TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Shoaib Kiyani

TorSNIP Hidden Service Proxy with End-to-End Security

Master of Science Thesis

Examiners: MSc. Joona Kannisto
Prof. Billy Brumley

Examiner and Topic approved in the Computing and Electrical Engineering council meeting on 16.05.2017

ABSTRACT

SHOAIB KIYANI: TorSNIP Hidden Service Proxy with End-to-End Security
Tampere University of Technology
Master of Science Thesis, 47 pages, 1 Appendix page
November 2017
Master's Degree Program in Information Technology
Major: Communication Systems and Networks
Examiner: MSc. Joona Kannisto & Prof. Billy Brumley

Keywords: Tor, Hidden Service, Tor2Web, LE, SNI

The onion router (Tor) is a software that provides an opportunity to access the blocked content over the Internet. It also provides anonymity to its users with the help of a protocol called Hidden Service (HS) protocol. It provides the ability to the users to conduct confidential communication without the possibility of getting trace back. It allows the operators to publish anonymous content without compromising their anonymity.

The '.onion' address can only be accessed using Tor browser. To access the HS with a regular Internet browser, for example, Google Chrome, Firefox etc., a service called Tor2web is used. It is a proxy server, which receives the user's request and forwards it to the targeted HS. The main issue identified in this service is that the service is not end-to-end secure and is prone to various attacks including content injection and content modification.

One of the possible solutions to this problem is to make an HTTPS connection directly to the onion site, rather than decrypting the packet at the intermediate node. This could make the communication secure from the user's browser until termination at the onion site i.e. makes it end-to-end secure. This is achievable with the deployment of TLS's Server Name Indication (SNI), which identifies the server name in the initial request.

The idea is to register a domain name and add an 'A' and 'AAAA' records and get a valid certificate from the certificate authority. Then create a hidden service and obtain its onion address. Map an onion:domain address and obtain a valid certificate for it. Modify the SNI script file according to the requirements and update the 'Table' field in the script. Finally, choose a virtual port and delegate the onion service name and all subsequent packets to the targeted hidden service.

PREFACE

I would like to express my deepest gratitude to my thesis supervisor and mentor Mr. Joona Kannisto, who provided me an opportunity to work with great people in a friendly environment. I would also like to thank him for his guidance, patience, and invaluable advice throughout the thesis work. I would like to thank my thesis examiner Prof. Billy Brumley for his kind support and advice before and during the evaluation of the thesis work.

I would like to thank my whole family for their love and support throughout my life. Last but not the least, I would like to thank my all friends, with whom I spent my valuable time and captured some of the precious moments of my life.

Tampere, 20.11.2017

Shoaib Kiyani

Table of Contents

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

Tor	The onion routing
SNI	Server Name Indication
TLS	Transport Layer Security
SSL	Secure Socket Layer
CA	Certificate Authority
M.I.T.M	Man-in-the-Middle
HS	Hidden Service
RP	Rendezvous Point
P2P	Point to Point
DNS	Domain Name System
DHCP	Dynamic Host Control Protocol
LE	Let’s Encrypt
HSTS	HTTP Strict Transport Security
HPKP	HTTP Public Key Pinning
EFF	Electronic Frontier Foundation
IP	Internet Protocol
HTTP	Hyper Text Transfer Protocol
TCP	Transmission Control Protocol
VPN	Virtual Private Network
VPS	Virtual Private Server
I2P	Invisible Internet Project
VoIP	Voice over Internet Protocol
Tor2Web	Tor to Web
TorSNIP	Tor Server Name Indication Proxy
OR	Onion Router
URL	Unified Resource Locator
ACME	Automatic Certificate Management Environment
AES	Advance Encryption Standard
SHA	Secure Hash Algorithm

1. INTRODUCTION

Today is the era of science and technology, and the enormous advancement in the field of information technology has led the Internet spread all around the world. The number of Internet users these days are more than 3 billion [1] and this figure is increasing day by day. Internet consists of an interconnected computer network connected using standardized communication protocols and thus forming a global network. World Wide Web invented in 1991 at CERN [2] consists of HTML and HTTP protocols used by the browsers to communicate with the servers. The Web is certainly a way of exchanging information over the Internet backbone. Interestingly, the Internet is divided into two parts, surface web and deep web internet [3].

The surface web also called visible web is the part of the Internet, commonly available to the public. The surface web is anything searchable using any search engine e.g. Google, Bing etc. These include the websites like Social websites, Blogs etc. or any such domains searchable by the search engines that end in '.com', '.org' or similar variations. Although many users use the surface web on the daily basis, it is interesting to know the fact that it only consumes 4% of the total web. Yes, it is true that only a small portion of the Internet web is available to the public, the rest is deep web. It is like an iceberg, which has small part above the surface of the water and generally has a larger percentage (around 91%) below the water [3].

Larger portion of the Internet is the deep web, which is not visible to the public. Search engines cannot index the deep web content. It includes the content that is generally important and private. For example, the content stored on the Google drive, bank account details, population index number and much more, unless it is shared on the surface web. Moreover, there are some secret contents kept hidden from the public, including criminal records stored in a database by police or security agencies like FBI, CIA. To access the content in the deep web, an authorized form with valid input values is required. For example, to access the bank account, the user has to authorize himself using provided authentication methods and upon successful verification, the user gets certain rights to access the account details.

There is another small portion of the deep web called dark web, which the search engines cannot index. This part of the deep web is not similar to the rest, although it is also indexable, it uses separate network within the Internet to reach the target website. It can be a small peer-to-peer network, or a large network like Tor, Freenet, and I2P, operated by individuals and public organizations. The users of darknet usually refer to a regular web as a 'Clearnet', because it is unencrypted in nature [4]. Unlike Clearnet, the Tor dark web has the top-level domain ends with '.onion' whereas I2P has '.i2p'.

1.1 Motivation

Anonymity means that the real person behind the message is unknown. A common variant of anonymity is pseudonym [5], which shows another name in place of a real name. Sometimes, the reason behind it is to show that the same person is behind different messages. Another variant is 'deception' in which the person behind, tries to impersonate different authority or expertise.

Back in history, America's farmers in 1787 published their arguments anonymously to argue for its constitution [6]. Similarly, many secret documents released, related to corruption and other mishaps during current and in previous years. The source is still unknown to these publications and investigation is still in process. Some of the biggest leaks in recent years [7] includes Panama papers, The Guantanamo Files, Cablegate and much more.

The dark web is only accessible using specific software e.g. Tor and I2P. Tor's Hidden Service also known as onion service is anonymously accessible only by using Tor browser bundle. Tor browser also provides access to the Clearnet content that is otherwise blocked or inaccessible using regular internet browser e.g. Chrome, Firefox etc. Essentially, Tor provides end-to-end anonymity in case of browsing HS using Tor browser, but to browse Clearnet domain, Tor does not guarantee end-to-end anonymity i.e. the HS operator remains anonymous but the client accessing the service is not anonymous. Although the Tor browser builds a circuit of encrypted connections through relays and extends one hop at a time, the exit node may reveal some information useful for the eavesdropper. The user (sender) may remain anonymous to the service [8]; however, the user's credentials may still be required to connect to the server.

The onion websites are also accessible without the use of Tor browser bundle. A proxy server called Tor2web handles the requests received from the regular Internet browser and forwards it to the requested HS. Tor2web proxy server has the Tor installed on it and the user does not require any installation on the system. This could be an advantage for the users, who do not want identification of any Tor related activity on their system. It is also useful because, the end users do not require any specific software to access the HS, rather a regular browser could retrieve the HS content. It helps anonymous publishing to reach a wider audience.

The use of Tor2web proxy for retrieving hidden content has a disadvantage, which could lead to several threats. The communication between the user agent (browser) and the Tor HS using Tor2web proxy lacks anonymity and security. The user agent encrypts the user's request and forwards it to the Tor2web proxy server. The Tor2web proxy server decrypts the received request and identifies the onion hostname. The proxy server then forms an onion circuit to the targeted HS and relays the user's request to it. Also, the data sent back

from the HS to the Tor2web proxy is already decrypted [8]. Thus, the Tor2web instance could possibly tamper the content transferred between the user agent and the onion site.

Based on the limitations associated with the Tor2web proxy server in forwarding the user's requests to the Tor HS, TLS's Server Name Indication (SNI) extension is, therefore, more suitable in relaying the user's requests securely to the HS. TLS's SNI extension indicates the name of the server (target HS in this case) at the beginning of the TLS handshake process, the user agent is trying to reach. Therefore, the user's request remains encrypted from the user agent until termination at the requested HS.

1.2 Problem Statement

The aim of this thesis work is to design and test a service that could provide access to the onion sites using regular Internet browser. There is already a proxy service Tor2web in use, which provides access to the onion sites on the surface web. The major drawback of this service is that it is not end-to-end secure and trustworthy. All the communication passing through is transparent to this service. Even if the browser encrypts the message earlier, the service decrypts the message and forwards it accordingly. The end users have to rely on this service and have to bear the risk of attacks such as message tampering and content injection.

This thesis work uses Tor network to build a service that provides end-to-end encryption to its users. The SNI proxy forwards the user's request directly to the target onion site without any alteration. This could make it secure and therefore trustworthy for the users, which increases its use by the public. To enhance the trust of the users, pinning techniques are useful and therefore can be used.

1.3 Structure of the work

The thesis starts with the review of Tor hidden service and related work. The literature review revealed more about the architecture of the Tor, its working methodology, and current deployment. Anonymity tools and services chapter discusses different available anonymity networks and their working. Furthermore, Chapter 3 discusses the Transport Layer Security protocol and its additive features including HTTP Strict Transport Security and HTTP Public Key Pinning. Chapter 4 is about the TLS certificate authorities and validation practices at different levels i.e. domain level, organization level and extended level. It also highlights chosen certificate authority. Chapter 5 is about the tools used and the implementation procedure as well as the results obtained with a brief description. Chapter 6 includes the analysis section and conclusion of the work.

2. Anonymity tools and services

This chapter describes the latest available anonymity tools and related services they provide. It also describes the relationship of these tools with the traditional Internet. Tor's Hidden Service and Tor2web proxy service are also discussed under this chapter.

2.1 The Onion Routing (Tor)

The onion routing abbreviated as Tor is a network of hidden nodes that act as a relay to protect the user's privacy and data over the Internet. It is named 'onion' because the messages are encapsulated in layers of encryption, which resembles layers of an onion, in the application layer of the communication protocol stack. The Tor network is a group of free volunteer routers called relays that are deployed more than seven thousand [9] in numbers, which are used for directing the Internet traffic and thus concealing the user's location and usage. Tor also provides the option to the sender and the receiver to keep their identity hidden from each other or the initiator disclose the identity or vice versa. In this situation, the sender and the receiver may know each other but are hidden from others on the network. In the Tor network, each chosen hop is encrypted until the exit node, thus making it difficult and challenging the attackers conducting the network surveillance, to break the network and steal or use the shared information. It makes it challenging to track back the user's Internet activity (website visited, instant messaging etc.) through its network [10]. Tor network also helps the users to access the blocked content, which is otherwise not accessible on the Internet. Besides keeping the user's privacy safe and secure, Tor also intends to keep the user's Internet activities safe from monitoring.

From individuals to the big organizations and Law enforcement agencies, Tor is increasingly becoming the need of everyone. Individuals use Tor to keep their information private from the websites they visit, the information they exchange with their family members or friends through instant messaging, also to visit the websites that are blocked by the local Internet providers. Tor also provides the users the ground for sharing socially sensitive communication [10]. The non-governmental organizations, whose workers are working in some foreign countries use Tor network to hide the actual location of the worker(s) and the organization. There are many other corporations and organizations like Electronic Frontier Foundation (EFF) and Indymedia, who recommend using Tor in order to safeguard their members' online privacy and security [11]. Some of the corporations recommend Tor over traditional VPN because the traditional VPN reveals the information about the time spent on the communication and the amount of communication that happened. Similarly, the law enforcement agencies use Tor for surfing the websites without leaving the government IP addresses in their weblogs [10].

Tor does not completely solve the issue of anonymity [11] and is not even designed to completely erase the fingerprints, but instead, reduces the risk of being traced back by focusing only on protecting the transfer of data. Many individuals and groups also use Tor to gain access to censored information, by keeping their identity anonymous, so that

they could carry out some political activities against the state etc. Tor has been named as the dark web in relation to Bitcoin and Silk Road [12]. It has been targeted by several security agencies and British National Crime agency remains more successful in gaining access to the Tor network showing that the anonymity of the network is not bulletproof, but of course, brings lots of hurdles and difficulties while breaking into the network [13]. Tor has gained popularity over the years and now millions of users prefer using Tor for their daily Internet surfing.

2.2 Traditional Internet and Tor

In traditional Internet communication, when two parties are communicating with each other, their connection is established directly [8] with each other through several nodes in the Internet cloud and the same channel is used for both sending and receiving of the packets unless some node disappears from the Internet cloud. If the established link is normal i.e. HTTP, then the sender and the receiver, both have the information about each other and even the eavesdropper who is observing the link may know that these two parties are communicating with each other and the information they are sharing with each other. However, if the communication link uses HTTPS, then the eavesdropper who is observing the link may know the two parties communicating with each other, but does not know the actual communication carried. This is because the actual payload sent from the sender to the receiver and vice versa is encrypted using encryption techniques (symmetric and asymmetric) and sent over the secure link. This way the communication remains safe, but the eavesdropper is still accessible to some information about the sender and/or the receiver i.e. the identity of the user, geographical location of the user, IP address and the sites visited by the user etc.

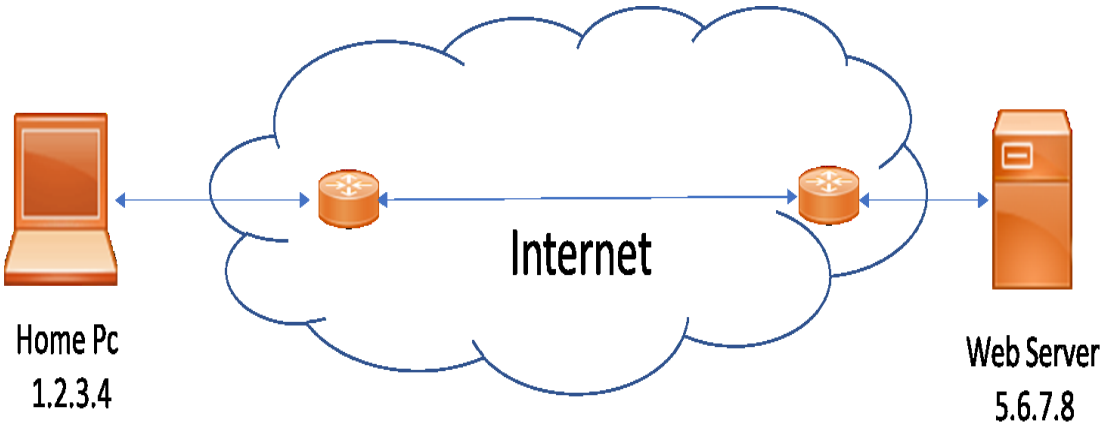

Figure 2.1: Traditional Internet communication

There are several technologies that may be helpful in dodging the eavesdropper by providing them false information, for example, use of Virtual Private Network (VPN) is

helpful in publishing different public IP address to the eavesdropper, which may show different geographical location of the user [14]. There are several benefits of VPN over other technologies including the use of encrypted tunnels for the data transfer and masking the user's real public IP with one that is different and foremost the connection is fast. However, there are still some public entities in the network e.g. VPN server, who may know the user's actual IP address and other information of the user. Since the VPN, providers may have the information about the user, so the security agencies may ask to share the information with them and help them in tracing back to the user in some situations. Therefore, the user is still identifiable and reachable.

The Tor network, on the other hand, is an independent network. It not only secures the communication that is being done between the sender and the receiver but also hides the user's details from others within the network [11]. Therefore, it is quite hard for the middleman to identify the information that is being shared and trace back the user's details [10]. This is the main reason for choosing Tor network as part of my Thesis work. As discussed earlier, the Tor network does not provide complete protection and not even other similar deep web networks, but secures most of the part and provides challenges to the middleman to get into the network [13].

The earlier paragraph briefly explained the basic analogy of the Tor network, which clarifies the basic concept of the Tor network. Getting deep into the Tor network, the mathematical analogy of the Tor network will be observed, which will explain the practical working of the network and will further discuss the possibilities of keeping the user's information secret.

The Tor network consists of nodes also called relay nodes, expanded throughout the world, used further for transmission of packets from source to destination and vice versa.

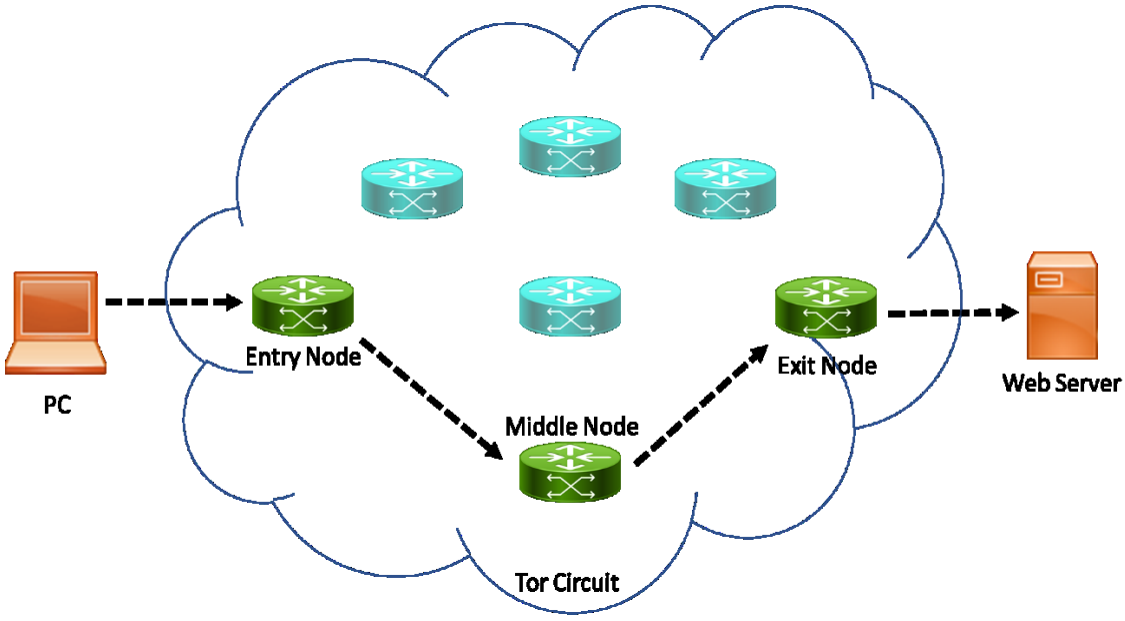

Figure 2.2 (a): Communication through Tor network

As mentioned earlier, a source computer selects some nodes from the list of Tor network database. To make it clear and understand the practicality of the Tor network, the following example explains the concept. Let's say there are two parties Alice and Bob (example names used in cryptography), where Alice wants to communicate with Bob over the Tor network [11]. Before Alice and Bob start communicating with each other, Alice's computer will select few of the random nodes from the Tor database. Essentially, to enhance the security level and build the anonymity, Alice's computer will negotiate in succession shared cryptographic keys with the selected nodes at each level. The main thing here is that the key negotiation process is in succession so that aside from the first node the rest of the nodes are not aware of Alice's involvement. Therefore, if three random nodes are chosen then the first node will be entry node, the second node will be middle node and the third node will be the exit node [8]. In this scenario, Alice would essentially get three session keys generated from these random nodes. Let's say she got session key 'k1' from entry node, then got key 'k2' because of negotiating with middle node and the key 'k3' after negotiating with the exit node. This session key 'k3' is also responsible for validating that the data coming into the Tor proxy is the same data being sent out of the exit node [10]. Now Alice will encrypt the message with each of these keys. For example, encrypting the message M with key K1 will generate a ciphertext C1. The second node will take the ciphertext C1 and re-encrypt with key K2 and generates C2. Similarly, the encryption process for ciphertext C2 with key K3 will generate ciphertext C1.

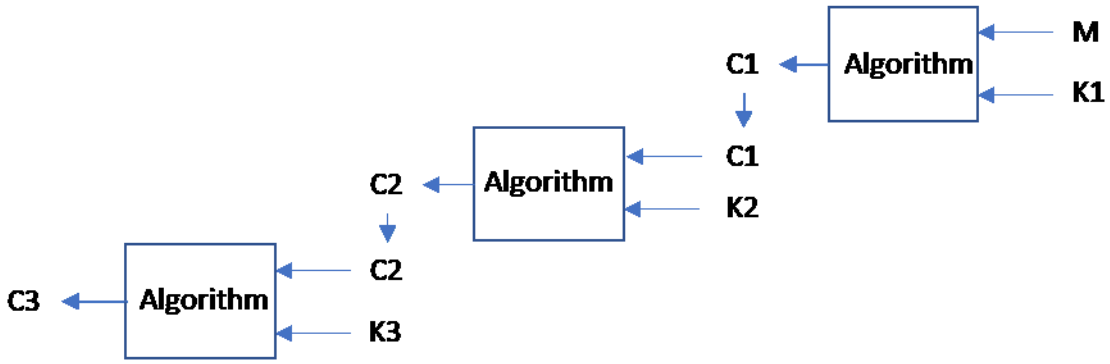

Figure 2.2 (b) Message encryption at each Tor node with different keys

Therefore, at each session of key negotiation, each node will think that it is communicating only with its predecessor node, but indirectly it will be communicating with the originator i.e. Alice. The technique behind the generation of the keys is the combination of Diffie-Hellman key exchange protocol [11] with some techniques of public key encryption after which Alice will be able to have three different keys. The connection drops unless the complete Tor relay circuit is established.

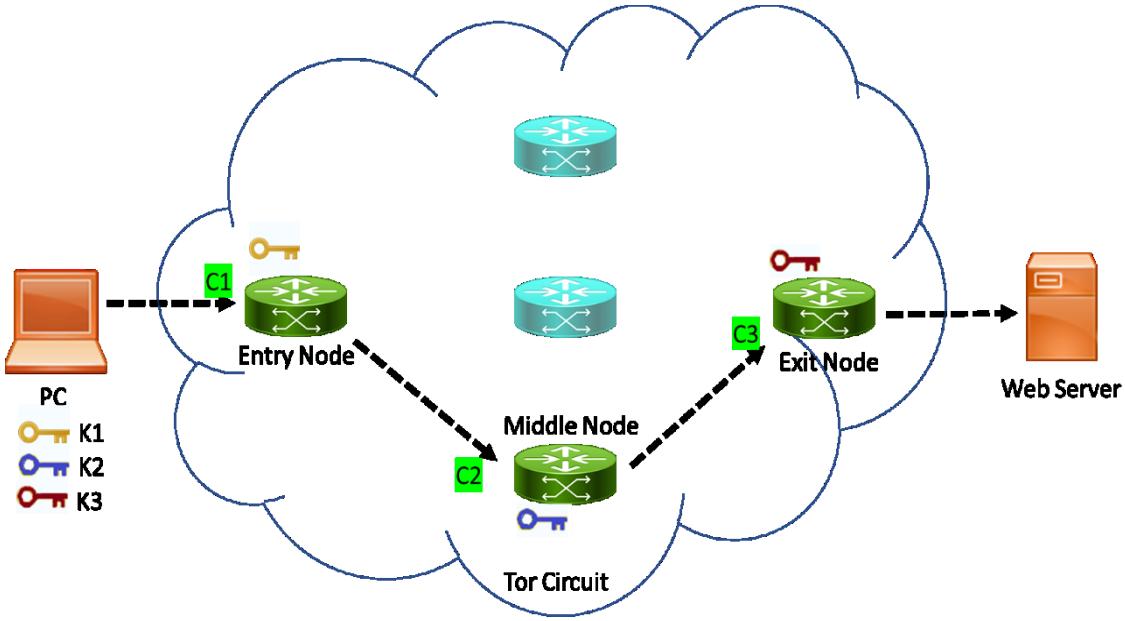

Figure 2.2 (c): Key origination at each chosen onion router

Once the Tor circuit is completed, it will look like a telescopic form of tunnels that unwrap much like the layers of an onion after which Alice will be able to send data to Bob. Now, when Alice forwards the data to Bob through the Tor browser, her computer will wrap the data into three layers of encryption using the same three different session keys generated before. Each node will unwrap one layer of it and forwards the packet to the next node. Here, the first two nodes will only see that they need to forward the packet to the next Tor node, but the last node will see that the packet is intended for Bob, so the packet will exit the Tor network and be forwarded to Bob [10]. Therefore, it also eliminates the traces of the path followed towards the destination, making it harder to diagnose the actual source of the packet and the path the data traveled. If Bob sends data to Alice, the same process is followed, but with a selection of different Tor nodes and each node will have different session key as well.

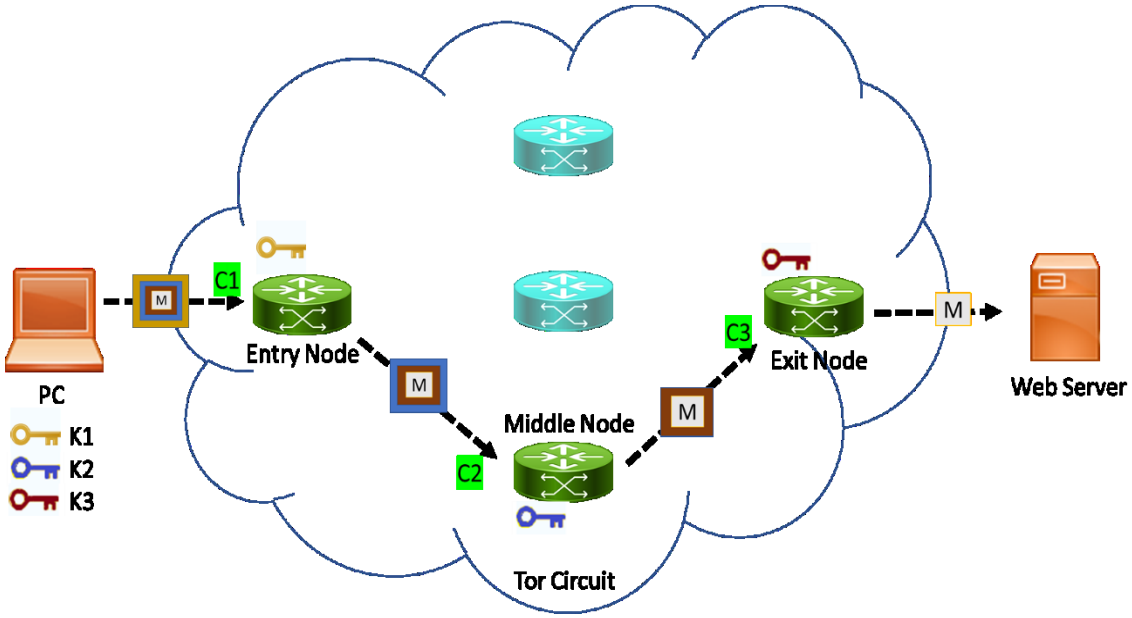

Figure 2.2 (d): Packet from source to destination

However, if the data sent to the Tor proxy from the computer is unencrypted, then it will also be unencrypted after leaving the exit node. So, if there is some sensitive information i.e. username/password etc. anyone listening at the exit node or on the link between the exit node and the destination, will be able to detect the information and may use it as a piece of data to figure out who you are. Similarly, if someone installs third-party applications e.g. multimedia, java etc. which may not be configured to use Tor proxy, then the server who is interested in identifying your identity, may request to load specific kinds of multi-media files, which may require your browser to bypass the Tor proxy and run the application over the regular Internet [15]. This way the user will no longer remain anonymous and the source will be easily identifiable.

2.3 Tor Hidden Service

As discussed earlier, Tor provides various kinds of services, from the web services to instant messaging services making it possible for users to stay anonymous. Other Tor users may connect to these hidden services using Tor rendezvous points without having knowledge about other networks [11]. The rendezvous point is a meeting point, which allows P2P network peers to find each other.

Initially, to get the hidden service known to the client, the hidden service needs to advertise its existence in the Tor network. To do that the server selects some random nodes, which act as introduction points and builds circuits to them. The server shares its public key with the chosen random nodes so that they act as an introduction point for the server. To keep the identity of the server anonymous, the link between the server and the introduction points are linked using Tor circuit, rather than direct connection making it hard for anyone to associate an introduction point to the server's IP address [8].

After selecting the introduction points, the hidden service assembles the descriptor, which includes the public key and the summary of chosen introduction points [11]. The hidden service than signs this descriptor with its private key and uploads the descriptor to the database. It produces the ‘dot onion’ address, further used by the client to reach the hidden service. The client gets this knowledge by connecting with the Tor database that has the descriptor uploaded by the hidden service. Once the client finds the descriptor of the ‘dot onion’ address, it will know about the introduction points and the public key of the hidden service.

The client also creates its own circuit with another random node, with which it shares its one-time secret (cookie), so that it acts as a rendezvous point. This rendezvous point will be helpful in establishing communication between the client and the server later. The client assembles an introduce message, including the rendezvous point and one-time secret and forwards it to one of the introduction points requesting it to be delivered to the hidden service [11]. The introduction point forwards this message to the hidden service, which decrypts the message and creates a circuit with the rendezvous point. After creating the circuit with the rendezvous point, the hidden service sends its one-time secret to the client in a rendezvous message and establishes a connection.

Once the connection is successfully established, the rendezvous point notifies the client about it. Now both the client and the hidden service can communicate with each other by using their circuits to the rendezvous point. The rendezvous point only relays the messages between the client and the service. One of the reasons for using the rendezvous point for communication instead of introduction points is that no single relay should appear to be responsible for the given hidden service [11].

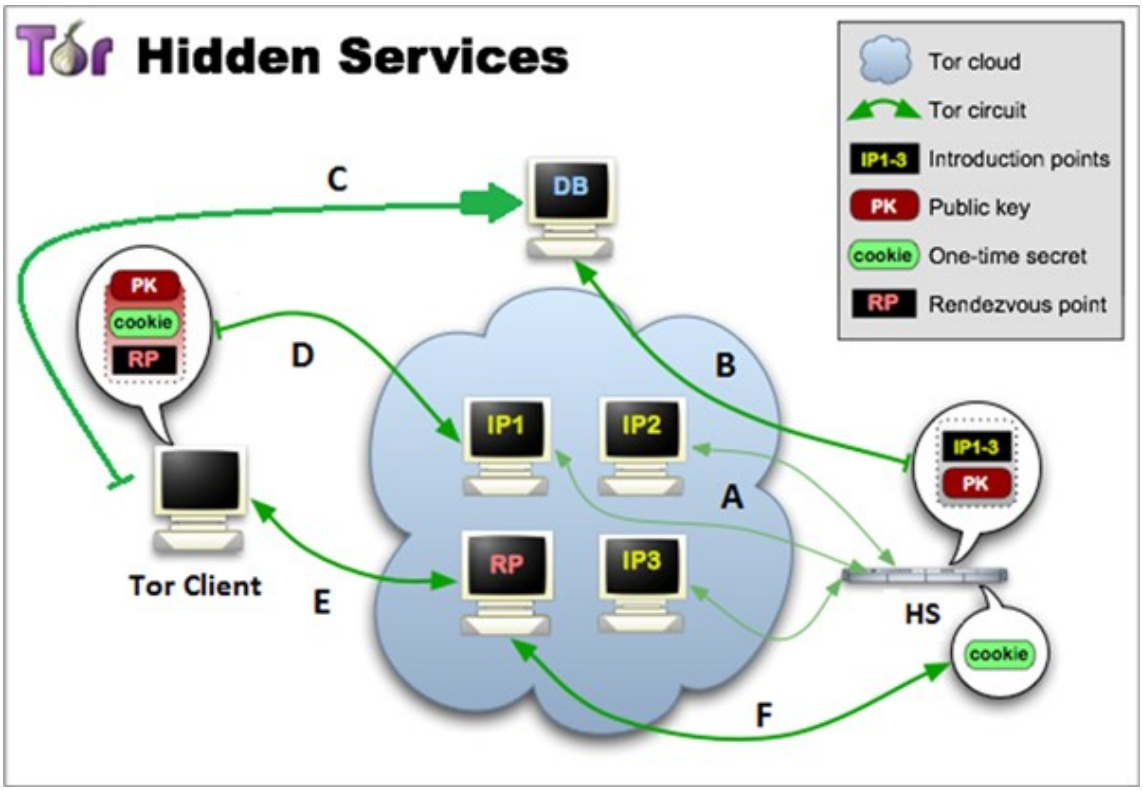

Figure 2.3: Tor Hidden Service [8]

2.4 Tor2web Proxy

Internet users who do not use or do not have access to Tor web browser bundle can still access the hidden services without using any Tor bundle software. Tor2web proxy [16] is an HTTP proxy that makes it possible for these users to access the hidden service 'dot onion' address without using Tor browser. The Tor2web service downgrades the anonymity of the user and lets them know by giving a warning message that the user is no longer anonymous and suggests downloading and using the Tor browser bundle if the user wants to stay anonymous. Like Tor, Tor2web also provides proxy based on the community integration [17]. Anyone can start and host the Tor2web proxy and relay the packets further to the network.

During the early stages of its development, Tor2web architecture was based on one domain, one wildcard digital certificates shared among trusted people [18]. Until now, the same architecture is being in use but there are some scalability issues in the current architecture.

- Because of single DNS, the service might be unavailable during maintenance or in case the service is down.
- There is only one digital certificate shared among only trusted people.
- A single person is managing the DNS server.

In the near future, Tor2web's architecture is going to have some changes and accommodates the scalability issues that are discussed above [18]. The targeted issues are as follows:

- There will be multiple administrators managing the network using multiple domains.
- There will be multiple certificates so that it can be shared among a group of people.
- It will support multiple DNS servers.

There are several features of Tor2web proxy and some of the current main features are listed below:

- It allows the blocking of illegal content.
- Keep the blocklist in hash format and not in the clear text.
- Only proxies the onion URLs and not the others.
- Requests are proxied transparently from the Internet to Tor hidden services.
- Contains standard template for errors.

In situations, where the user is unable to reach torproject.org, then the user can use an extra feature of Tor2web proxy called GetTor [18] which provides the user an alternative to downloading the Tor browser.

2.5 Invisible Internet Project (I2P)

Invisible Internet Project (I2P) which initially began in 2003 [19] is an open-source project like the Tor project and its main purpose is also to provide anonymity over the Internet, supporting secure and anonymous communication. Most of the project is written in Java (1.5+) but some of the third-party applications have been written in Python and other languages [19]. I2P was designed as a true "Darknet" and functions as a "network within a network" meaning that the outbound relays either do not exist or few of those which exist are rarely usable, so keeping the traffic within the network. I2P provides control over the tradeoffs between the anonymity, reliability, bandwidth usage, and latency to its users [19]. The integrity, security, and anonymity of the system are never compromised by I2P and dynamic reconfiguration support of the network helps to keep the system safe and secure from various vulnerable attacks.

I2P, unlike other anonymous networks, provides end-to-end anonymity. It does not provide anonymity by just hiding the sender and not the receiver or the way around, but it hides the identity of both the sender and the receiver. Therefore, the peers communicating with each other using I2P network are anonymous and unidentifiable to each other and to third parties. Currently, I2P allows anonymous web publishing or hosting as well as HTTP proxying i.e. the publishers can host their websites using the I2P network as well as the clients who can browse the Internet using an I2P proxy. This will not only keep the user's information safe but also the communication happening between

them will be safe from leaking. The traffic over the Clearnet is quite likely to be monitored and prone to malicious attacks. For example, even if the user is using HTTPS to browse the website, the middleman or the hacker and many other middle entities may see some information that is exchanged between the client and the server and that information may be helpful for the middleman to monitor your web surfing or use of the Internet.

There are three key concepts to understand the operation of I2P. The first concept is “strict separation of the router and the endpoints”. The concept behind it is to hide the details of the user and the server. The second important concept is the use of “tunnel”. The tunnels used in I2P are uni-directional tunnels. It uses two separate tunnels, each for sending and receiving a message. The tunnel is made through an explicitly selected list of routers [12]. I2P uses layer encryption concept similar to Tor but it layers like garlic. At each point of the router, one layer is decrypted and the rest of the packet is forwarded to the next router’s IP address. Each tunnel has two points, one is start point (gateway) and the other is end-point. Every tunnel can get a message from any other user at its start point and further it delivers it to the end-point. The third critical concept is the use of I2P’s “network database”. There are two types of metadata that are carried; “routerInfo” and “leaseSets” [19]. The “routerInfo” has information about the list of I2P routers and leaseSets has the information about the destinations. These are the three main concepts that I2P uses in combination to get the users connected and share the information with each other.

3. Transport Layer Security and its additive features

SSL, developed by Netscape, was one of the early cryptographic protocols used to secure communication over computer networks. Due to serious security flaws in SSL 1.0, the version was never published (was not introduced commercially) but later in 1995 SSL 2.0 was released [20]. There were also several flaws in that version of SSL, which ultimately led to the design of SSL 3.0. As of 2014, SSL 3.0 is considered insecure, as it is vulnerable to various online attacks. TLS was first defined in 1999 by IETF as an upgrade of SSL 3.0 [21]. The latest version of TLS is 1.3, which is still a working draft and is incomplete but includes many differences from 1.2. This thesis work uses TLS version 1.2, which has TLS SNI extension.

TLS provides privacy and data integrity between the two communicating applications. The key difference that makes the TLS more secure and efficient from SSL is the support of newer security algorithms, supported cipher suites, message authentication and key generation through the process of handshaking [22]. TLS is a widely deployed security protocol and has its applications in Web browsers, VPN connections, VoIP and other data transferring applications [22].

There are two primary layers of TLS protocol, the TLS record protocol, and TLS handshake protocol.

3.1 TLS Record protocol

The record protocol works at a lower level, layered [22] on top of some reliable transport protocols e.g. TCP. The protocol provides security to the communicating connection making sure to keep the privacy and reliability of the connection. For encryption of the data, a symmetric key encryption [22] technique is used e.g. AES, RC4, etc. Since the protocol uses symmetric key encryption, the keys generated for the connection are unique for each connection and based on the secret negotiated by TLS handshake protocol. Reliability of the connection depends on the integrity check using the keyed MAC. For MAC computation, secure hash functions are used e.g. SHA-1, etc.

The record layer function can be called at any time after the handshake process is finished [22] when there is need of communication. The record protocol encapsulates various higher-level protocols such as the TLS handshake protocol.

3.2 TLS Handshake protocol

Before the actual communication starts and the application protocol starts sending or receiving the data, the protocol allows the client and the server to authenticate each other, negotiate the algorithms and the cryptographic keys to be used. The TLS handshake protocol [22] provides three basic properties of connection security:

- Asymmetric cryptography i.e. RSA, DSA, etc. used for authentication process

- Keeping the shared secret secure, i.e. the negotiated private key is unavailable to eavesdroppers or attackers.

- The communication is reliable so that the message is exchanged without any alterations.

During the initial stage, when the client wants to start a communication with the server, the client sends a 'Client Hello' message to the server. The 'Hello' message includes cryptographic information such as version number of TLS/SSL, the cipher suites supported by the client and a random byte string used for computation. Once the server receives the 'Client Hello' message, it replies with the 'Server Hello' message. It includes the information about the particular suite it chooses and the TLS version number. The server also sends its certificate including the public key along with the 'Hello' message to the client.

Client upon receiving the certificate from the server verifies it and extracts the server's public key. The verification process may include, checking of the digital signature, the certificate chain, and validity period of the certificate, including the activation and expiry dates and the cancellation status of the certificate. The client uses the public key to encrypt a new key called 'pre-master key'and sends it to the server. The server decrypts the 'pre-master key' with its private key. At this stage, if the client receives the "client certificate request" from the server, then the client sends a random byte string encrypted with the client's private key along with the client's digital certificate. The server verifies the client's certificate upon receipt.

The client sends an encrypted message to the server requesting the server to check the message by decrypting it and verify if it is up to the specifications. This is the 'finished message' from the client side indicating that the communication onward will be encrypted with the shared secret key. The server decrypts and verifies the message and if everything is normal, the server replies with the 'finished message' too, requesting the client to decrypt and verify the message content. The server also indicates that the communication exchanged onward will be encrypted with the shared secret key. The communication between the client and the server is therefore, encrypted and authenticated for the whole session until either side terminates the session.

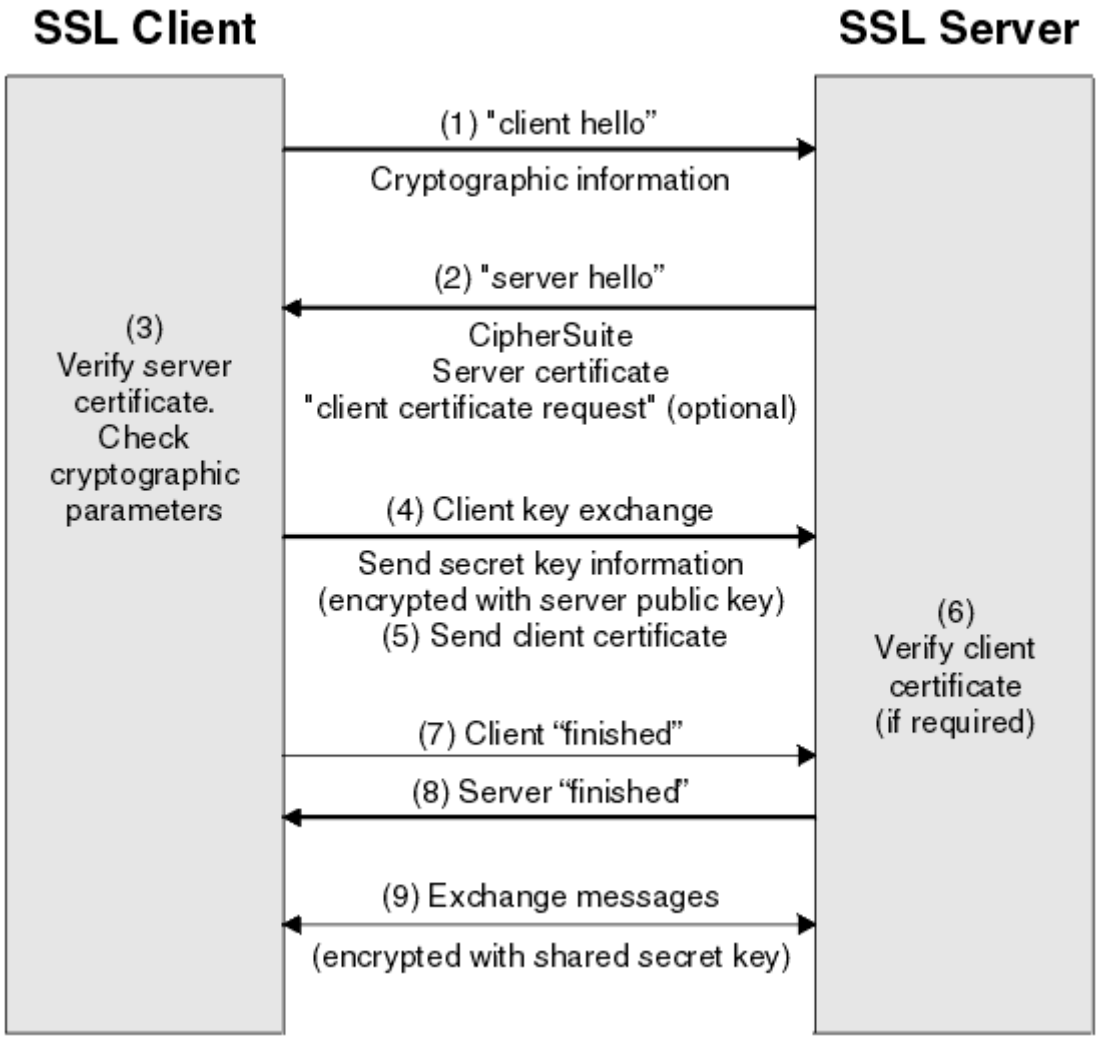

Figure 3.1: The complete overview of the TLS handshake protocol [22]

The TLS handshake itself uses asymmetric encryption i.e. one 'public' and one 'private' key. The message is encrypted using public key and decrypted using private. Since asymmetric encryption has higher overhead, therefore, it is not suitable for providing real-time security. Thus, the asymmetric encryption is used during handshake only to allow setting up and exchanging newly created 'shared key'. Once the session is established [23], the session itself uses symmetric encryption which makes the secure connection feasible in real-time.

3.3 HTTP Strict Transport Security (HSTS)

HSTS is a web security feature that helps to protect websites against different attacks such as man-in-the-middle attack, cookies hijacking and protocol downgrade attacks [24]. This simple website security feature forces the websites to communicate with servers through HTTPS connection instead of HTTP only. HSTS is an IETF standard track protocol and is specified in RFC 6797 [25]. The specification was submitted as an Internet draft on 17

June 2010, but the specification name was altered from “Strict Transport Security” to “HTTP Strict Transport Security” because the specification applies only to HTTP [25].

Since, HTTP is a simple transport protocol, which does not provide any encryption for the payload it takes from source to the destination, so any data or web page delivered over HTTP connections will expose users to security risks. The risks may include the manipulation of the data or information of the sender and or receiver, stealing the login credentials information, forwarding or redirecting to insecure or malicious websites and many others. This kind of interception cannot be detected since the HTTP response looks the same as the genuine response [26].

HSTS forces all the data to be passed through HTTPS connection instead of plain text HTTP, thus making the entire channel encrypted before sending the data and less possible for the attacker to attack the channel and steal the information during the transmission.

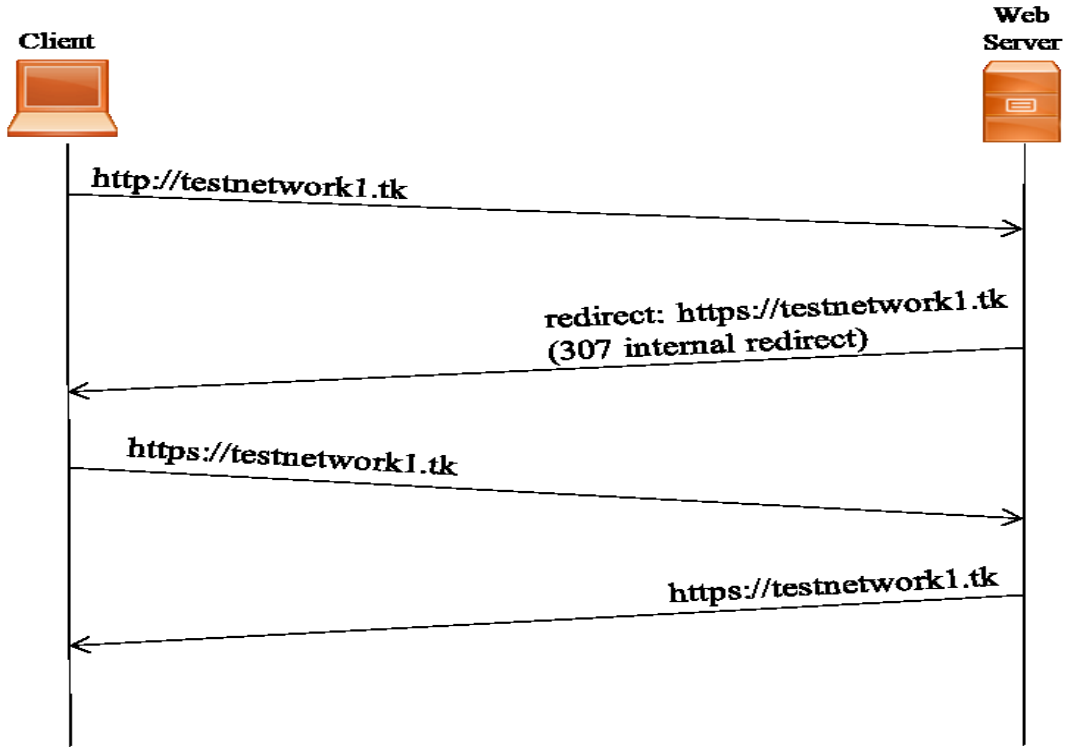

Figure 3.2: HTTP Strict Transport Security (HSTS)

To enable HSTS on a server the following HSTS response header must be added in an HTTPS reply:

For example:

```
Strict-Transport-Security: max-age=16070400; includeSubdomain
```

The max-age parameter is in seconds and this specifies the time the browser should connect to the server using the HTTPS connection. 'IncludeSubdomain' specifies that the browser uses the HTTPS connection for the existing and the future subdomains.

Now, when the browser accesses the website, the server replies with the HSTS response header. The web browser will then remember to use the HTTPS connection for the specified age i.e. max-age. During the specified max-age period, the browser will even upgrade the HTTP request to the HTTPS so the website is accessible on a secure link. For example, if the user types http://www.testnetwork1.tk or simply writes testnetwork1.tk, the browser will upgrade it to https://testnetwork1.tk. Once the specified max-age expires, the browser will start accessing the website normally through HTTP, unless the user specifies HTTPS in the link.

Almost, all of the popular web browsers support HSTS and some maintain the HSTS preload list that automatically informs the browser which websites are accessible through HTTPS. This is possible by adding a parameter 'preload' to the header and then adding the domain to the list.

```
Strict-Transport-Security: max-age=16070400; includeSubdomain; preload
```

Beside solving many problems, the most important security vulnerability that HSTS can fix is SSL-stripping or protocol downgrade attack [27]. Such attacks transparently convert the HTTPS connection into plain HTTP connection. Since many websites do not use TLS/SSL; therefore, there is no prior knowledge whether the plain HTTP is due to an attack or simply because of not implementing TLS/SSL on the website. Also, there are no warnings presented to the user during the downgrade process [27].

HSTS fixes this problem by informing the browser to always keep the connection HTTPS to the server. Although, this will limit the attack to some extent but has a few limitations, because if the user visits the site for the first time with HTTP, then the attacker may intercept the connection and steal the information. Most of the popular browsers have provided a solution to this problem by including a 'pre-loaded' list of HSTS sites but unfortunately, it cannot scale to include all the websites on the Internet [27]. In addition, since the HSTS is time-based, therefore it is sensitive to the attacks involving the shifting of time of the victim's computer by using false NTP packets.

3.4 HTTP Public Key Pinning (HPKP)

Until now, the solution to minimize attacks on websites was to force browsers to use HTTPS. This technique discussed was HTTP Strict Transport Security (HSTS). Besides forcing the browsers to use HTTPS, it also mitigates Man-in-the-Middle attack. However, what if the Certificate Authority (CA) is compromised and the attacker starts miss-issuing the certificates for the websites; the browser will accept it implicitly without creating any

problems, as for the browser everything seems to be ok and valid because that CA will be in its list of trust. If the attacker succeeds in doing this, it can create a lot of issues and it is very hard to provide a real-time solution. Although, there are certain methods discussed in the Certificate Transparency topic to overcome these issues in real-time, sometimes it might take months to diagnose the problem.

HPKP provides the solution for mitigating the MIM attack by eliminating the generation of bogus and rogue certificates. It is a security policy which is delivered via HTTP response header [28] just like HSTS and CSP (Content Security Policy). The main purpose of the HPKP is to allow the host to deliver a set of fingerprints to the User Agent (UA) about the cryptographic identities (TLS certificate) it should accept in the future. This way, it will accept only those identities fingerprinted and allows them going forward. It may reject any change found at any point. To make it more secure, the better way is to include the fingerprints of your current TLS certificate and at least one backup [29]. Certain security analysts suggest backing up the Certificate Signing Request (CSR) fingerprints in order to avoid purchasing of the backup certificate. So, at any stage, if the certificate's key is ever compromised, this CSR could sign a new public key. This can be done by creating the CSR with a brand-new RSA key pair and must be stored securely offline [30]. This way, it is easy to switch to the new certificate, since the fingerprint of the CSR is already in the HPKP header. Therefore, if the hacker compromises the CA and issues a rogue certificate for any domain, the browser that had an HPKP header in a response will not accept it.

The broader example could be a bank which always has a certificate from some authorized CA, say CA company A. In today's world, any CA authority can create certificate for that bank, say CA Company B or CA Company C since they all have a valid trust path and the browser will accept it without issuing any warnings. But, if the bank implements HPKP and pin the first certificate issued by (company A), then the browser will not accept other certificates issued or generated by other CA authorities, like in this case CA authority B and CA authority.

3.5 HPKP Response Header

HPKP response header has the following format:

```
Public-Key-Pins:    pin-sha256="base64==";    max-age=expireTime    [;
includeSubDomains][; report-uri="reportURI"]
```

The Pin directive comes from the certificate that has information like the subject, issuer etc. It binds the public key with the identity. The identity in our case is testnetwork1.tk and the syntax of the Pin directive is as follow:

```
pin-sha256=\"YLh1dUR9y6Kja30RrAn7JKnbQG/uEtLMkBgFF2Fuihg=\";
```

```
pin-sha256=\"NKjQ9MVvj48l1/6nm5IqyV/WGZRAiuylkJo+2R0GetA=\";
```

The first one is the Public Key and the other is Backup key, which is important to have in case the private key is compromised at any stage, then it can generate the new certificate. It is mandatory to keep the backup key offline and not at the same place where the private key is stored.

According to the rules, the User Agent must ignore the pin-directives with tokens naming a hash algorithm it does not recognize [29]. The pin-directive name is case sensitive according to processing rules set by IETF.

This max-age directive indicates how long you want the policy to be enforced, or in other words, during this period the UA should consider the host as a known pinned host. Initially, it is better to keep the max-age value smaller so that in case of incorrectly installed headers, or the invalid policy, the UA only remember it for a small number of seconds.

In this thesis work, initially max-age 30 seconds is used and once it is tested and confirmed that the policy is working fine, then the max-age directive value can be increased to max.

The syntax for the max-age directive is as follow:

```
max-age = 30
```

Beside these two directives, there are two more directives that are optional and are as follows:

This includeSubdomain directive is an optional directive and if present and applied, tells the UA that the pinned domain and subsequent subdomains contain pinning policy. This ensures adequate security for domain level cookie.

```
includeSubdomains';
```

The report-URI directive is also optional and it indicates where to post the report in case of PIN failure. The good idea is to set different host URI in the report-URI directive because if the domain is under attack there will be no valid connection between the user and the server due to MITM. The RFC also indicates that the client can try to send the reports later in case of report failure [28].

```
report-uri="https://report.testnetwork.tk"';
```

4. TLS Certificate Authorities and Domain Name Validation Practices

Certificates are an essential part of today Internet and they are the proof that the owner of a certain certificate is authentic and has passed different levels of examinations/challenges. This chapter focuses on the CA that provides the certificates to the domain owners and organizations. Besides that, it also discusses the popular digital certificates that are a part of this thesis work as well as the process of issuing the certificates to owners.

4.1 Certificate Authority and Digital Certificates

Security and privacy are essential parts of daily life and without it, the lives of the people will always be at risk and danger. Similarly, over the Internet, security also plays a vital role in making and enhancing the trust relationship between the user and the organization (website). Therefore, the trusted entity (authority) that deals with such matters is known as Certificate Authority. It is a trusted entity that issues an electronic document, which helps in identifying the identity of the digital entity over the Internet. The issued electronic document is thus known as a digital certificate and is an important part of Public Key Infrastructure (PKI) and secure communication [31]. The certificate may typically include the owner's public key, expiration date of the certificate and other related information about the owner. Operating Systems (OSes) and Web Browsers maintain the list of trusted CA root certificates to verify certificates that a CA has issued and signed [32]. CAs issues millions of certificates each year to help secure critical information such as online transactions and information about the sender and/or receiver.

Now, one can think about who decides a CA to be trusted. The CA can be trusted through an authorized process called CA membership program which is operated by browsers, operating systems and mobile devices [32]. In this program, the CA must meet detailed criteria to be accepted as a member [31]. Once the program accepts the CA as a member, the CA can issue SSL certificates transparently trusted by the browsers, operating systems and other devices relying on the certificate. The trusted relationship depends on the operational age of the CA, the longer the CA is operational, more browsers and other devices will trust the certificates issued by that CA. There are several CA and every device, browser or operating systems have their own policies of selecting the CA and make their own list of authorized CA. For example, Google Chrome web browser uses the certificates that are included with the Operating System [32]. Similarly, Firefox has a list of its own default authorized CAs e.g. Certum CA, Buypass Class 2 CA 1 and many others.

4.2 Let's Encrypt

Let's Encrypt is a certificate authority that issues an authorized electronic document for securing websites for free. It uses an automated process to issue the certificate, rather than the manual procedure thus eliminating the complex process of validation, signing, installation and renewal of the certificate. The aim of the project was not only to eliminate the complexity of the system but also to provide TLS encryption at a very low cost, which would help in securing more and more websites on the Internet. Thus, now execution of only two commands would set up HTTPS encryption and acquire and install the certificate on the website in Linux web server [33]. The following is a figure that explains how the Let's Encrypt authorization works.

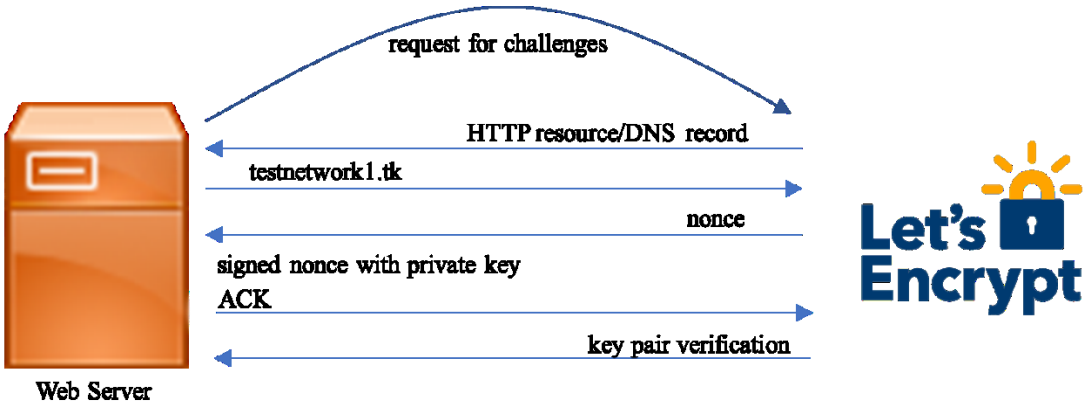

Figure 4.1 (a): Let's Encrypt Authorization process

The agent authorizes the key pair, and once authorized, the rest is simple. To obtain the certificate for the domain "testnetwork1.tk" a Certificate Signing Request "CSR" is generated by the agent with a specified public key and sent to the Let's Encrypt CA. Along with the public key, the agent includes the signature by the private key, so that the Let's Encrypt CA knows it is an authorized key pair. Once the CA receives the request, it verifies the signature and if everything looks good, the certificate is issued for "testnetwork1.tk" with the public key from CSR.

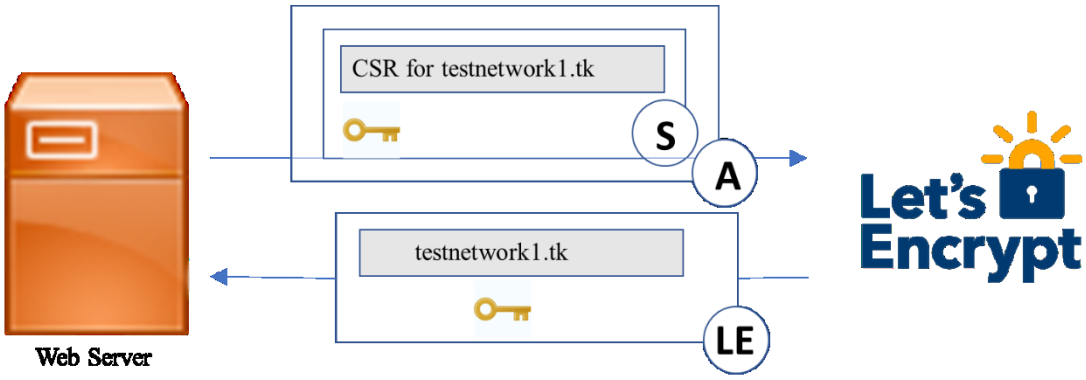

Figure 4.1 (b): Assigning certificate after authorization

Currently, Let's Encrypt offers free domain validated SSL certificates but not the Extended Validation (EV). Domain validation is the lowest validation type, showing that the certificate holder has control over a particular domain [34]. Extended Validation (EV) certificates, on the other hand, are used for securing information that is critical and sensitive in nature and requires enough security to keep it safe from unauthorized users and attackers. This type requires a stringent set of security checks and is therefore implausible to offer for free.

Let's encrypt certificate authority is trusted by all major web browsers and cross-signed by IdenTrust [35]. It allows for anyone who wants a domain validated SSL certificate for HTTPS to easily have one free of charge by entering only a few commands.

4.3 Certificate validation process

Security is an essential part especially when it comes to websites and the Internet. Several types of websites are in use around the world nowadays. Some of the websites are basic ones i.e. blogs which are only used to convey information to the reader. Some of them provide online shopping services to clients and some hold very critical information about the user i.e. banks etc. Since, the level of each website varies based on the information it contains and the data it shares, the level of protection required by each website varies.

Therefore, by looking at the requirements, there are several different types of certificates, each provides a certain level of protection and security to the websites and its clients. The level of verification also varies depending on the type of certificate requested.

4.3.1 Domain Validation

Domain Validated SSL certificates provide the basic level of security and therefore are the cheapest among other certificates. In this validation process, only the domain name of the website is authenticated and processed based on very little information gathered from the individual or the owner for authenticating the website/domain name [36]. Each certificate authority has its own rule for validating the domain name. Some authenticate the process just on the basis of user email address and some via call or through some other alternative methods.

Let's Encrypt provides domain validated certificates [37]. Before issuing the certificate for the certain domain name, the Let's encrypt client queries some of the challenges from the individual to prove the information provided including country, location, zip code, and an email address along with additional question of choosing between the Easy and the advance usage (HTTP & HTTPS both or enforcing HTTPS) respectively. The email address is required for future communication in case of lost key recovery and/or notices of the validity of the certificate.

4.3.2 Extended validation

To enhance the trust level of the users and make them sure about the legitimacy of the website they are visiting, the validation process can be carried out to the utmost level of validation; the extended validation. Not only, the domain name is validated during the process of authentication, the whole process is analyzed meticulously [38]. During the authentication process, the authority may require various documents confirming the legitimacy of the business beside the details of the certificate request. The Certificate Authority takes its time to verify the provided details and documents and issues the EV certificate to the owner of the domain upon clearance.

For issuance of EV SSL certificates [38], the CA verifies the following measurements:

- The domain owner must have domain control. The CA could verify WHOIS registration record for the domain.
- The CA also verifies the applicant of the certificate. The applicant should be the same who has signed the contract.
- The CA matches the name of the company provided in the certificate with the official company's name.
- Besides the verification of the company's name, CA also verifies the address of the company in the state's record with the one mentioned in the certificate.
- Finally, before issuing the certificate, CA also verifies the contact information like telephone number and email address provided by the company in the certificate.

To get the EV SSL certificate, one has to go through the strict authentication process of the CA and should provide complete details to the CA, so that the CA could allow the users to trust the website they are visiting.

4.3.3 Organization validation

The owner of the domain, who may be running a small organization, can also use more advanced validation services than simply the domain validation service, which helps the owner of the organization to promote the organization name that helps in marketing the business [36]. Besides the details about the domain name, the organization's details are provided out by the owner, during the process of certificate authentication challenges. This acquired information about the organization is then embedded (the organization name) just beside the green padlock in the address bar when the user visits the website.

The CA may verify the business through different means (by contacting the local authority) and if the authority fails to get the verification of the business then it may ask for further details from the owner of the business. Further information may include the

Business license or a tax number or bank statement etc. The personal business name cannot be used if the verification of the business fails.

4.4 Certificate Transparency

Enhancements in modern cryptography from the past few years have strengthened modern browser capabilities of detecting malicious websites that are provisioned with fake SSL certificates [39]. However, the modern cryptography is not capable of detecting any malicious websites, provisioned with mistakenly issued SSL certificates, or certificates maliciously acquired from an unimpeachable CA. In these situations, it is hard for the browser to detect any malicious and bogus information about the CA. The CA appears to be authentic and in good standing condition. Therefore, it gives the users an impression that the website is authentic and the connection is secure.

Since there is currently no effective way to monitor or audit the certificates in real time [40], it takes sometimes a week or a month to possibly detect the maliciousness or bogusness of the certificate. The effect of these issues can be far ranging and harmful.

Certificate Transparency helps in providing a solution to these certificate-based threats by making issuance and existence of the SSL certificates open to the scrutiny by domain owners, CAs, and domain users. CT has three main goals that are listed below:

- Visibility of the certificate to the owner of that domain [40]. Transparent auditing and monitoring system that lets the domain owner or CA determine the maliciousness or wrongly issuance of the certificate.
- To protect users from being duped by malicious and mistakenly issued certificates.

Certificate Transparency is an open framework [41] allowing anyone to build or access the basic component of the framework making it possible to achieve the above-mentioned goals. CT open framework consists of three main components [41], which are described below:

- **Certificate logs** are simple network services that maintain cryptographic assurance, auditing and append-only logs of the certificates. Anyone can submit these logs but certificate authorities are foremost submitters. Like submitting the logs, anyone can query the log for cryptographic proof, which can be used to verify logging of the particular certificate and/or that the log is behaving correctly.
- **Monitors** are running servers that keep watch on the validity and authenticity of the certificates. These servers periodically contact the log servers and keep a check on malicious certificates. For example, monitors can tell if any unauthorized certificate has been issued for the domain or if there are some issues with the certificate i.e. unusual certificate extensions or strange permissions.

- **Auditors** are software components that are used to inspect the status and authenticity of the logs and the certificates. They have two functions to perform:
    - They have to check the behavior of the logs i.e. verify if the logs are working in a correct manner and are cryptographically consistent. If there is any ambiguity in the logs, then the log will need to explain it or otherwise will face a risk of being shut down.
    - They should check the availability of a particular log in the log file. This is particularly important because Certificate Transparency framework requires all the certificates to be registered in the log [41]. If the particular certificate has no information in the log file, then the client considers it as a bogus or suspected certificate and the TLS client may refuse the connection with such websites.

  Auditors can be standalone or can be a secondary function of a monitor. Auditors and monitors can also communicate with each other in order to exchange information about the logs. This helps both to detect forked logs [42]. A single person or a certificate authority can run the Auditors, but it is more likely the CAs that run the bulk of all auditors.

All these components together, create an open framework that provides somehow the observance of the authenticity of the newly issued and existing SSL certificates in real time.

Certificate transparency when implemented helps in providing protection against several certificate-based threats, including wrongly issued certificates, misused certificates, and rogue certificates. The effects of these threats are far ranging and may increase financial liabilities for the owners and users. Because of these, Internet users may face several attacks such as web spoofing, man-in-the-middle attacks etc.

Certificate Transparency provides solutions to these problems and secures Internet users from hijacking. Because of its open framework feature which allows publicly run monitors and auditors. CT provides several benefits over the current SSL certificate system. These benefits include:

- Early detection of wrongly issued or malicious and bogus certificates.
- A faster solution to the problem as soon as the malicious certificate is detected.
- Transparent TLS/SSL system.

As a solution, CT provides security from the top to the bottom [41] of the pyramid i.e. from the CAs all the way down to the end users, making the connection more secure and reliable and less prone to attacks and thus allows safer browsing for the users.

5. Experiments using Tor SNI proxy

This chapter describes the work carried out throughout this thesis. Some of the figures in this chapter describe the working of components theoretically but their behavior has also been checked with practical implementation and analyzing the result.

5.1 TorSNIP

As the earlier topic discusses the significance of Tor and shows how drastically Tor has increased its number of users in the previous few years, but still there is a huge number of users who do not use Tor. One of the many reasons is that they cannot install a separate tool for accessing some limited content that is available on Tor's hidden service and even if they want to access any Clearnet domain, they had to wait a little bit more as compared to regular Internet surfing. There can be many other issues as well; for example, the domain name confusion. Onion addresses are quite different from regular URIs. One of the services discussed in the early topics is Tor2Web proxy that helps in providing the hidden service content over the regular Internet browser. During the previous research, it has also been observed that while accessing the Tor hidden service from some other means e.g. Tor2Web in this scenario except Tor browser bundle, the communication is not found to be end-to-end secure. It is quite evident that the eavesdropper observing the channel, clearly sees the information exchanged between the sender and the receiver. As a result, the user's requests forwarded through Tor2Web instances are at risk of different attacks.

The main reason behind this issue is that when an intermediate proxy server (Tor2Web in current scenario) forwards the user's request to the targeted hidden service, it decrypts the HTTPS packet and creates the onion circuit. Decrypting the HTTPS packet and relaying that to the targeted hidden service would lead to insecure communication. It is the same as two parties communicating with each other openly, such that anyone who wishes to hear their communication can easily listen to them. Keeping in mind the users, who are not using the Tor browser bundle and still get access to the hidden service, without compromising on the security of the data, could require a potential solution to the problem.

Rather than decrypting the HTTPS packet at the intermediate node, it is better to make an HTTPS connection directly to the onion site (HS). Server Name Indication (SNI) can be the solution to this identified problem. The purpose of this SNI SSL header would be to provide the name of the server during the TLS handshake process. Once the server name is identified during the handshake process and the SNI header indicates that the user is trying to reach the hidden service, the packets will be further forwarded to the backend

hidden service, which will have an SSL listener there. This way the user who is not using the Tor browser bundle would be able to reach the targeted hidden service using the Clearnet domain. The communication, therefore, will be end-to-end encrypted and the eavesdropper will not be able to identify communication exchanged.

Server Name Indication (SNI) is an extension of the TLS protocol and its purpose is to indicate the 'hostname' to be contacted before the TLS handshake process begins. This way, a single IP address may host multiple domain names with multiple certificates. This broadens the concept of single IP, single domain, and given a new concept of single IP, multiple domains, which makes it cheaper to have multiple domains.

In a traditional TLS handshake process (a), the client requests a certificate from the server. Once the client receives the certificate provided by the server, the client examines the certificate and matches with the one it was looking for. If a match occurs, the connection proceeds normal, else, the client's browser shows a warning message of the mismatch certificate and the connection may terminate. With the addition of the SNI header (b), now the client may request the specific certificate from the server. For example, if the IP address (server) is hosting three websites say 'site1', 'site2', and 'site3', the client will specifically mention the site it is requesting from the server. The server will forward the specified certificate to the client. If the received request has no name mentioned, the server may forward the default certificate to the client.

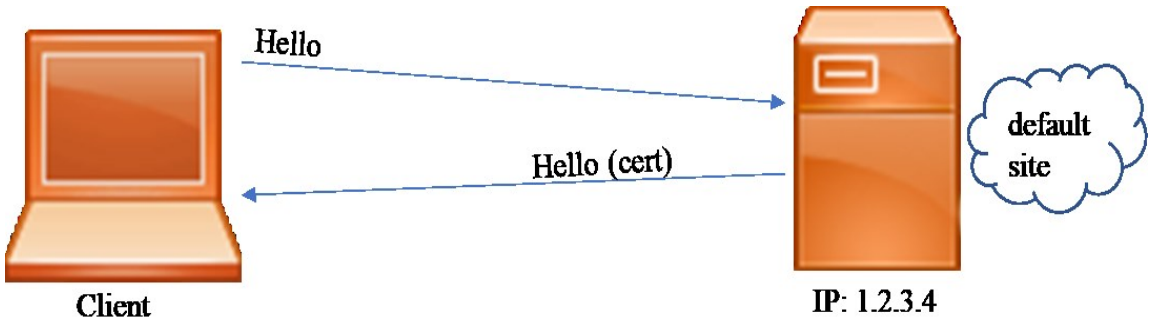

Figure 5.1 (a): Certificate request by client PC (without SNI)

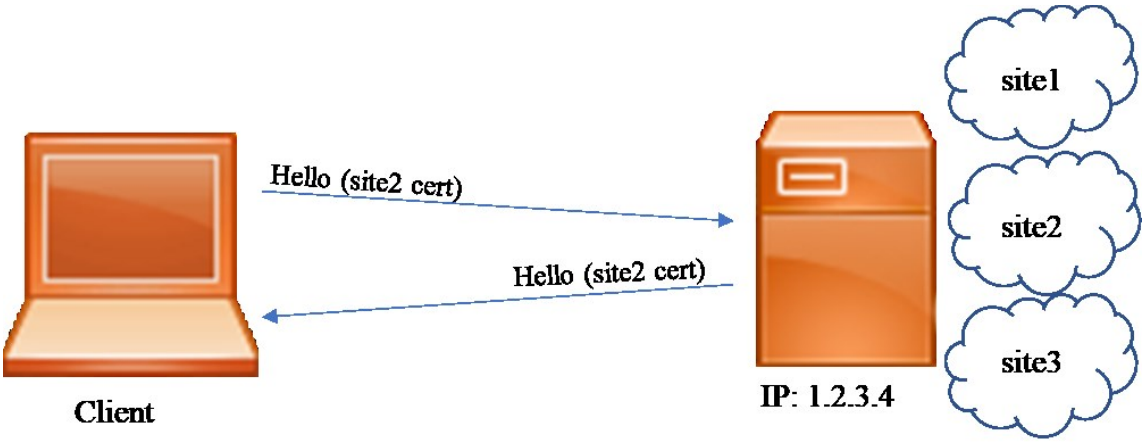

Figure 5.1 (b): Certificate request by client PC (with SNI header)

5.2 Experiments Setup

To accomplish the goal, the work is divided into three experimental tasks. Each experiment is linked with its successor until the final result is achieved.

5.2.1 Experiment 1: Basic Setup with Multiple Virtual Hosts in Apache

The literature related to the thesis topic i.e. Tor, HS, and Tor2web is reviewed during the early stage of the work. The literature was more about the working of Tor and HS as well as the basic configuration of these. The tools are chosen during this stage on which the actual implementation had to be carried out. Besides this, the knowledge related to the web server, domain name, and client/server communication has been gained. This has made the base of the work and cleared the basic concepts about the Thesis. Once done, a block diagram has been made to make the picture of the work clearer as shown in fig (a). Then a small prototype has been made to test the basic functionality.

Since the requirement of the thesis work is to have a client and server (HS) interaction using the Tor network, so, for that, a separate VPS has been reserved (a server with a static IP). To make use of the VPS, a domain name is registered and the DNS pointed towards the VPS. Initial client/server interaction scenario uses Apache web server. Now, the default Apache web page is accessible over the regular Internet.

To have multiple domain names on a single IP address, Apache offers Virtual Host support. Apache offers VH with different features for example; Name based VH, IP based VH etc. In this thesis, Name based VH with different ports has been used i.e. different domain names are on different ports.

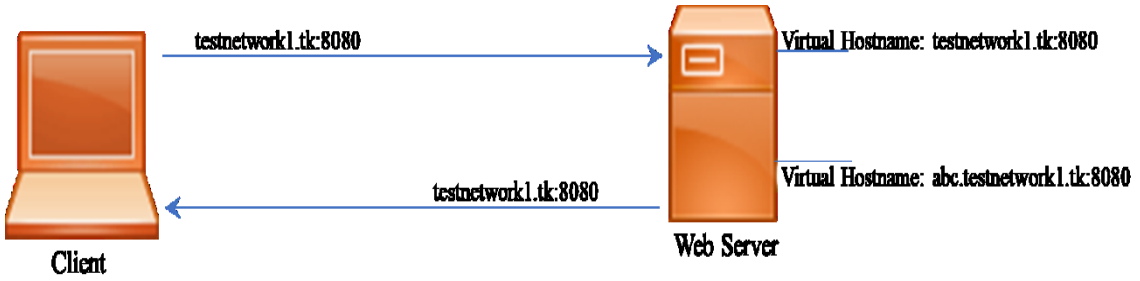

Figure 5.2: Apache Virtual Host

5.2.2 Experiment 2: TLS SNI proxy testing with Apache

As the main theme of the thesis work is to get access to the HS from the regular Internet without compromising security at the intermediate proxy, so, for that Tor's HS is required to be configured, which is done and tested during this stage. Since Apache's VH is working now, Dlundquist SNI proxy code is added to the implementation [43]. The default SNI configuration file is modified and is scripted according to the requirements of the project, shown in the appendix.

Apache's default ports 80 and 443 have been disabled after running the SNI proxy code. The SNI proxy at this stage listens on port 80 and 443, which forwards and receives the user's request using these ports. For example, if one domain name is configured on port 8080 and the other is configured at port 8090, the SNI configuration script will have their names with the port numbers, which will listen to the user's request on the default port 80 and forwards this request to the respective nameserver and port number. The client will be able to access the domain names at the default ports 80 & 443.

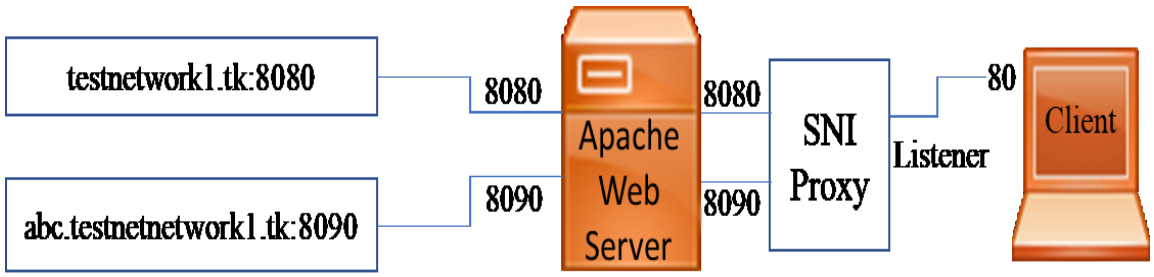

Figure 5.3 (a): SNI proxy is handling HTTP traffic at port 80

To have HTTPS on each of the domain names, a digital certificate is requested from Let's Encrypt CA which provides the authorization that the visited domain(s) is legitimate. One thing to be noted here is that to have a certificate for the domain, it must have its VH in Apache. If there is no VH found in the Apache, the certificate will not be issued. Another important thing here is that, before requesting the certificate, it is better to disable SNI proxy during that time and enable Apache's default ports i.e. 80 & 443. This way, there will be fewer chances of getting some errors or getting the request denied from the LE server.

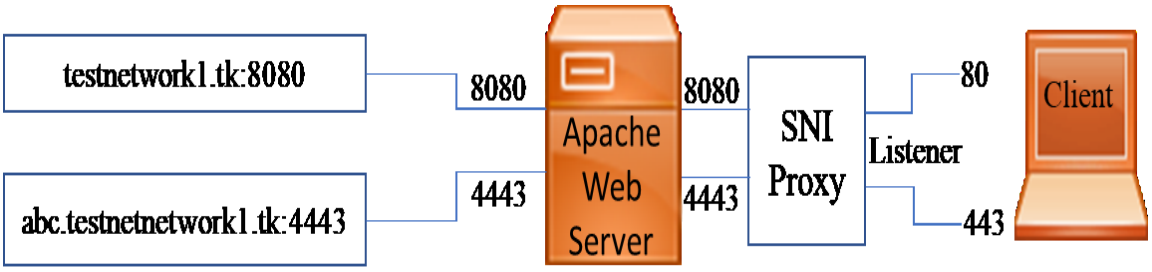

Figure 5.3 (b): SNI proxy is handling both HTTP & HTTPS traffic at default ports

Once the certificates are issued to the domains, SNI is enabled again and the domains are tested. At this stage, it has been observed that the desired certificate is issued to the client on its request from the server, otherwise, the client will get a certificate warning and the request will unable to proceed further.

5.2.3 Experiment 3: Approaching target Hidden Service using TorSNIP

Now, since the base of the network has been laid down, the next step is to forward the user's request to the targeted HS. The idea behind this approach is that the user could be able to reach the HS through the Clearnet domain with a regular Internet browser, rather than the Tor browser. In this thesis work, HS is configured on a separate virtual machine so that the user's request from the Clearnet domain could be relayed to the HS.

Onion address is the hash of the public key and is 16-characters long, which includes alphabets between 'a' to 'z' and numbers between '0' to '7' [44]. Since the onion addresses don't have 'Top Level Domain (TLD)', it is not yet possible to have domain validated certificates for these addresses from any well-known CA. The 16-character onion address is appended with the Clearnet domain name and a separate VH has been configured in Apache. This domain:onion pair made it possible to have a separate domain validated certificate from LE CA.

Now, once the certificate is issued to the domain:onion address pair, the next step is to relay the packets to the targeted HS. Delegate proxy is used to delegate the packets to the targeted HS. It also allows choosing a listening port on which the whole communication will take place.

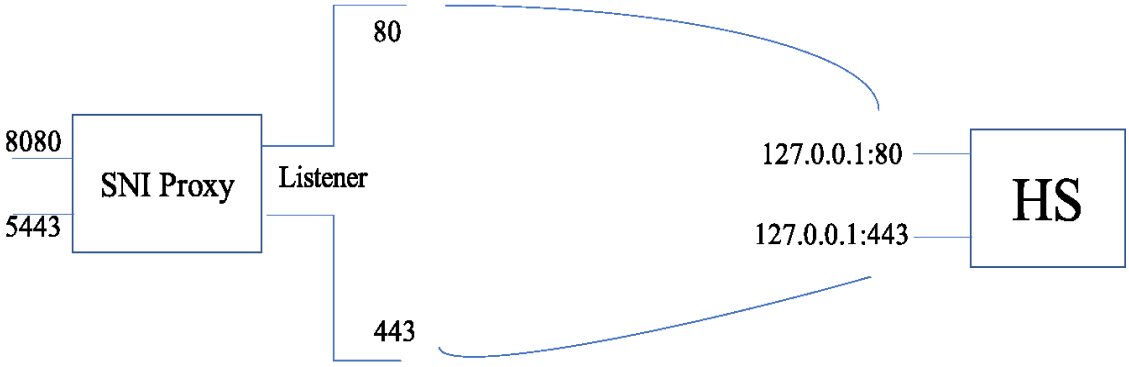

Figure 5.4: Packet is delegated to HS

The results are tested with the domain 'testnetwork1.tk' until SNI proxy. Later, the onion address of the HS is appended with the regular domain name 'hwboe2ozfcsn6n55.testnetwork1.tk' and obtained a certificate for it.

5.3 Scenario 1 (without MITM)

Pinning techniques as discussed earlier are very useful techniques in order to make the intermediate proxies reliable and trustworthy for the users. These scenarios elaborate the importance of using pinning techniques.

Day one; the user connects to the website (hwboe2ozfcsn6n55.testnetwork1.tk) and validates that the website has a valid certificate. It also validates the authenticity of the certificate signed by one of the well-known Intermediate or Root CAs. The browser accepts the certificate if signed by some well-known Intermediate or Root CA.

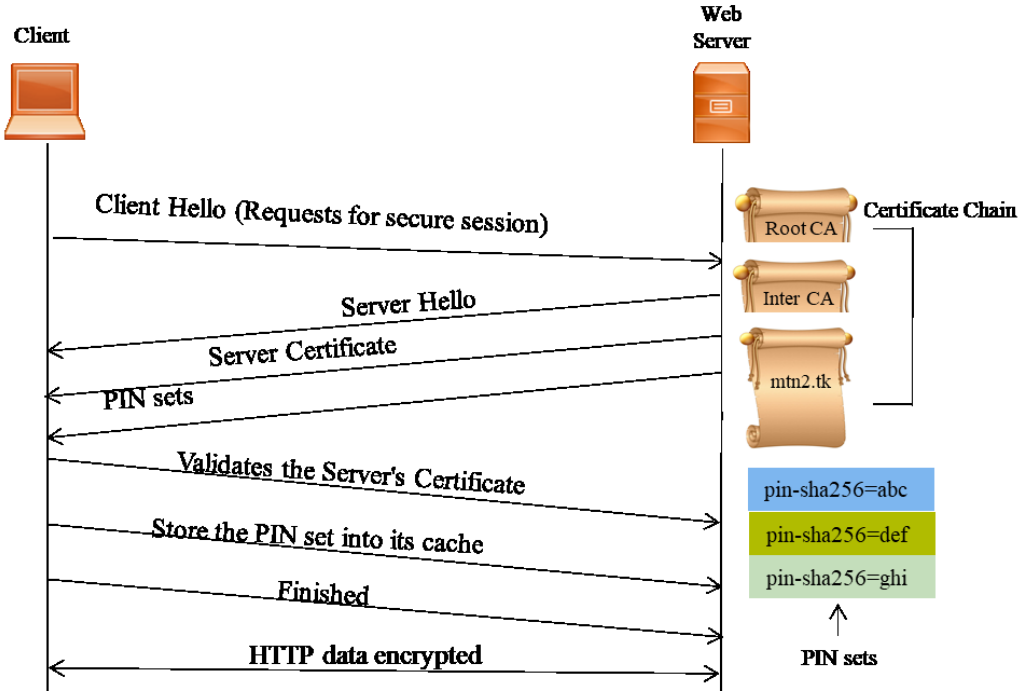

Figure 5.5 (a): Certificate & PIN is authenticated and accepted

Next, it checks that the website (hwboe2ozfcsn6n55.testnetwork1.tk) is publishing a Public-Key-Pinning set and it has a different PIN for each certificate. Then it makes sure that one of these PINs should match the one in the Certificate. If it found any one of the PIN set matchings, the user connects to the website securely and the browser will remember the PIN for the next connection.

For example, if the user connects to the website the next day, the browser will verify the website's certificate in the same manner as it did previously. This time since the browser already has the PIN set stored from the previous connection, so it will check the stored PIN with the one in the certificate. If it found the matching, the connection successfully establishes, and the user may surf the website securely.

5.4 Scenario 2 (with MITM)

Let’s say, at some other day a hacker is successful in taking control over the CA and starts issuing rogue certificates to the clients. The same user who had used the website (hwboe2ozfcsn6n55.testnework1.tk) previously a few days ago, again wants to connect to the website now. The client’s browser will again try to validate the certificate provided by the web server. The browser sees some unsecured CA (compromised CA) signs the certificate issued to testnetwork1.tk and that unsecured CA is in the list of its trust store, therefore the browser will find everything good and will happily accept the certificate and moves forward.

Next, the client’s browser will see that the website also has the Public-Key-Pinning policy applied and it also sees that the PIN set is already stored in its database, so it will try to validate those stored PINs with the one in the certificate. It tries to match the first PIN with the domain level certificate PIN. Since there is no match found, it checks the same PIN with intermediate CA PIN. It again does not match, so it checks the root CA PIN and still it does not find any match. This way, it tries to match all the PIN one by one it had stored or cached in the PIN set with the ones in the CA. Since the CA has been compromised, the new PIN will not match the one stored in the client’s database, and the connection will be refused and the client will get an alert that there is some problem in the middle of the connection. Therefore, it is better to try later once the secure connection is available and established between the client and the server.

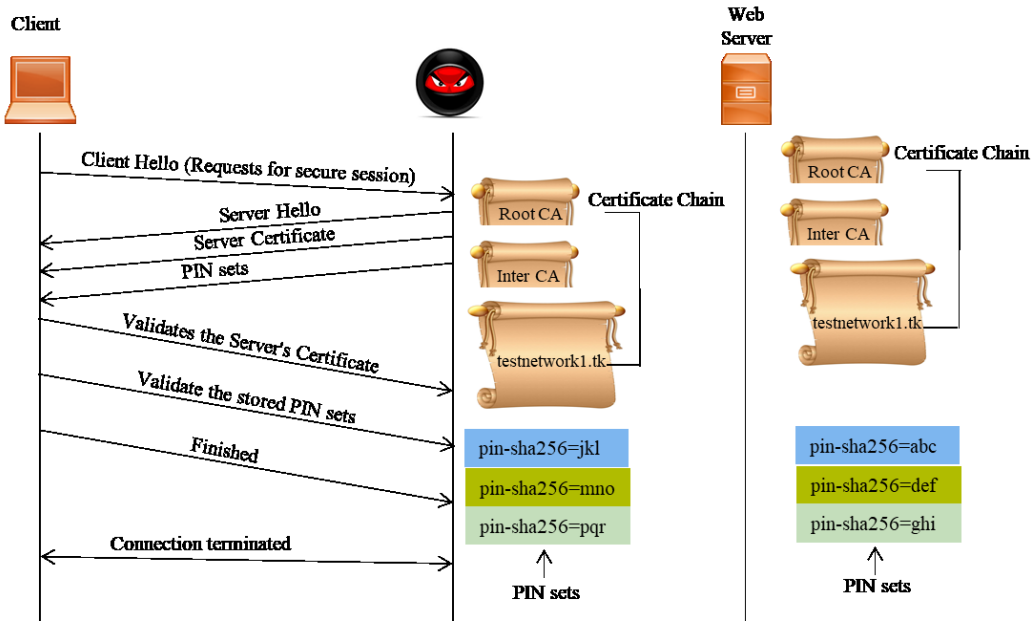

Figure 5.5 (b): Rouge certificate provided by MITM & is accepted by the client's browser but PIN mismatch occurs

This policy keeps the user safe from connecting to malicious websites that have a valid certificate and tries to trick the clients by showing that everything is valid and they are connecting to the safe website. According to the RFC document [41], the report_URI field can be used to send such reports to the server, so that the problem should be checked and resolved as soon as possible. The document also recommends that the client can try later to connect to the website.

5.5 Results

Below are some of the results, gathered after the completion of the implementation part. Since the work is in the initial phases, the domain name and the content of the web page are kept at a basic level, just to make it sure that the right web page is accessed on the user's request.

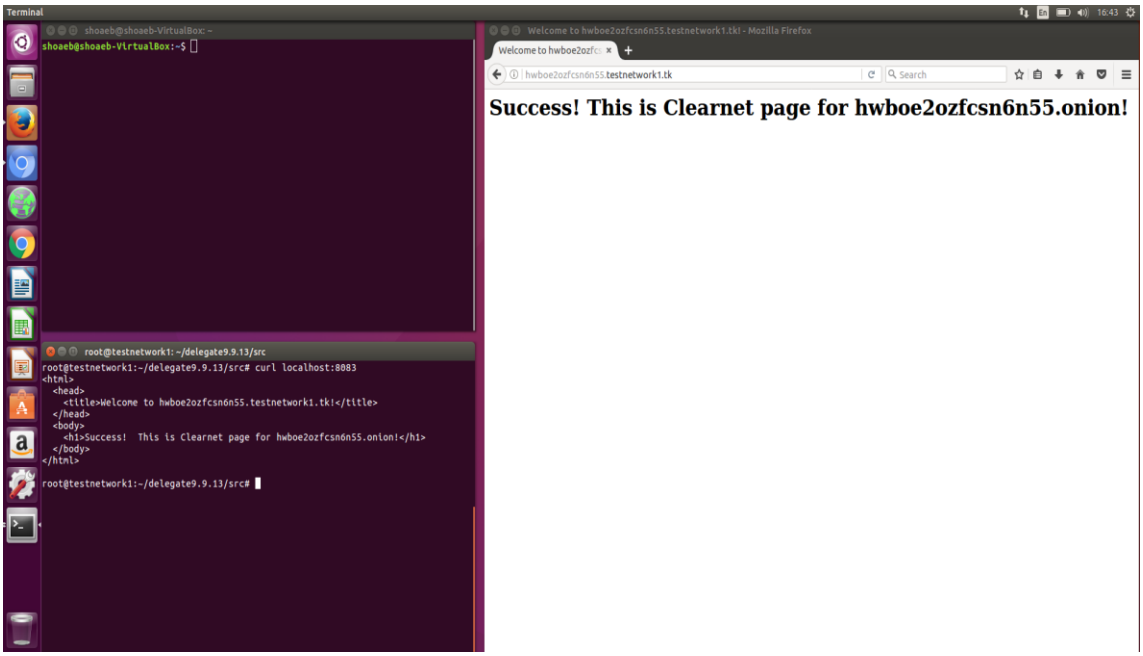

Figure 6.1 (a): Web page accessed using HTTP without enabling SNI proxy

At this point, the web page "hwboe2ozfcsn6n55.testnetwork1.tk" is available from the Apache server. The SNI proxy is still enabled at this point and the information of the website is defined in the SNI proxy script, along with the assigned port number. Therefore, since the SNI proxy is listening on port 80 and 443, it forwards the user request to the assigned port number in the Apache web server at the backend. Thus, the user gets the web page at default port 80 in this scenario rather than the defined port in Apache i.e. 8083.

On the left side of the picture, there are two terminal windows. The top one is the terminal window of the OS installed on the virtual machine. The machine has Tor and HS installed on it. The bottom one is the terminal window of the VPS, towards which the DNS is pointing. It has the SNI proxy script but has no Tor or HS installed on this system. It also contains the required VH. The bottom terminal also shows the port number on which the actual "hwboe2ozfcsn6n55.testnetwork1.tk" virtual host is.

The picture below is also similar but at this point the secure web page of "hwboe2ozfcsn6n55.testnetwork1.tk" is accessed using SNI proxy. Here again, the actual VH port is "6443" but the user is able to access it at default HTTPS port 443. The SNI proxy script has the information added to it, of the website along with the port number.

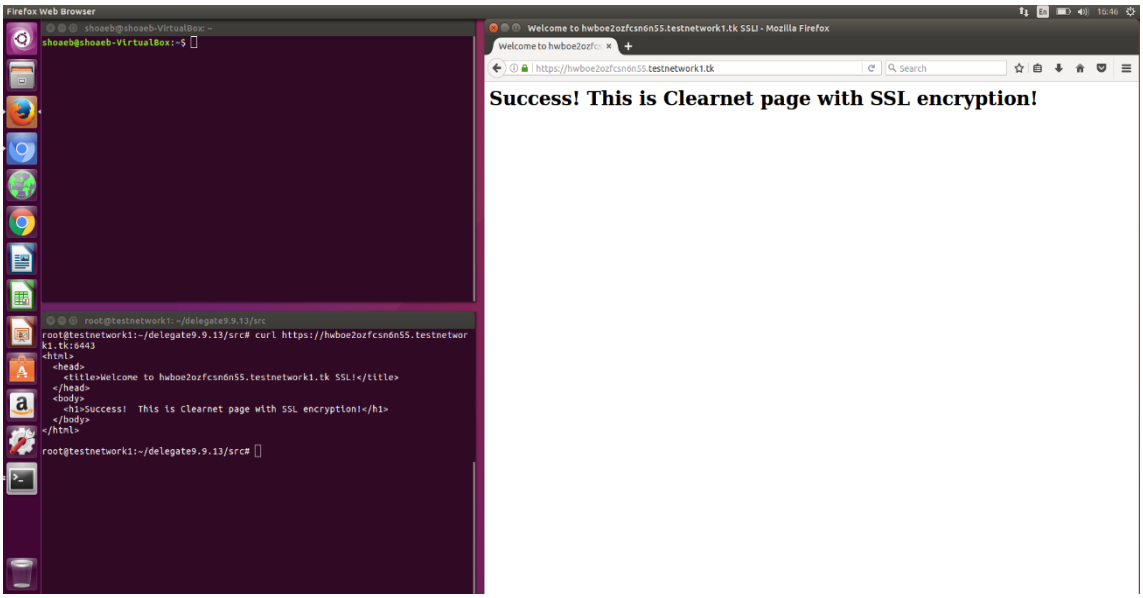

Figure 6.1 (b): Web page accessed using HTTPS without enabling SNI proxy

In this picture, the HS "hwboe2ozfcsn6n55.onion" page is shown, which is accessed using Tor browser. The Tor HS with ".onion" can only be accessed using Tor browser and not through any regular Internet browsers.

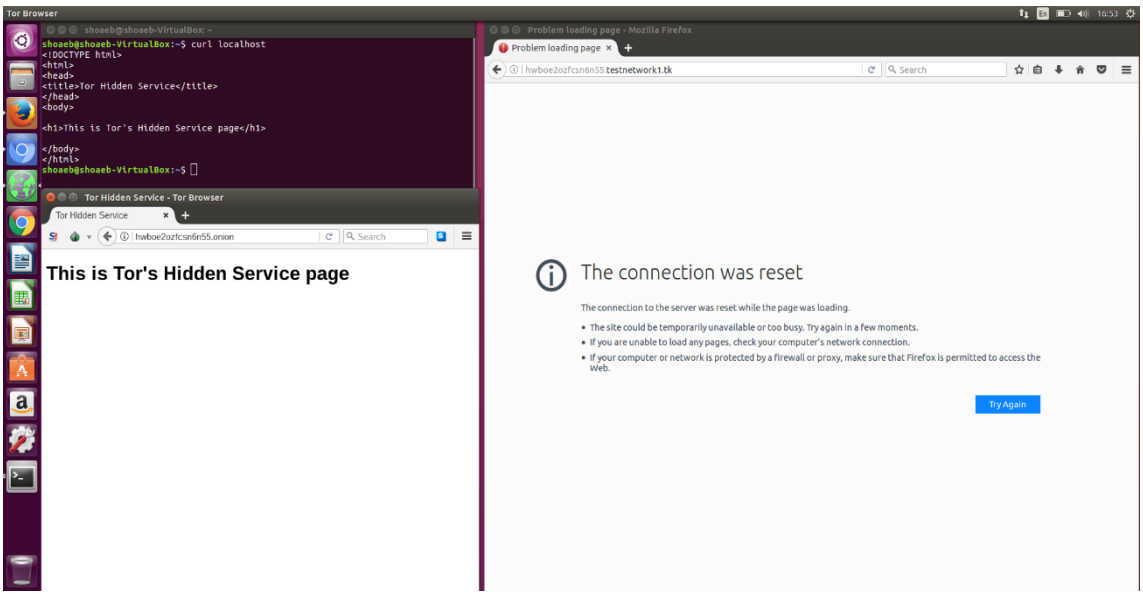

Figure 6.1 (c): Tor Hidden Service accessed using Tor Browser

To access the HS through a regular Internet browser, this thesis work uses TorSNIP which delegates the user's request to the targeted HS using Clearnet domain name. As shown in the picture below, the tunnel created using delegate proxy helps in reaching the HS on the regular Internet browser. The channel is still not secure this time because the HS is accessed on the default HTTP port 80.

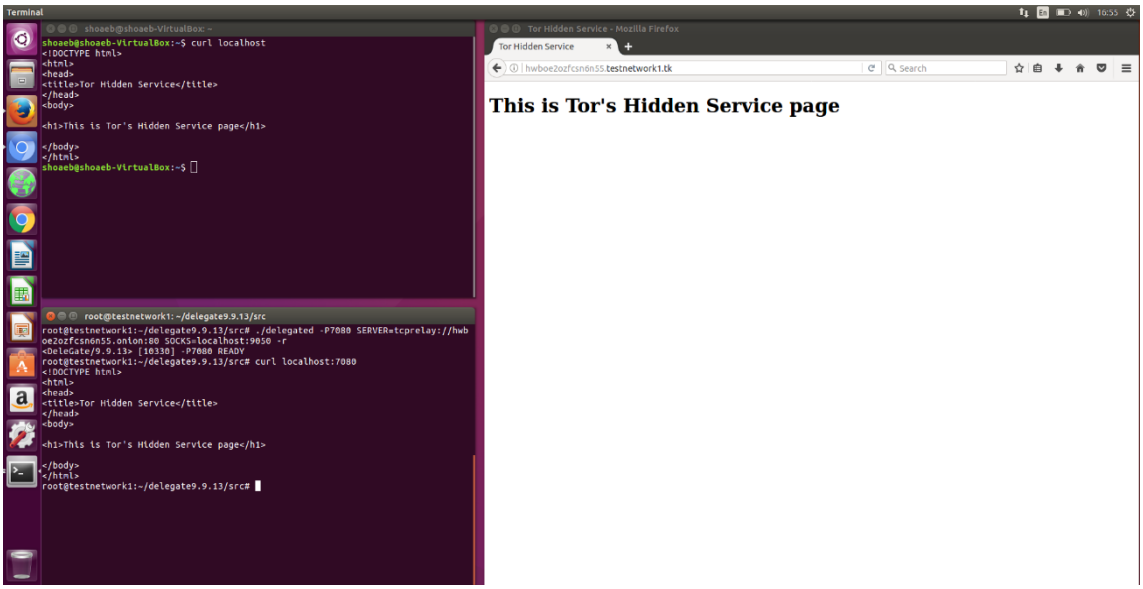

Figure 6.1 (d): Hidden Service is accessed using HTTP after enabling SNI proxy

In the picture below, the HS is accessed on port 443 using SNI proxy.

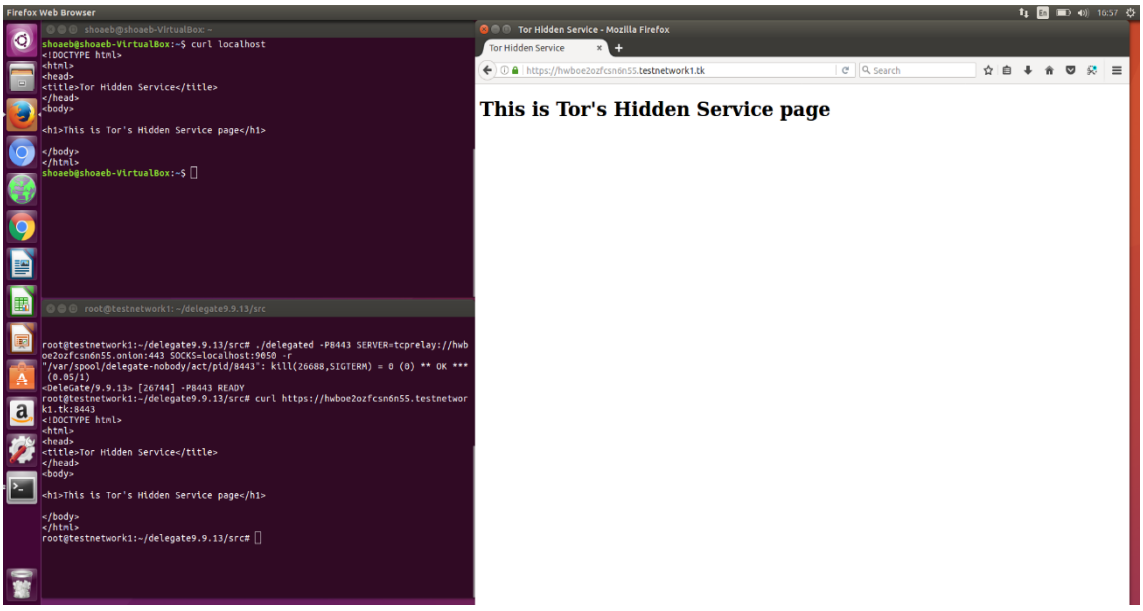

Figure 6.1 (e): Hidden Service is accessed using HTTPS after enabling SNI proxy

6. Discussion and Technical Analysis

Tor2web provides an alternative platform to access the onion sites. It acts as a proxy by taking the user's request and forwarding it to the targeted onion site, using the regular Internet. The user does not need to have Tor installed on the system, rather Tor2web server has it pre-installed. Technically, the Tor2web node takes the request 'foo.com' from the user and translates it into 'foo.onion' over Tor, i.e. the node decrypts the packet received from the user's browser and builds an onion circuit to establish a connection with the HS, requested by the user.

In its own, it provides a great opportunity for the larger audience to reach the HS. It is also good for some of the users who do not want to expose any Tor related activity on their system. However, Tor2web service is vulnerable to several attacks, including alteration of the data and content injecting. Since it is a proxy; everything that passes through it will be visible to the intermediate node including URLs, username/password, cookies etc.

As noticed from the above results, the target HS is reachable through the deployment of SNI proxy, which helped in forwarding the HTTP and HTTPS packets directly to the onion site on two separate virtual ports chosen randomly. The TCP packet is delegated to the HS using TCP relay protocol, which relays the communication between the client and the server without interpreting the transmission. This makes it possible to forward the TCP packet directly to the HS with the help of SNI proxy, which sends the server name before the TLS handshake begins.

One of the main advantages of this service is that it reduces the risk of coercive and other types of network attacks that occur if the communication is not encrypted at any point to/from the HS. Thus, it provides an extra layer of encryption to the user's data, from the user's browser until termination at the onion site. This allows anonymous publishers to provide access to larger audiences, without any fear of losing data and user's privacy. Because, when the data to be shared between the client and the server is wrapped in an extra layer of encryption i.e. HTTPS, the eavesdropper will not able to identify the communication that is done.

To enhance the security of the service and the user's trust, it is also possible to use some pinning techniques e.g. HPKP and HSTS as used in this thesis work. These techniques help in preventing man-in-the-middle attacks by mitigating fake certificates and forcing the browser to use HTTPS to communicate with the server instead of HTTP.

From the viewpoint of proxy operator this could be very helpful to increase the user's trust in their proxy so that more and more users rely on their proxy and feel safer in using it. The operator has the task to receive the request from the client, read the TLS SNI

header without breaching the original content and forwarding it to the targeted HS. Also, the operator receives the response from the HS and forwards it back to the client without any tampering or content injection. The operator also has to have a gateway open to all the hidden services except the ones that are considered illegal including child pornography, drug dealer etc. The operator can also take feedback from the clients just in case any illegal content is found or can run its own search to identify illegal sites and restrict access to such sites.

On the other hand, scalability issues may arise in such situation because the proxy operator has to update the HTTP and HTTPS table entries explicitly by themselves. Therefore, it has a separate local database which includes the list of domain:onion mapping along with the port number to operate as a public gateway. This increases the liability for the proxy operators.

From the HS operator's point of view, this service will be an additive advantage in terms of freedom of speech. The HS operators could be safer in publishing content and passing it to a larger audience. The proportion of the number of users visiting HS would increase and therefore the demand will grow. Moreover, the users will be safer in accessing the HS content without the use of any separate software. The HS operators will also be able to provide the content to the users without any risks of running Clearnet domain.

This can be very useful in countries where people are afraid of talking against higher officials. Especially, in the middle east during the Gulf crisis, many countries posted restrictions on the use of the Internet, including social media websites and communication tools. In such situations, many of the users can benefit from using HS and propagate the footages and videos to the people who are otherwise restricted from viewing such content by the government.

Beside several advantages of the service, there are some limitations as well. Although the link is secure using HTTPS, still the middleman could be able to retrieve some information e.g. user's geographical location and the website visited. This could be helpful in analyzing the activity of the user. Besides that, the HS operator would have to share their personal information with the CA before issuing the certificate for their website. This leaks information about the HS in the common name field. Therefore, it is always recommended to use Tor browser to access the HS in order to achieve the maximum level of end-to-end anonymity.

Besides security and privacy, another important part is the number of servers that are up and running at the same time. If there is only a single server, providing services to the user, it is quite possible that the service may be interrupted due to maintenance or unavailability of the server. The servers could be distributed evenly around the globe so that the majority of users could get access to available services without interruption.

Like holding a CA and issuing a rogue certificate, there is another attack called coercion attack which is a far worse attack than the one discussed earlier. Coercion attack is the one in which the attacker takes control of the main system and forces the target to give up the cryptographic credentials or keys. The system in this case scenario could be the proxy server that is handling user requests for reaching HS. The attacker forces the proxy operator to hand over the sensitive information or the keys to decrypt the information or otherwise punishes the operator. Use of ephemeral keys limits the attack of stealing the keys because these are short-lived keys, therefore, the main target of the attacker would be stealing the sensitive information.

There is another situation in which the attacker sits in between the CA and the proxy. Therefore, controlling both the CA and the proxy, which could be the worst threat model because the attacker will be issuing the rogue certificates and manipulating the data at the same time.

One of the possible solution to handle this situation is to implement Pinning techniques HSTS and HPKP as discussed above. The generated PIN will be stored in the client's browser and as soon as the PIN mismatches, the client gets alert and stops communicating using the proxy. The client who is communicating for the very first time will not have the PIN stored in the browser, therefore, there is more chance of the client getting trapped. One thing can be done is that the client who has the PIN can propagate to other clients that the proxy is under attack and therefore avoid communicating through it. This will help the clients who are connecting with the proxy for the first time and does not have any PIN stored in their browser from previous session.

Another possible method against this kind of attacks is to use bots that can scan the systems integrity and update the administrator immediately. It can make harder for the outsider to gain access to the system. Although it cannot prevent the system from attacks such as DDoS, but it can be an asset if used correctly.

Besides coercion attack there are many other network attacks that can help eavesdroppers and hackers gain access to the system. For example, spoofing, session hijacking, cookie, and traffic flow analysis. Use of Tor protects the user from a common form of internet surveillance attack called "traffic analysis" by protecting both the content of the message and the header information that identifies sender and receiver.

TorSNIP proxy implements various security features including HSTS and HPKP which enhances the security of the system and increases the trust level of the users. In comparison to Tor2web proxy, TorSNIP is more efficient in terms of exchanging information with the HS and is also secure enough to build user trust on the proxy. TorSNIP also keeps the user data end-to-end secure from the browser until termination to HS. This lowers the risk of message tampering or manipulation during the message exchange.

Below is a comparison table with list of threats the proxy services can face and comments whether the service is secure or not.

Table 1: Security Considerations		
List of Threats	Tor2web	TorSNIP
Content impersonation	No	Yes
Traffic Analysis	No	Yes
Coercive threats	No	Yes
DDoS	No	No

7. Conclusion

Tor network provides a platform for the users to route packets through nodes that only get limited information about the packet. Each node in the Tor network only has information about its predecessor and successor nodes thus, hiding the source and destination of the packet. The end node may leak some information only if the destination is on the regular Internet, but the use of secure channel HTTPS would limit that risk too. If the destination is the HS which resides inside the Tor network, then end-to-end anonymity could also be achieved with the use of Tor browser.

A limited number of users access the HS because of its dependency on Tor browser which is one of the many reasons. To access the hidden content through a regular Internet browser, Tor2web provides services at the risk of compromising the privacy of both the user and the HS. The information exchanged through Tor2web proxy is at risk to a man-in-the-middle attack. The risk could also include injecting the additional content as well as modification of the original content.

TorSNIP addresses these issues and eliminates them by the use of SNI proxy. TorSNIP does not decrypt the received packets at the intermediate node, rather forwards it directly to the target HS. This limits the risk of different attacks mentioned above and makes the communication end-to-end secure. Since, in the earlier scenario, the HS does not have an IP address, therefore scalability issues could arise. With the use of SNI proxy, multiple HS can link with a single IP address, which enhances the scalability of the HS.

The current prototype is a testing prototype and can be enhanced, developed, and deployed for the larger networks. Docker could be one of the tools that can be used to merge different processes, including different virtual hosts, certificates, and packet delegation. This could make the work deployment faster and easier. This could also make the process more dynamic rather than static i.e. the information in the SNI script could be updated dynamically, which makes it useful for the larger network.

Currently, only the text data has been observed, that has been retrieved from the HS. In the future, besides expanding the network, research can also be done on handling metadata including multi-media files (audio, video). It could also be analyzed, whether a single proxy could handle all the metadata or a separate proxy would be required to handle multi-media data.

BIBLIOGRAPHY

[1] "HTTPS Everywhere | Electronic Frontier Foundation." [Online]. Available: https://www.eff.org/https-everywhere/faq#what-if-https-everywhere-breaks-some-site-that-i-use. [Accessed: 06-Sep-2017].

[2] S. S. McPherson, *Tim Berners-Lee : inventor of the World Wide Web*. Twenty-First Century Books, 2010. Available: https://books.google.fi/books?id=wK0xBWfL9GkC&printsec=frontcover#v=onepage&q&f=false

[3] K. Chayka, "Everything You Need to Know About the Mt," *Time.com*, p. 1, 2014. Available: https://www.whoishostingthis.com/blog/2017/03/07/tor-deep-web/

[4] Roger Jolly, "CLEARNET VS HIDDEN SERVICES - WHY YOU SHOULD BE CAREFUL - Deep Dot Web," 2017. [Online]. Available: https://www.deepdotweb.com/jolly-rogers-security-guide-for-beginners/clearnet-vs-hidden-services-why-you-should-be-careful/. [Accessed: 11-Oct-2017].

[5] J. Palme and M. Berglund, "Anonymity on the Internet," *Website*, pp. 4–12, 2002. Available: https://people.dsv.su.se/~jpalme/society/anonymity.pdf

[6] Aaron Swartz, "In Defense of Anonymity (Aaron Swartz's Raw Thought)," 2008. Available: http://www.aaronsw.com/weblog/tor2web

[7] Christian de Looper, "10 biggest leaks of the past 10 years | TechRadar," 2017. [Online]. Available: http://www.techradar.com/news/10-biggest-leaks-in-the-past-10-years. [Accessed: 11-Oct-2017].

[8] M. John Marcos and A. Joe Melgarejo, "Anonymity Is King," p. 11. Available: https://www.virusbulletin.com/uploads/pdf/conference/vb2015/Marcos_Melgarejo-VB2015.pdf

[9] J. Kowalski and K. Gabert, "TorStatus - Tor Network Status," 2017. [Online]. Available: https://torstatus.blutmagie.de/. [Accessed: 06-Sep-2017].

[10] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," 2004.

[11] S. F. Sabatini F, "Tor: Overview," *Tor*, p. 1, 2014. Available: https://www.torproject.org/about/overview.html.en

[12] Timekeeper, "Monetarists Anonymous.," *Economist*, vol. 404, no. 8804, p. 1, 2012. Available: http://www.economist.com/node/21563752

[13] J. H.-C. Eerke Boiten, "Can you really be identified on Tor or is that just what the cops want you to believe?," 2014. [Online]. Available: https://phys.org/news/2014-07-tor-cops.html. [Accessed: 06-Sep-2017].

[14] P. Ferguson and G. Huston, "What is a VPN ?," vol. 6, no. 1, pp. 15–22, 1998.

Available: https://www.potaroo.net/papers/1998-3-vpn/vpn.pdf

[15] A. S. Dean Drew, “Using Client Puzzles to Protect TLS Drew Dean Adam Stubblefield Xerox PARC,” p. 8. Available: http://studylib.net/doc/11607460/using-client-puzzles-to-protect-tls-drew-dean-adam-stubbl...

[16] G. Pellerano, “Who am I?,” 2013. Available: http://urna.winstonsmith.org/materiali/2013/atti/16_ep2013_pellerano_tor2web.pdf

[17] F. Pietrosanti, “TOR2WEB: Past, present, future of anonymous publishing infrastructure,” *e-privacy 2012*, 2012. Available: http://urna.winstonsmith.org/materiali/2012/atti/Pietrosanti_Tor2web.pdf

[18] Giovanni Pellerano, “Tor2web,” 2014. Available: https://github.com/globaleaks/tor2web/wiki

[19] G. Introduction, “A Gentle Introduction to How I2P Works,” pp. 2–5, 2011. Available: https://geti2p.net/en/docs/how/intro

[20] Netscape, “The SSL Protocol.” . Available: https://web.archive.org/web/19970614020952/http://home.netscape.com/newsref/std/SSL.html

[21] D. Robert, “SSL: Intercepted today, decrypted tomorrow,” 2013. Available: https://news.netcraft.com/archives/2013/06/25/ssl-intercepted-today-decrypted-tomorrow.html

[22] E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.3,” in *Internet Engineering Task Force (IETF)*, 2015. Available: http://www.ietf.org/rfc/rfc5246.txt

[23] Kemmerer Chris, “The SSL/TLS Handshake: an Overview - SSL.com,” 2015. [Online]. Available: https://www.ssl.com/article/ssl-tls-handshake-overview/. [Accessed: 11-Oct-2017].

[24] J. Hodges, C. Jackson, and A. Barth, “HTTP Strict Transport Security,” *Req. Comments*, pp. 1–46, 2012.

[25] J. Hodges, A. Barth, and C. Jackson, “Section 14. Security Considerations,” *RFC 6797*, 2012. [Online]. Available: https://tools.ietf.org/html/rfc6797#section-14. [Accessed: 24-Jan-2017].

[26] the chromium projects, “HTTP Strict Transport Security - The Chromium Projects,” 2010. [Online]. Available: https://dev.chromium.org/sts. [Accessed: 24-Jan-2017].

[27] A. Deveria, “Strict Transport Security,” *The Chromium Projects*, 2015. [Online]. Available:

https://www.chromium.org/sts%5Cnhttp://caniuse.com/#feat=stricttransportsecurity. [Accessed: 24-Jan-2017].

[28] B. Johansen, "HTTP Public Key Pinning (HPKP)," *July 16*, 2015. [Online]. Available: https://www.bjornjohansen.no/public-key-pinning. [Accessed: 08-Feb-2017].

[29] S. Helme, "HPKP: HTTP Public Key Pinning," *January 20*, 2015. [Online]. Available: https://scotthelme.co.uk/hpkp-http-public-key-pinning/. [Accessed: 10-Feb-2017].

[30] Remy van Elst, "HTTP Public Key Pinning Extension HPKP for Apache, NGINX and Lighttpd - Raymii.org," *December 30*, 2014. [Online]. Available: https://raymii.org/s/articles/HTTP_Public_Key_Pinning_Extension_HPKP.html. [Accessed: 25-Feb-2017].

[31] Margaret Rouse, "What is certificate authority (CA)? - Definition from WhatIs.com," *4 June*, 2007. [Online]. Available: http://searchsecurity.techtarget.com/definition/certificate-authority. [Accessed: 24-Jan-2017].

[32] "List of certificate authorities in browsers and mobile platforms - Information Security Stack Exchange." [Online]. Available: http://security.stackexchange.com/questions/49006/list-of-certificate-authorities-in-browsers-and-mobile-platforms. [Accessed: 24-Jan-2017].

[33] J. Aas, *Let's Encrypt is Trusted*. 2015. Available: https://letsencrypt.org/2015/10/19/lets-encrypt-is-trusted.html

[34] J. Sanders, "Let's Encrypt initiative to provide free encryption certificates," *TechRepublic*, 2014. Available: https://www.techrepublic.com/article/lets-encrypt-initiative-to-provide-free-encryption-certificates/

[35] Z. Zorz, "Let's Encrypt CA releases transparency report before its first certificate," *Help Net Secur.*, 2015. Available: https://www.wikiwand.com/ja/Let%27s_Encrypt

[36] SSL2BUY, "Learn Different SSL Certificate Validation Process by Its Types." [Online]. Available: https://www.ssl2buy.com/wiki/learn-different-ssl-certificate-validation-process-by-its-types. [Accessed: 19-Sep-2017].

[37] LINUX FOUNDATION COLLABORATIVE PROJECTS, "How It Works - Let's Encrypt - Free SSL/TLS Certificates." [Online]. Available: https://letsencrypt.org/how-it-works/. [Accessed: 09-Mar-2017].

[38] Comodo, "Overview of EV SSL Certificates Validation Process." [Online]. Available: https://www.instantssl.com/ssl-certificate-products/ssl/ssl-ev-validation.html. [Accessed: 19-Sep-2017].

[39] H. F. Gaines, *Cryptanalysis; a study of ciphers and their solution.* New York:

Dover Publications, 1956. Available: http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2010-2011/cryptanalysis.pdf

[40] "What is Certificate Transparency?," 2013. Available: https://www.certificate-transparency.org/what-is-ct

[41] G. Sadasivan, N. Brownlee, B. Claise, and J. Quittek, "Architecture for IP Flow Information Export," RFC Editor, 2009. Available: http://www.ietf.org/internet-drafts/draft-ietf-ipfix-architecture-12.txt

[42] G. Sadasivan, "Architecture for IP Flow Information Export," RFC Editor, 2006. Available: http://www.ietf.org/internet-drafts/draft-ietf-ipfix-architecture-12.txt

[43] Dlundquist, "SNI proxy," *SNIProxy*, 2014. [Online]. Available: https://github.com/dlundquist/sniproxy. [Accessed: 10-Aug-2017].

[44] Lachesis, "GPU-based Onion Hash generator," *GitHub*, 2016. Available: https://github.com/lachesis/scallion

APPENDIX. SNI Script

```
user daemon
pidfile /var/run/sniproxy.pid

# loaded from /etc/resolv.conf
  resolver { mode ipv4_only }
  error_log {
  syslog daemon
  priority notice
  }

# HTTP
listen 80 {

       proto http
       table http_hosts
       reuseport yes

       }
# HTTPS
listen 443 {

       proto tls
       table https_hosts

}
table http_hosts {

#hwboe2ozfcsn6n55.testnetwork1.tk 127.0.0.1:8083     # Apache
hwboe2ozfcsn6n55.testnetwork1.tk 127.0.0.1:7080      # Delegate

}

table https_hosts {

#hwboe2ozfcsn6n55.testnetwork1.tk 127.0.0.1:6443 # Apache (HTTPS)
hwboe2ozfcsn6n55.testnetwork1.tk 127.0.0.1:8443  # Delegate (HTTPS)

}

```