cryptography in IoT," 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2017, pp. 887-890, doi: 10.1109/I-SMAC.2017.8058307.
- [37] Mohammed Abbas Fadhil Al-Husainy, Bassam Al-Shargabi, "Secure and Lightweight Encryption Model for IoT Surveillance Camera", in proceedings of International Journal of Advanced Trends in Computer Science and Engineering, Volume 9, No 2, pp:1840-1847, 2020
- [38] E. R. Naru, H. Saini and M. Sharma, "A recent review on lightweight cryptography in IoT," 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2017, pp. 887-890, doi: 10.1109/I-SMAC.2017.8058307.
- [39] Bogdan Jeliskoski and Biljana Stojcevska, "Securing a home Network by Using Raspberry Pi as a VPN Gateway", in Conference 15<sup>th</sup> ciiT, 2018 at republic of North Macedonia, Mavrovo.
- [40] OpenWrt, "LuCI Web Interface OpenWrt Development", [Online] Available:- <https://openwrt.org/docs/guide-user/luci/start>
- [41] Scott McCarty, "Architecture Containers in Linux", [Online] Available: - <https://www.redhat.com/en/blog/architecting-containers-part-1-why-understanding-user-space-vs-kernel-space-matters>
- [42] Netfilter, "libnetfilter\_queue", [Online] Available:- <https://netfilter.org>
- [43] Netfilter Queue, "libnetfilter\_queue library for queuing", [Online] Available:- [https://netfilter.org/projects/libnetfilter\\_queue/doxygen/html/](https://netfilter.org/projects/libnetfilter_queue/doxygen/html/)

- [44] Protocol, "TCP/IP Protocol and Packet Structure", [Online] Available: - <https://cs.fit.edu/~mmahoney/cse4232/tcpip.html>
- [45] OpenSSL, "libcrypto API and libssl API of OpenSSL", [Online] Available: - [https://wiki.openssl.org/index.php/Libcrypto\\_API](https://wiki.openssl.org/index.php/Libcrypto_API)
- [46] Wikipedia, "Wi-Fi Protected Access and Wi-Fi Protected Access 2 ", [Online] Available: - [https://en.wikipedia.org/wiki/Wi-Fi\\_Protected\\_Access](https://en.wikipedia.org/wiki/Wi-Fi_Protected_Access)
- [47] Morris J. Dworkin, Elaine B. Barker, James R. Nechvatal, James Foti, Lawrence E. Bassham, E. Roback, James F. Dray Jr., "Advanced Encryption Standard (AES)", Federal Inf. Process. Stds. (NIST FIPS) - 197
- [48] Linux Info, "Kernel Space Definition and User Space in Linux", [Online] Available:- [http://www.linfo.org/kernel\\_space.html](http://www.linfo.org/kernel_space.html)
- [49] Linux Kernel, "Netfilter Framework for Linux Kernel", [Online] Available:- <https://programmer.group/netfilter-framework-of-linux-kernel.html>
- [50] OpenWrt, "OpenWrt Downloads, Firmware and Packages", [Online] Available:- <https://openwrt.org/downloads>
- [51] OpenWrt, "Creating Packages for OpenWrt Development", [Online] Available:- <https://openwrt.org/docs/guide-developer/packages>.
- [52] OpenWrt, "Toolchain", [Online] Available:- <https://openwrt.org/docs/guide-developer/toolchain/start>
- [53] Thomas Wilhelm, "Netcat Shell", in Professional Penetration Testing, 2010
- [54] Wireshark, "Wireshark", [Online] Available:- <https://www.wireshark.org/>
- [55] Wikipedia, "Network Throughput in Communication Channel, Minimum, Maximum and Average ", [https://en.wikipedia.org/wiki/Network\\_throughput](https://en.wikipedia.org/wiki/Network_throughput)
- [56] JPerf, "JPerf", [Online] Available:- <https://sourceforge.net/projects/jperf/>
- [57] Latency, "What is Network Latency, How to measure it", [Online] Available:- <https://www.a10networks.com/glossary/what-is-network-latency/>.



- [58] Derek Phanekham, Rick, "What a trip! Meseasuring Network Latency in cloud"  
[Online] Available:- <https://cloud.google.com/blog/products/networking/using-netperf-and-ping-to-measure-network-latency>.
- [59] Wikipedia, "NetPerf", [Online] Available:- <https://en.wikipedia.org/wiki/Netperf>
- [60] Wikipedia, "IPsec", [Online] Available:- <https://en.wikipedia.org/wiki/IPsec>.