August 2015
Master's Degree Thesis

A Comprehensive Analysis on End-to-End Node Selection Probability on the Tor Network

Graduate School of Chosun University

Department of Computer Engineering

Saurav Dahal

A Comprehensive Analysis on End-to-End Node Selection Probability on the Tor Network

Tor 네트워크에서 종단간 노드 선택 확률에 대한 분석

August 25, 2015

Graduate School of Chosun University

Department of Computer Engineering

Saurav Dahal

A Comprehensive Analysis on End-to-End Node Selection Probability on the Tor Network

Advisor: Prof. Seokjoo Shin, PhD

A thesis submitted in partial fulfillment of the requirements for a Master's degree

April 2015

Graduate School of Chosun University

Department of Computer Engineering

Saurav Dahal

다할 사우라브의 석사학위논문을 인준함

위원장 조선대학교 교수 모 상 만 (인)

위 원 조선대학교 교수 신 성 주 (인)

위 원 조선대학교 교수 강 문 수 (인)

2015 년 5 월

조선대학교 대학원

TABLE OF CONTENT

TABLE OF CONTENT i

LIST OF FIGURES iii

LIST OF TABLES v

ABSTRACT vi

한 글 요 약 vii

I. INTRODUCTION 1

A. Introduction to Privacy Enhancing Technologies 1

B. The Nymity Slider 2

C. Introduction to TOR Network 3

D. Problem Statement 6

E. Research Contribution 6

F. Thesis Layout 7

II. RELATED WORKS 8

A. Anonymity Networks 8

B. Node Selection Analysis 10

III. TOR NETWORK 11

A. Components of TOR 11

B. TOR Streaming Cells 14

C. Status Flags (Types of Nodes) 16

D. Tor Configuration File 17

E. Circuit Creation and Data Transmission 19

F. Processing Cells at Onion Routers 21

G. Bandwidth Weighted Node Selection 22

H. Equation Based Results 24

IV. SHADOW SIMULATOR 27

V. PERFORMANCE EVALUATION 29

A. Simulation Environment 29

B. Performance Results and Discussion 30

VI. CONCLUSIONS 41

BIBLIOGRAPHY 42

LIST OF FIGURES

Fig. 1. Components of TOR Network. 11

Fig. 2. TOR Design. 13

Fig. 3. Tor Cell Format (Control cell) 16

Fig. 4. Tor Cell Format (Relay cell) 16

Fig. 5. Overview of client Torrc Configuration script file 18

Fig. 6. Circuit Creation and Data Transmission. 20

Fig. 7. Processing the cells at onion routers. 21

Fig. 8. Catch Probability for n=10. 25

Fig. 9. Catch Probability for n=20. 25

Fig. 10. Catch Probability for n=40. 26

Fig. 11. Shadow's architectural design. 27

Fig. 12. Shadow working procedure. 28

Fig. 13. End-to-End Selection Probability when guard- exit pair and EE node are added to the Tor network for n=1. 300

Fig. 14. End-to-End Selection Probability for n=10. 31

Fig. 15. End-to-End Selection Probability for n=20. 32

Fig. 16. End-to-End Selection Probability for n=40. 32

Fig. 17. End-to-End Selection Probability when EE nodes, in different locations, are added to the Tor network. 33

Fig. 18. Guard selection Probability when guard-exit pair and EE node are added to the Tor network for n=1.....34

Fig. 19. Guard selection Probability for n=10. 35

Fig. 20. Guard selection Probability when guard-exit pair and EE node are added to the Tor network for n=20.....35

Fig. 21. Guard selection Probability when EE nodes, in different locations, are added to the Tor network. 36

LIST OF TABLES

Table 1. Parameters of Live TOR Network.....24

Table 2. Simulation Parameters.....29

Table 3. Prediction for number of nodes.....38

Table 4. Prediction for bandwidth required.....40

ABSTRACT

A Comprehensive Analysis on End-to-End Node Selection Probability on the Tor Network

Saurav Dahal
Advisor: Prof. Seokjoo Shin, Ph.D.
Department of Computer Engineering
Graduate School of Chosun University

Tor is an open network that helps to defend against traffic analysis and thus achieves anonymity and resisting censorship online. So, it is a useful tool for those users who want to remain anonymous participating online. On the other hand, there are such users also who takes advantage of anonymous system and perform illegal activities. In order to frustrate such users, nowadays many researches have been carried out to attack Tor and to break the anonymity. To deanonymize the Tor, the attacker must be able to control both the guard node and exit node of a circuit.

In this thesis, after classifying different types of node in TOR network the analysis on end-to-end node selection probability when an attacker adds different types of compromised nodes in the existing Tor network has been widely studied. Shadow simulator which is believed to show as similar environments as the real Tor network has been used for the experiment besides mathematical analysis. By extensive performance evaluation, it is concluded that when guard + exit flagged compromised nodes are added to Tor network, the selection probability of compromised nodes gets higher.

한 글 요 약

Tor 네트워크에서 종단간 노드 선택 확률에 대한 분석

다할 사우라브
지도 교수: 신석주
컴퓨터공학과
대학원, 조선대학교

Tor는 트래픽 분석에 의한 공격과 온라인 검열로부터 익명성을 확보할 수 있는 오픈 네트워크이다. 따라서, 인터넷에 접속한 사용자가 익명성을 유지할 수 있도록 도와주는 효과적인 응용계층 프로토콜로 정의될 수 있다. 이러한 익명성을 이용하여 인터넷 상에서 불법적인 행위 (해킹, 마약거래, 돈세탁 등)를 하고자 하는 사용자들이 존재하며, Tor가 악의적 목적으로 활용되는 것을 억제하기 위하여 익명성을 파괴하는 방향으로의 Tor에 대한 공격 방법들이 연구되고 있다. Tor의 익명성을 파괴하기 위하여 공격자는 반드시 생성된 서킷의 가드노드와 엑시트노드 동시에 제어할 수 있어야 한다.

본 논문에서는 Tor 네트워크에서 사용되는 여러 형태의 노드들에 대한 분석으로부터 공격자가 서로 다른 형태의 조작된(compromised) 노드들을 Tor 네트워크에 제공하였다고 가정하였을 때의 종단 노드 선택 확률에 대한 분석 연구를 수행하였다. 수리적 분석에 더하여 실제 Tor 네트워크 환경에 가장 유사하게 구현된 섀도우 시뮬레이터를 활용하여 성능 분석을 수행하였다. 연구 결과로부터 조작된 가드+엑시트 노드가 Tor 네트워크에 추가된 경우에 종단 노드 선택 확률이 가장 높았음을 확인하였다.

I. INTRODUCTION

A. Introduction to Privacy Enhancing Technologies

A privacy enhancing technology (PET) is a technology whose goal is to enhance the privacy of the user. That is, a technology that prevents the position on the nymity slider[1] [33] from careening needlessly towards verinymity for its user's transactions. These technologies are designed for use in the online world, and are employed in a variety of contexts including instant messaging, web-browsing, and electronic publishing. Goldberg et al. [28, 29, 30] present a quintennial series that examines the progression of PETs. They contend that the most widely successful PET in history is Transport Layer Security (TLS), which is used to encrypt and secure Internet communication. It has become an invaluable tool on the Internet, and is used for nearly all authentication, e-commerce, and server remote control panels. Another successful PET is Off-the-Record Messaging [31], a plugin for popular instant messaging programs that provides authentication, security, confidentiality, and deniability to conversations. Contributing to its success is both the widespread use of instant messaging for private conversations and the fact that it is provided as a standard component for Adium [32] and is used automatically when both parties support it; some users do not even realize that their conversations are being protected from eavesdroppers. This is the ideal way to give privacy to users. To be widely used, it must be transparent to use and trivial to configure. Anonymity in the real world depends on the number of users of a system. Post office boxes allow the owner to receive mail anonymously since there are a lot of mail boxes in the system. If the owner was the only recipient of mail in

[1] It is a virtual one-dimensional slider that ranks the nymity of transactions along the axis of privacy, which clarifies the differences between levels of identity disclosure. The highest position, verinymity, means true name.

the system then it would afford him very little privacy. The pragmatic definition of privacy, in the electronic age, is based on this reality of anonymity. One is said to be anonymous if it is indistinguishable from a set, which is called the anonymity set. The definition considers the existence of an adversary who has the goal of identifying an anonymous user. A successful PET would result in the adversary being unable to determine which user (from the anonymity set) is the target for whom they are searching. More strongly, it would prevent the adversary from pairing any user of the system with their identity. The goal in PETs is to have the anonymity set equal to the set of users. This definition creates a corollary goal for any PETs that are developed. It is imperative that such technologies are popular and well-used as this will increase the anonymity set and thus strengthens the degree of anonymity afforded.

B. The Nymity Slider

Different transactions disclose identity to different degrees. In Goldberg's Ph.D. thesis [33], he presented the nymity slider, which clarified the differences between levels of identity disclosure. It is a virtual one-dimensional slider that ranks the nymity of transactions along the axis of privacy. The highest position, "verinymity", means true name. A verinymous transaction involves the disclosure of one's true identity, or information linked uniquely to their true identity. Examples in real life include using one's credit card, or showing government identification to prove one's age. Below verinymity is "pseudonymity", meaning false name. A pseudonym, or an alias, is a persistent front name behind which the true identity is hidden. In the real world, pseudonyms include author's pen names, or nicknames used by bloggers.

The important distinction between verinymity and pseudonymity is that one has control over the disclosure of their own identity—only the holder of the

pseudonym can choose to reveal it. Verinymity can be obfuscated, such as a Social Insurance Number (SIN) that uniquely corresponds to an identity. However, the lack of personal ownership over the mapping from one's SIN to their identity makes it decidedly verinymous.

The remaining positions on the slider are two varieties of anonymity, meaning without names: "linkable anonymity:" and "unlinkable anonymity". Linkably anonymous transactions are transactions that can be linked together, however the linking does not correspond to a real identity or a pseudonym: they are simply correlated. An example of this is seen in retail outlets that use loyalty cards to offer a small discount. (Often these cards are trivial to acquire and offer immediate benefit.) Such cards require no registration and are not linked to any identity. They are valuable to retail outlets for collecting data about customer buying patterns so as to juxtapose items to encourage "impulse buying". Finally, the lowest level on the nymity slider is unlinkable anonymity. This is true anonymity, where individual transactions are unlinkable to an external party or to each other. Using cash for a purchase, or providing the police with an anonymous tip over the telephone, are examples of unlinkably anonymous transactions. Importantly, the position on the nymity slider can move towards verinymity during a transaction much more readily than it can move towards anonymity. When one chooses to disclose an aspect of their identity during a transaction, such as using ID to prove their age of majority, then that transaction becomes verinymous even if the purchase is then done anonymously with cash. Similarly, an unlinkably anonymous transaction becomes linkable when the decision is made to use a loyalty card for a discount.

C. Introduction to TOR Network

The rapid growth of the Internet applications has made communication privacy an increasingly important security requirement. Although encryption aims at

preserving the contents and some modification of data, it is still possible for the adversaries to get significant information about the path carried on the network packets and the physical entities, such as the network addresses of the sender and the receiver of the message. One of the main problems is the exposure of the network address which might result in severe consequences: Adversaries can easily overhear all the messages and perform track analysis; even if the communication content is encrypted, routing information is still sent in the clear because routers need packets destinations in order to route them in the right direction. Thus an anonymous network should bear the properties such as receiver untraceability, sender untraceability and unlinkability.

Several platforms were developed to help increasing anonymity in Internet such as Threshold Mixes [1], Timed mixes [2], Threshold and/or Timed Mixes [3], Pool Mixes [4], Stop-and-Go Mixes [5], Binomial Mixes [6], RGB Mixes [7]. But these networks have certain limitations such as high latency, since too many public-key encryption and decryption were used and they were message based anonymity networks such as e-mail.

To overcome the limitations of other anonymity networks, TOR (The Onion Router) was introduced. Tor [8] is a distributed onion routing network deployed for fighting censorship and online privacy, supporting several hundreds of thousands of users daily [9, 10] and transferring roughly 8 GiB/s in aggregate [11]. TOR is designed to meet the following goals:

- Deployability: must be inexpensive to implement (e.g. requiring kernel patches), can't require non-anonymous parties like websites to run the software.

- Usability: a security requirement to hide users among users which should not require modification of applications, no prohibitive delays, and few configuration decisions, and easy implementation on all common OS platforms.

- Flexibility: the protocol should be flexible and well specified for future research work.

- Simple design: the protocols design and security must be user friendly.

Different kinds of people use the Tor network for different reasons. Military and Law enforcement use it for sting operations so their IP addresses cannot be traced to known police or military addresses. Military field agents, while in the field, use it to prevent being tracked down for visiting military related websites. Some journalists use Tor to remain anonymous from governments who would shut them down. Non-governmental organizations (NGOs) use Tor to allow their workers to connect to their home website while they're in a foreign country, without notifying everybody nearby that they're working with that organization. Tor is also commonly used by individuals to maintain privacy and evade harsh firewalls, especially firewalls with strict government enforced censorship.

Tor is composed of thousands of relays, volunteered by different volunteers all over the world. For anonymity, Tor passes the data stream through a tunnel circuit of three nodes. Data is encrypted using Onion Routing [12, 13], so that the relay cannot know the source and destination of any message. Any node in the circuit knows only about its one hop neighbors in the circuit. However, Tor is known to be insecure against end-to-end attacks (traffic confirmation attacks) i.e., an adversary that can observe a user’s traffic entering and exiting the anonymity network. This is

only possible if the adversary controls both the entry node and exit node in the circuit.

D. Problem Statement

Tor is a double edged sword. On its good aspect, it preserves privacy of the users. It has also helped people whose communication is censored by the government and the organizations. But along with the good side, it also possesses bad aspect. People use Tor to perform illegal activities over the internet such as drugs and ammunitions deals, hosting child pornography sites, black markets, threatening etc. These activities cause a serious problem in the security of the country. Hence deanonymization of Tor network becomes inevitable. Deanonymizing Tor network has two areas, one is the deanonymizing the Tor network itself and another is the deanonymizing of Tor hidden services[2] [41]. To deanonymize Tor network, an adversary must be able to control both guard as well as exit node of a Tor circuit. Thus, it is desirable if a way could be found out such that an adversary can control both the guard as well as the exit node of any circuit in order to deanonymize the user.

E. Research Contribution

The main objective in this thesis is to know how to achieve an end-to-end selection of compromised node in a Tor circuit. In order to meet this goal, each of the available nodes of Tor network will be tested and a conclusion about the findings of the simulation results will be made. Therefore, the following things will be looked in this thesis [40]:

- End-to-End Node selection Probability.

[2] Tor hidden services are the websites inside the Tor network. Their IP is hidden from the client.

- Guard Selection Probability.
- Selection Probability in different locations.
- Number of nodes required for different catch probability.

F. Thesis Layout

The rest of the thesis is organized as follows. First, in Chapter II, related works on anonymity networks and node selection analysis are presented. More detailed information on Tor network such as its components, flags, circuit creation etc. are presented in Chapter III. In Chapter IV, detail information on Shadow simulator is presented. In Chapter V, Performance evaluation of the simulation is done. Chapter VI concludes the thesis.

II. RELATED WORKS

A. Anonymity Networks

Anonymous communication can often be classified into two general categories: high-latency systems and low-latency systems. High-latency anonymity systems are able to provide strong anonymity, but are typically only applicable for non-interactive applications that can tolerate delays of several hours or more, such as email. As expected, low-latency anonymity systems often provide better performance and are intended for real-time applications, particularly Web browsing. Consequently, some have referred to high-latency anonymity systems as message-based systems, and low-latency anonymity systems as connection-based systems [33]. In [1], Chaum proposed a classic mix which simply collects incoming encrypted messages until it has received n messages and after receiving n messages, the mix then decrypts all of them and forwards them to their next destination in a random order. Another type of anonymity network is defined in [2], where rather than collecting a fixed number of messages like a threshold mix, a timed mix instead collects messages for a fixed length of time t. After t seconds have elapsed, the mix decrypts all messages it has received and then forwards them to their next destination in a random order. A threshold-or-timed mix is proposed in [4] in which it collects messages until either it has received n messages or until t seconds have elapsed since the last time the mix was flushed, whichever occurs first. Similarly, a threshold-and-timed mix collects messages until t seconds have elapsed and it has collected at least n messages. Instead of flushing all messages collected during the previous round, Pool mixes [5] select a random subset of the collected messages to flush and then retain the rest in the mix for the next round. Kesdogan et al. proposed a Stop-and-Go mix (SG-mix), in which Instead of batching messages together, SG-mixes individually delay messages as they pass

through the mix. Another anonymity network is defined in [6], in which it flushes a random fraction of messages it has collected over previous rounds rather than output a deterministic number of messages given the internal state of the mix and its parameters. Danezis et al. proposed RGB mixes [7] for detecting when an attacker is conducting the attack, and minimizing its negative impact on the anonymity of legitimate messages.

To overcome the high latency of mix-based systems, low-latency anonymity systems were introduced. One of such system is Anonymizer [34] which is based on the notion of a proxy. Proxies simply forward all incoming traffic (e.g., a TCP connection) immediately without any packet reordering. Thus it conceals the IP address of real user. PipeNet [35] was a low-latency anonymity system, in which it first selects a random sequence of servers in the network. Clients then set up a multiply encrypted tunnel by establishing a symmetric key with the first hop, tunneling through that encrypted connection and establishing another key with the second hop, and so on. Reiter et al. proposed Crowds [36] for anonymous web browsing. Users in the system are represented by processes called jondos. When the user's browser first makes a Web request, his or her jondo establishes a random path through the network by first randomly picking another jondo (perhaps even itself) from the crowd and forwarding the request to it. In [37], Berthold et al. proposed Java Anonymous Proxy (JAP) provide anonymity for interactive, real-time Internet applications, like Web browsing. The system consists of the Java Anonymous Proxy (JAP) software that runs on a client's computer and forwards client traffic into one of several available mix cascades. Freedman et al. proposed Tarzan [38], a low-latency anonymity system which is similar to Onion routing but uses UDP protocol. Rennhard et al. proposed MorphMix [39], a peer-to-peer low-latency anonymity system in which all nodes in the network relay traffic for other nodes. Along with these, other low-latency anonymity networks are used and

developed in recent times such as VPN [42], Proxy, Telex [43], and The Invisible Internet Project (I2P) [44].

B. Node Selection Analysis

Although different methods to attack TOR network have been already implemented, very less contribution to the field of analysis on end-to-end node selection probability has been made.

Zhen Ling et al. published a paper [20] in which they mentioned about the catch probability as a part of their project. But they didn't show the catch probability when an attacker inserts different types of nodes in the Tor network. They also failed to show the impact of compromised bandwidth and the impact of location of the compromised nodes in TOR network.

Zhen Ling et al. performed an extensive analysis of TOR Bridge discovery [19]. They donated some non-flagged compromised routers in TOR network to discover the total number of bridges which are used to enter into the TOR network.

Bauer et al. [21] showed that an adversary who controls only 6 malicious Tor routers can compromise over 46% of all clients' circuits in an experimental Tor network with 66 total routers.

Edman et al. [22] identified the risk associated with a single autonomous system (AS), which observes both ends of an anonymous Tor connection are greater than previously thought.

III. TOR NETWORK

A. Components of TOR

Tor is a network of virtual tunnels that allows people and groups to improve their privacy and security on the Internet. It is an open-source project and provides anonymity service for TCP applications [9]. As shown in Fig. 1, there are four basic components of Tor.

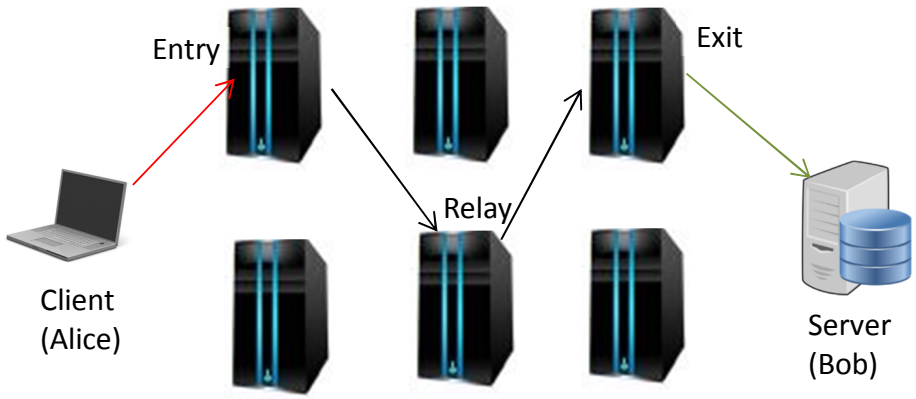

Fig. 1. Components of TOR Network.

1) Alice (i.e., Client): These are the normal users who requests service from web server through Tor network. To anonymize the client data into Tor, the client runs local software called onion proxy (OP).

2) Bob (i.e., Server): These are the web servers, located outside or inside the Tor network, which runs TCP applications such as a Web service. They respond to client's request by sending web pages.

3) Onion routers (ORs): Onion routers are special proxies that relay the application data between Alice and Bob. In Tor, transport-layer security

(TLS) connections are used for the overlay link encryption between two onion routers. The application data is packed into equal-sized cells of 512 B carried through TLS connections.

4) Directory Authorities (DA): They act like a database server of Tor network and hold all the authoritative information of all running onion routers such as public keys, bandwidth, nickname, uptime etc. of onion routers. Routers may act as directory caches to reduce the load on DAs. Directory caches download directory information of onion routers from authorities. A list of DAs is shipped with the Tor software, for a client to download the information of onion routers and build circuits through the Tor network. They are also responsible for assigning flags.

Tor network consists of roughly 5000 ORs pushing over 8 GiB/s in aggregate [11]. In Tor network, each onion router (OR) runs as a normal user-level process without any special privileges. Each OR maintains a transport layer security (TLS) [27] connection to every other OR. Fig. 2 shows the Tor design process of how the Tor clients build the circuit path through the Tor network. Step 1, at the initial stage before the Tor client (Alice) build the circuit path, she contacts the directory server and download the lists of the entire Tor relays available in the overlay network. Step 2, from the relay lists, Alice randomly select the relays and build the circuit path that consists of three hops. Normally, once a client starts up OP, Tor tries to maintain at least a certain number of clean circuits (one that has not yet been used for any traffic) by default path length of three, so that new streams can be handled quickly. This path length can be varied by changing the configuration file of the Tor client. But, by increasing the path length, the throughput of a circuit becomes slower. Step 3, after several times the Tor client (Alice) visits another sites, she can

choose to use the same path created initially or build new circuit path to reach the desired destination.

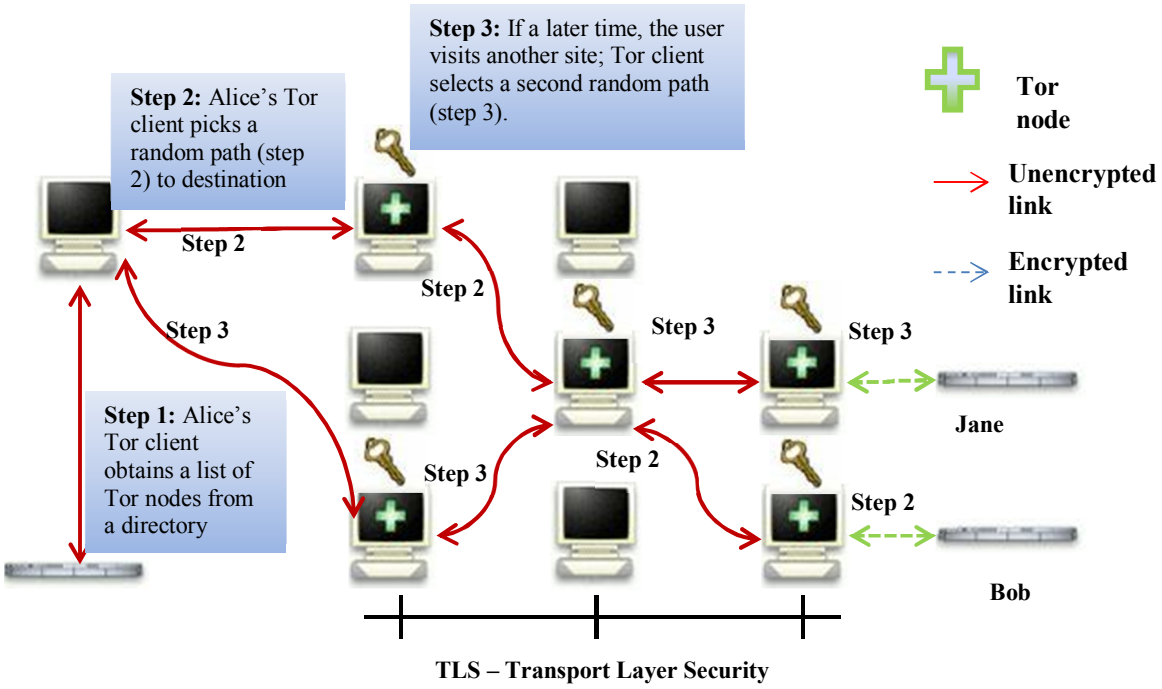

Fig. 2. TOR Design.

Each Tor user locally runs software, which is called an onion proxy (OP) to fetch directories, establishes the circuit across the network, and handles the connection from the user applications. These onion proxies accept the TCP streams and multiplex them across the circuits. The onion router (OR) on the other side of the circuit connects to the requested destinations and relays data packets. Each onion router maintains a short-term onion key [27]. The onion key is used to decrypt requests from users to set up a circuit and negotiate ephemeral keys. The TLS conceals data on the connection using an asymmetric cryptography for the key exchange, a symmetric encryption for privacy, to prevent an attacker from

modifying data. Details of transporting data packets from the clients to the destination are further explained in section 2-E.

B. TOR Streaming Cells

All data in Tor are sent in fixed size cells of 512 bytes. The cells are either the control cells that are always interpreted by the node that receives them, or the relay cells, which carry end-to-end data. Fig. 2 and Fig. 3 illustrate the cell format used by Tor.

For the control cell, the header includes a circuit identifier (circID) that specifies which circuit the cell refers to (many circuits can be multiplexed over the single TLS connection), and a command to describe what to do with the cell's payload. The circuit identifiers are connection-specific; each circuit has a different circID on each OP/OR or OR/OR connection it traverses. The control cell commands (CMD) are padding (currently used for keep alive, but also usable for link padding); create or created are used to set up a new circuit, and destroy are to tear down a circuit.

The relay cells have an additional header, which consists of the relay commands header at the front of the payload. The relay commands are: 1. Relay data for carrying data. 2. Relay begins to open the stream. 3. Relay end to close the stream cleanly. 4. Relay teardown to close a broken stream. 5. Relay connected, to notify the OP that relay begin of connection has succeeded. 6. Relay extends and relay extended (to extend the circuit by a hop and to acknowledge). 7. Relay truncate and relay truncated to tear down only part of the circuit and to acknowledge. 8. Relay sendme used for the congestion control. 9. Relay drop used to implement long-range dummies.

The recognized header field has 2 bytes which is always set zero in any unencrypted relay payload. The Digest header field is computed as the first four

bytes that have been destined for the hop of the circuit or originated from the hop of the circuit, seeded from the forward or backward respectively. When the recognized header field of a relay cell is zero, and the digest is correct, the cell is considered “recognized” for the purposes of decryption.

The length field of a relay cell contains the number of bytes in the relay payload which contains real payload data. The remainder of the payload is padded with NULL bytes. “StreamID” header holds the stream identifier as many streams can be multiplexed over a circuit, and an end-to-end checksum for integrity checking, the length of the relay payload, and a relay command. “StreamIDs” are arbitrarily chosen by the OP. Relay cells that affect the entire circuit rather than a particular stream use the “StreamID” zero. When an OR receives a relay send me cell with “StreamID” zero, it increments its packaging window. This situation happens when OR is willing to deliver more cells. When new anonymous TCP connection are build, the OP chooses an open circuit to an exit that may be able to connect to the destination address. Selects an arbitrary “StreamIDs” not yet used on that circuit, and constructs the relay begin cell with a payload encoding the address and port of the destination host [25].

The “StreamID” is assigned, as when all node on receiving a relay cell looks up the corresponding circuit. Decrypts the relay header and payload with the session key for that circuit, and replaces the circuit ID of the header with the correspondence successor nodes. Then forwards the decrypted cell to the next OR [26].

The entire contents of the relay header and the relay cell payload are encrypted or decrypted together as the relay cell moves along the circuit, with using 128-bit AES cipher together in counter mode to generate a cipher stream [9]. To construct a relay cell addressed to a given ORs, the clients assign the digest, and then iteratively encrypt the cell payload (that is, the relay header and payload) with the

symmetric key of each hop up to the OR. Upon receiving a relay cell, the OR looks up the corresponding circuit, and decrypts the relay header and payload with the session key for the circuit. When the OR replies to the client with a relay cell, it encrypts the cell's relay header and payload with the public key, which the OR initially shares with the client, and sends the cell toward the client along the circuit.

Fig. 3. Tor Cell Format (Control cell)

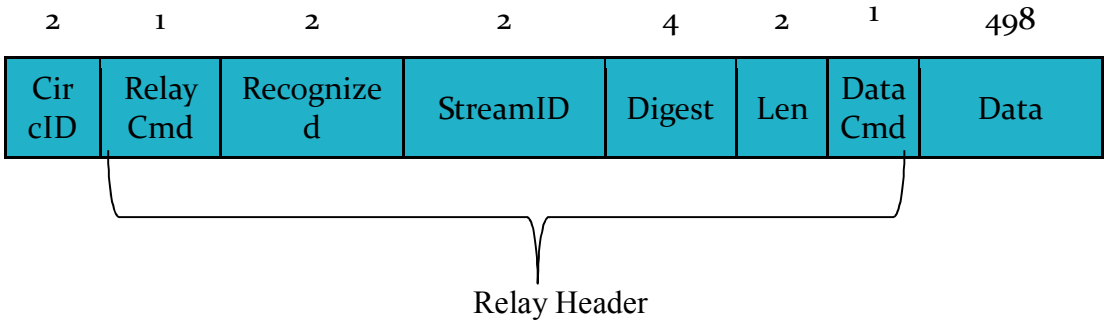

Fig. 4. Tor Cell Format (Relay cell)

C. Status Flags (Types of Nodes)

Relays have status flags assigned to them by DAs, which clients consider when choosing relays for a circuit. The type of flags assigned to the relays makes the type of nodes in Tor network. DA assigns GUARD flag to relays whose uptime is at least the median for familiar relays, and if their bandwidth is at least the minimum of 250 KiB/s and the median relay bandwidth. Clients choose and maintain three active guards and use them as the entry relay for all of their circuits

to reduce the chance of directly connecting to an adversary. Clients keep each guard for 30 to 60 days. DA assigns EXIT flag to relays who allow direct connections with external web servers. The ORs with Exit flag set individual exit policies specifying the IP address ranges and port ranges to which they are willing to connect. Clients use these policies to determine which relay to choose for the final position in each circuit they make. Guards and exits are weighted highly for the entry and exit position in a circuit, respectively, since all relays do not fulfill the requirements to obtain those flags. Additionally, DA assigns a relay with a STABLE flag if its weighted mean time before failure is at least the median for known active relays. Clients building streams to a port among the long-lived ports list must choose stable relays in each position of the circuit. Finally, clients not choose two relays from the same /16 subnet or family for the same circuit. A family is a set of relays that mutually indicate that they belong to a group together. The GUARD+EXIT flag is assigned to relays that have acquired EXIT flag in the beginning and have fulfilled the requirements of GUARD flag as well [14].

D. Tor Configuration File

The Tor installs a text file called *Torrc that* contains configuration instructions for how the Tor program should behave [27]. *Torrc* file can be changed by using software called Vidalia. Fig. 5 shows the *Torrc* file that holds the configurations which specify the algorithm for building circuit path, such as the sets of lists for entry, middle and exit stable guard and fast relays to build the circuit path and perform download and upload for the clients. Every installation of Tor systems includes a server and communication protocol, used to control all aspects of client's operation and can be done in *Torrc* file. However, there are several variables (e.g. DisablePredictedCircuits) which can only be set through control interface. With the *Torrc* interface, the client can modify the Tor's configuration on

how the circuits can be build and perform other communication operations. The *Torrc* configuration file holds the instructions such as the control port 9051, the socks listen address 127.0.0.1, and unique Hash Control Password etc., for controlling the virtual circuit build through the Tor network. In addition, *Torrc* holds the SocksPort 9050 to provide a generic interface for TCP proxies and requests connection to another address and port to neighboring allocated relays.

```
# The port on which Tor will listen for local connections from Tor
ControlPort 9051
SocksListenAddress 127.0.0.1
SocksPort 9050
# Try for at most NUM seconds when building circuits. If the circuit isn't open
in that time, then drop it.
CircuitBuildTimeout (time values)
#send a padding cell every N seconds to keep firewalls from closing the
connections while Tor is not in use.
KeepalivePeriod (time values)
//Entry Nodes – {lists of entry stable guarding and fast relays nodes} (1st Hop)
//StrictEntryNodes
//TestVia {lists of middle stable guarding and fast relays nodes} (2nd Hop)
//ExitNodes {lists of exit stable guarding and fast relays nodes} (3rd Hop)
//StrictExitNodes
```

Fig. 5. Overview of client Torrc Configuration script file

E. Circuit Creation and Data Transmission

Before making a circuit, a client chooses three nodes as path for a circuit. To ensure the performance of circuits and load balancing, Tor adopts weighted bandwidth routing algorithms [14] for choosing these three nodes. Tor currently utilizes a set of trusted Bandwidth Authorities which are responsible for actively probing the Tor routers and estimating each router's capacity [15]. Additional constraints are placed on router selection, including the use of entry guards [16] for the first hop to defend against the predecessor attack [17] and exit policies that specify the destination addresses and ports allowed by an exit router's operator. After choosing three nodes, a client makes a circuit to the destination as follows: a client first set up a TLS connection with OR1 using the TLS protocol. Then, using this connection, Client sends a CELL_CREATE cell and uses the Diffie–Hellman (DH) handshake protocol to negotiate a symmetric key $K = g^{xy}$ with OR1. OR1 responds with a CELL_CREATED cell upon succession [18]. In this way, a 1-hop circuit C1 is created. Now, Client extends this circuit to a 2-hop circuit by sending CELL_EXTEND. This command is wrapped up by CELL_CREATE in the first hop. The middle OR then responds with a CELL_EXTENDED cell. In the same way, client extends this circuit to 3-hop. After the circuit is set up between the client and OR3, client sends a RELAY_COMMAND_BEGIN cell to the exit OR, and the cell is encrypted as, $\{\{\{Begin <website,\ Port\ no.>\}kf_3\}kf_2\}kf_1$, where the subscript refers to the key used for encryption of one onion skin. The three layers of encryption are removed one by one each time the cell traverses an OR through the circuit. When OR3 removes the last onion skin by decryption, it recognizes that the request intends to open a TCP stream to a port at the destination IP, which belongs to Bob. Therefore, OR3 acts as a proxy, sets up a TCP connection with Bob, and sends a RELAY_COMMAND_CONNECTED cell back to the client. Then, the client can download the file.

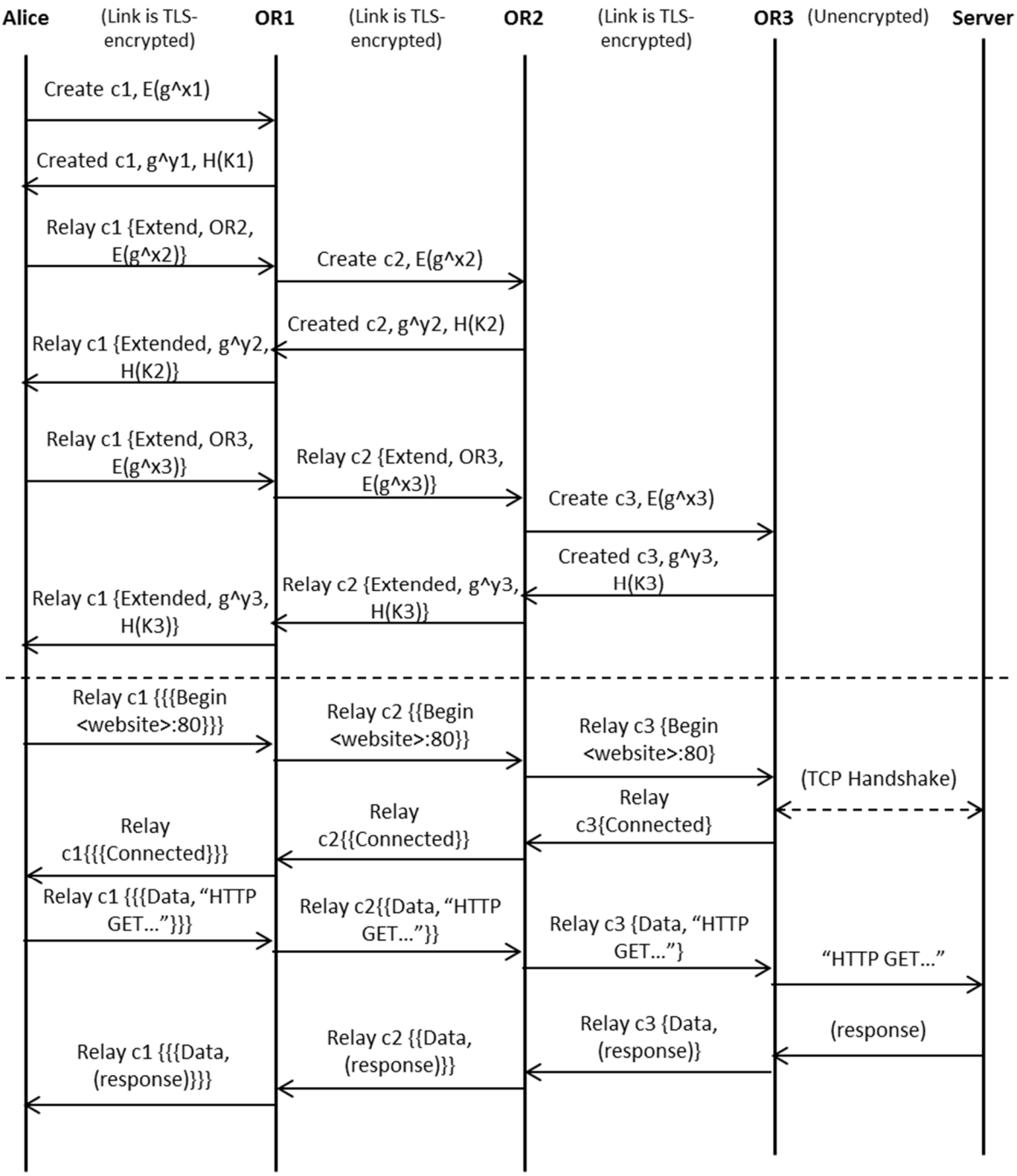

Fig. 6. Circuit Creation and Data Transmission.

F. Processing Cells at Onion Routers

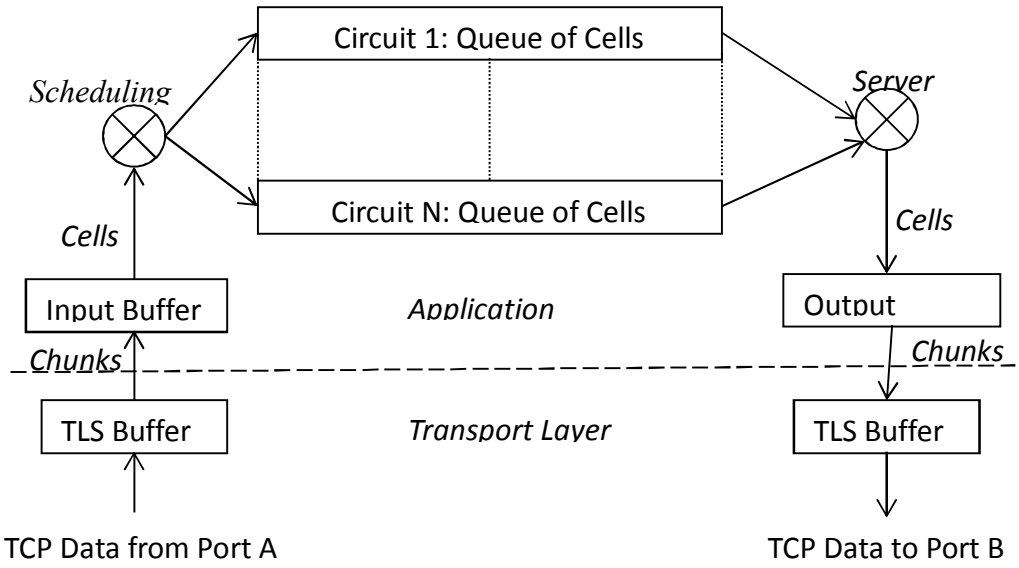

Fig. 7. Processing the cells at onion routers.

Fig. 7 illustrates the procedure of processing cells at onion routers. It is noteworthy to mention that the cells mentioned below are all CELL_RELAY_DATA cells, which are used to carry end-to-end stream data between Alice and Bob. To begin with, the onion router receives the TCP data from the connection on the given port A. After the data is processed by TCP and TLS protocols, the data will be delivered into the TLS buffer of the connection. When there is pending data in the TLS buffer, the read event of this connection will be called to read and process the data. The connection read event will pull the data from the TLS buffer into the connection input buffer. Each connection input buffer is implemented as a linked list with small chunks. The data is fetched from the head of the list and added to the tail. After the data in the TLS buffer is pulled into the connection input buffer, the connection read event will process the cells from the connection input buffer one

by one. As stated earlier, the cell size is 512 B. Thus, 512-B data will be pulled out from the input buffer every time until the data remaining in the connection input buffer is smaller than 512 B. Since each onion router has a routing table that maintains the map from source connection and circuit ID to destination connection and circuit ID, the read event can determine that the transmission direction of the cell is either in the forward or backward direction. Then, the corresponding symmetric key is used to decrypt/encrypt the payload of the cell, replace the present circuit ID with the destination circuit ID, and append the cell to the destination circuit queue. If it is the first cell added to this circuit queue, the circuit will be made active by being added into a double-linked ring of circuits with queued cells waiting for a room to free up on the output buffer of the destination connection. Then, if there is no data waiting in the output buffer for the destination connection, the cell will be written into the output buffer directly, and then the write event of this circuit is added to the event queue. Subsequent incoming cells are queued in the circuit queue.

When the write event of the circuit is called, the data in the output buffer is flushed to the TLS buffer of the destination connection. Then, the write event will pull as many cells as possible from the circuit queue of the currently active circuit to the output buffer and add the write event of this circuit to the event queue. The next write event can carry on flushing data to the output buffer and pull the cells to the output buffer. In other words, the cells queued in the circuit queue can be delivered to the network via port B by calling the write event twice.

G. Bandwidth Weighted Node Selection

For load balancing in Tor nodes, Tor adopts weighted bandwidth routing algorithms. First, from a set of exit routers, i.e. pure exit routers and EE routers, a client chooses an appropriate exit onion router OR3. The bandwidth of exit routers

is weighted as follows: Assume that B is the total bandwidth, B_E is the total exit bandwidth, and B_G is the total entry bandwidth. If $B_E < (B/3)$, i.e., the bandwidth of exit routers is scarce, the exit routers will not be considered for non-exit use. The bandwidth of EE routers are weighted by, $W_G = 1 - (B/3B_G)$, where W_G is the bandwidth weight of entry routers and $B_G > (B/3)$.

If $B_G < (B/3)$, then $W_G = 0$. The probability of selecting the i^{th} exit router from the exit set is $B_{iE}/(B_{exit} + B_{EE} * W_G)$, where B_{EE} is the total bandwidth of EE routers, B_{exit} is the total bandwidth of pure exit routers and B_{iE} indicates the bandwidth of the i^{th} exit router . Note that $B_E = B_{exit} + B_{EE}$. Second, the client chooses an appropriate entry onion router OR1 from the set of entry routers, including the pure entry routers and EE routers. To ensure sufficient entry bandwidth, if $B_G < (B/3)$, the entry routers will not be considered for non-entry use. Then, the probability of selecting the i^{th} entry router from the entry set is $B_{iG}/(B_{entry} + B_{EE} * W_E)$, where $W_E = 1 - (B/3B_E)$, where is the exit bandwidth weight and B_{iG} is the i^{th} bandwidth in the entry set. If $B_E < (B/3)$, then $W_E = 0$. Eventually, the client chooses the middle from the rest of Tor routers [19].

Let us denote e_1 as the number of compromised exit routers, e_2 as number of compromised entry routers and e_3 as number of compromised EE routers. Let b be the bandwidth assigned to each of the compromised router. Based on the above weighted bandwidth selection algorithm, the weight can be derived by:

$$W_E = \begin{cases} 1 - \dfrac{B}{3.(B_E + B_{EE} + (e_1 + e_3) * b)} : W_E > 0 \\ 0 : W_E \leq 0 \end{cases}$$

$$W_G = \begin{cases} 1 - \dfrac{B}{3.(B_G + B_{EE} + (e_2 + e_3) * b)} : W_G > 0 \\ 0 : W_G \leq 0 \end{cases}$$

Then, the catch probability (end-to-end node selection probability) can be calculated as follows:

$$P(e) = \frac{e_1 \cdot b}{B_E + W_G * (B_{EE} + e_3 \cdot b) + e_1 \cdot b} \cdot \frac{(W_E \cdot e_3 + e_2) \cdot b}{B_G + W_E * (B_{EE} + e_3 \cdot b) + e_2 \cdot b}$$

$$+ \frac{W_G \cdot e_3 \cdot b}{B_E + W_G * (B_{EE} + e_3 \cdot b) + e_1 \cdot b} \cdot \frac{(W_E \cdot (e_3 - 1) + e_2) \cdot b}{B_G + W_E * (B_{EE} + (e_3 - 1) \cdot b) + e_2 \cdot b}$$

Probability (at least 1 time among n) = 1- (1 - P (e))^n, where n = number of circuits.

H. Equation Based Results

Table 1. Parameters of Live TOR Network

Bandwidth of Guard routers (B_G) = 3822	Total Bandwidth (B) = 7688 MB/s
Bandwidth of Exit routers (B_E) = 478	No. of exit router = 833
Bandwidth of EE routers (B_{EE}) = 2201	No. of entry router = 1750
Bandwidth of N-EE routers ($B_{N\text{-}EE}$) = 1187	Total router = 4569

Using the above equation and parameters of TABLE I., the catch probabilities for different number of circuits were calculated. Four different cases were considered. In first case, $B_G > B/3$ & $B_E < B/3$, was taken, in second case $B_G < B/3$ & $B_E > B/3$, in third case, $B_G > B/3$ & $B_E > B/3$ and finally $B_G < B/3$ & $B_E < B/3$. For each case, different combinations of the compromised nodes were looked at, with the total bandwidth remaining same. Following graphs were obtained.

For n=10

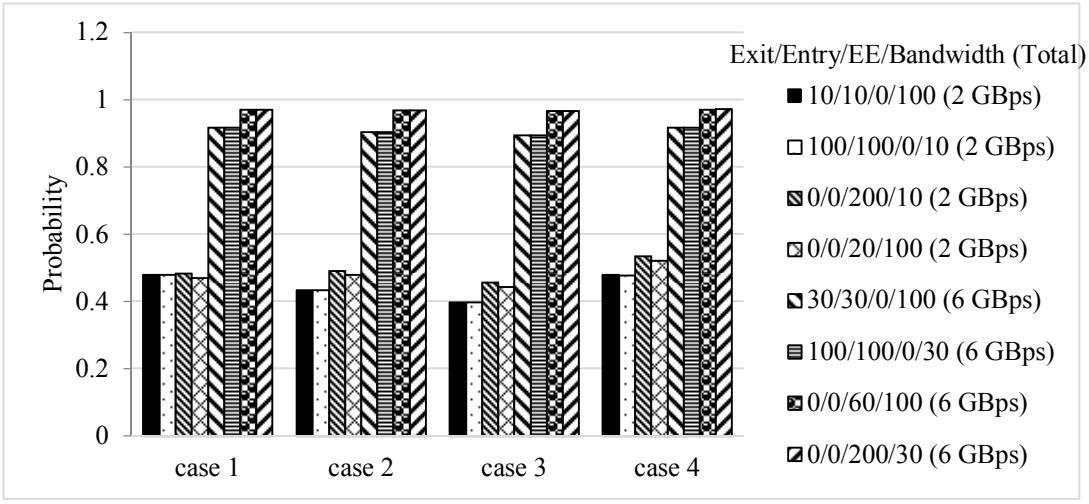

Fig. 8. Catch Probability for n=10.

For n=20

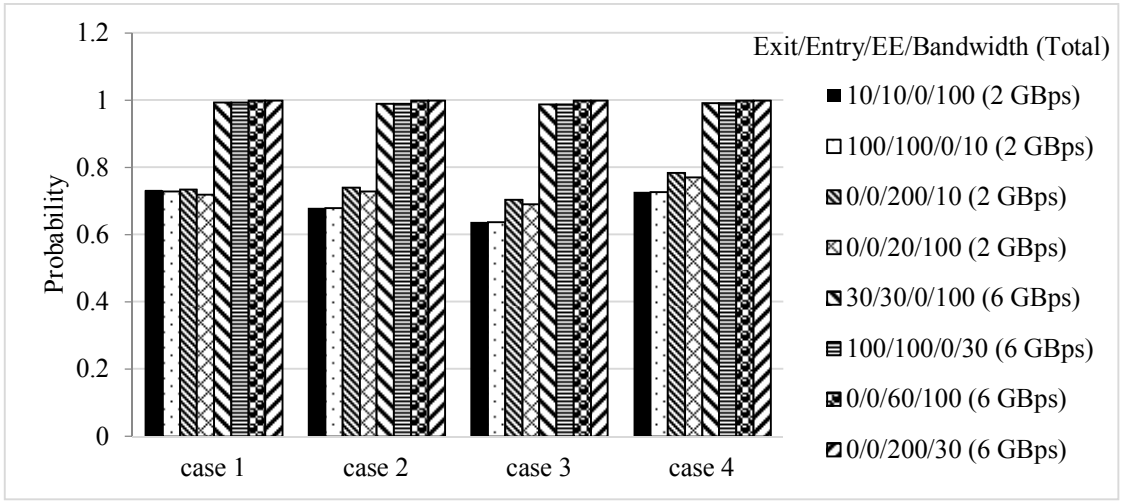

Fig. 9. Catch Probability for n=20.

For n=40

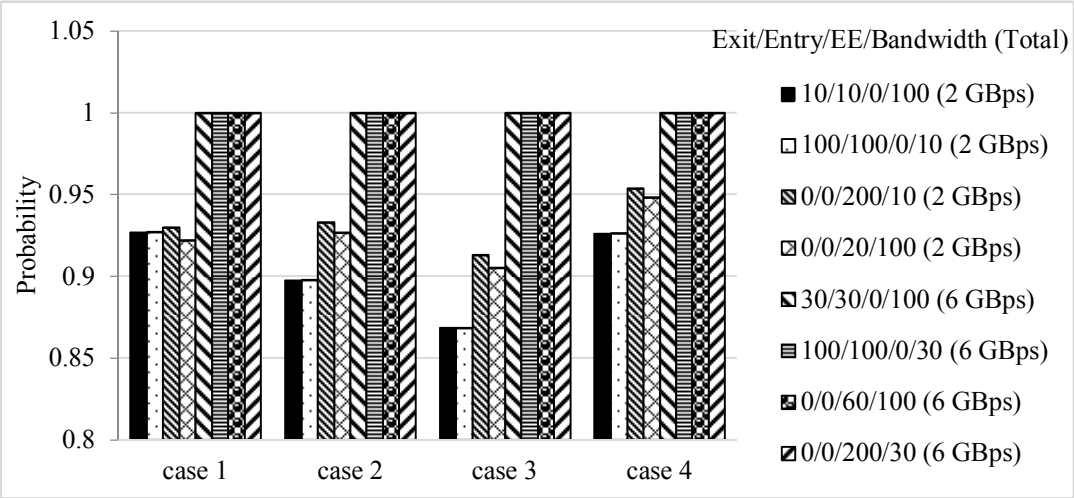

Fig. 10. Catch Probability for n=40.

Fig. 8, 9 and 10 showed the results obtained by using mathematical equation. From all of the graphs, it can be observed that, if number of malicious EE routers are increased and number of other two routers are zero then catch probability is high. Also although the bandwidth is high, less number of malicious EE routers means less probability i.e. If the number of malicious EE router is high and bandwidth is low the catch probability is high.

Equation based result gives the ideal output. It only considers the number of compromised nodes and their bandwidth. It doesn't consider the other parameters such as location of nodes, packet loss, jitter, delay etc. Hence, an environment is required which can consider all these parameters and calculate the selection probability as well as verify the math based result. One of such environments is Shadow simulator, designed to run Tor network.

IV. SHADOW SIMULATOR

Shadow [23, 24] is a unique, open source discrete-event network simulator that runs real applications like Tor and distributed systems of thousands of nodes on a single machine. Shadow contains plugins, which are the libraries that are linked to real applications, to natively execute the application code. Scallion is a shadow plug-in that simulates the Tor anonymity network. It wraps the Tor source code to integrate Tor into shadow. An overview of Shadow's design [23] is depicted in Fig. 11.

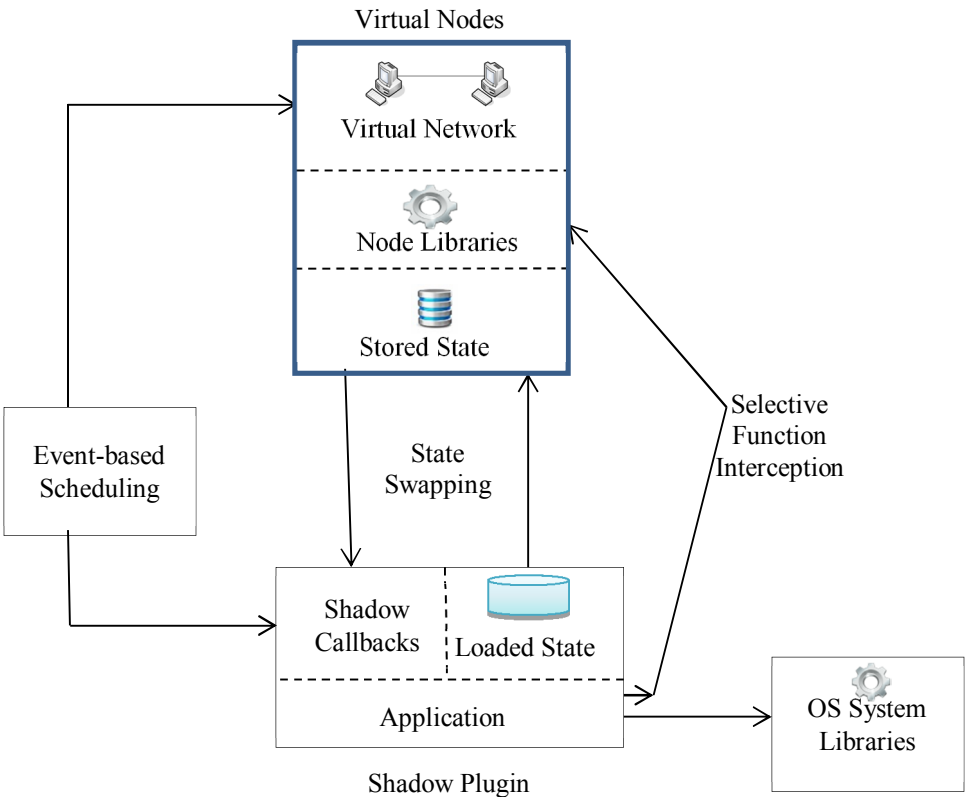

Fig. 11. Shadow's architectural design.

Shadow dynamically loads plug-ins and instantiates virtual nodes as specified in a simulation script. Communication between Shadow and the plug-in is done through a well-defined callback interface implemented by the plug-in. When the appropriate callback is executed, the plug-in may instantiate and run its non-blocking application(s). The application will cause events to be spooled to the scheduler by executing a system call that is intercepted by Shadow and redirected to a function in the node library. The interceptions allow integration of the application into the simulation environment without requiring modification of application code. Virtual nodes communicate with each other through a virtual network which spools packet and other network related events to the scheduler. Each virtual node stores only application-specific state and loads/unloads the state as necessary during simulation execution [25].

The working procedure of Shadow is shown in Fig. 12.

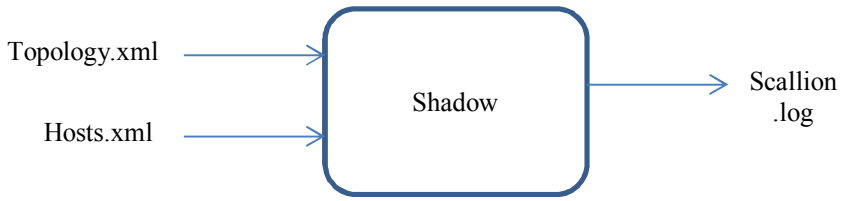

Fig. 12. Shadow working procedure.

Topology.xml file creates a network for downloading a file requested by a client through Tor network. The virtual nodes contain properties such as downstream bandwidth, upstream bandwidth and packet loss. The link between two virtual nodes has properties such as latency, jitter and packet loss. Shadow.config.xml file creates client, relays, authority and webserver among the virtual nodes. After the simulation, all the results are stored in a log file called scallion.log.

V. PERFORMANCE EVALUATION

A. Simulation Environment

To maintain the privacy and anonymity of live Tor network, the simulation was performed in Shadow Simulator. 4 numbers of Directory Authorities, 116 Guard relays, 262 Middle relays, 76 Exit relays and 41 Guard+Exit relays were taken in the simulation. Similarly, there were 1800 clients and 500 servers and the total time of simulation was 60 virtual minutes. This simulation consumed around 61 GiB of RAM. Each experiment took about 5-6 hours to complete. The Table 2 below presents the total bandwidth (BW) per type of node.

Table 2. Simulation Parameters

Total BW per type of node		
Node	**BW (MB/S)**	**% w.r.t Total BW**
Directory Authority	0.25	0.05%
Guard	258.15	50.75%
Exit	13.23	2.60%
Guard + Exit(EE)	213.40	41.95%
Middle	23.69	4.66
Total	**508.72 MB/S**	
Compromised Nodes	**400**	**78.63%**

During simulation, the total bandwidth, contributed by the compromised nodes, were kept to be same. For e.g., if a compromised node of 5 MB/S was taken, then 80 numbers of compromised nodes were donated to make the total compromised bandwidth of 400 MB/S. Similarly, for 10 MB/S, total numbers of compromised

nodes taken were 40 to make total compromised bandwidth of 400 MB/S. Generally, two cases were considered: 1) Guard and Exit (G-E) pair 2) Guard-Exit (EE) exclusively. For the first case, X numbers of guard nodes and Y numbers of exit nodes were added. Again, for the second case, (X+Y) EE nodes were added in the existing Tor network, to make the condition equal.

B. Performance Results and Discussion

For results, the bar graph of the bandwidth of a single compromised node vs end-to-end node selection probability was plotted, such that the values of bandwidth in x-axis represents the bandwidth of each compromised nodes following that total number of compromised nodes * bandwidth = total bandwidth added by compromised nodes to existing Tor network. To calculate the selection probability, a Perl script was written which parses the result of the shadow simulator to give the selection probability.

1. End-to-End Selection Probability

For n=1

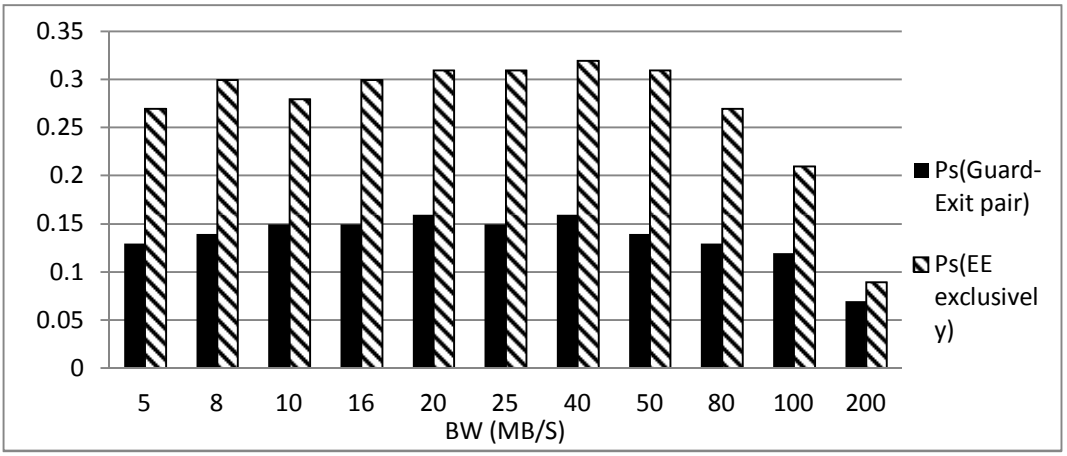

Fig. 13. End-to-End Selection Probability when guard- exit pair and EE node are added to the Tor network for n=1.

Fig. 13 shows the bar graph of node selection probability that a client chooses compromised node in guard position as well as exit position during circuit creation. The figure shows two graphs showing selection probability for two cases which are G-E pair and EE exclusively. It is observed that adding EE nodes in the Tor network yields higher end-to-end selection probability of compromised nodes than the adding Guard and exit nodes separately. This is due to the bandwidth weights, which are described in section II-E, for different types of nodes.

For n=10

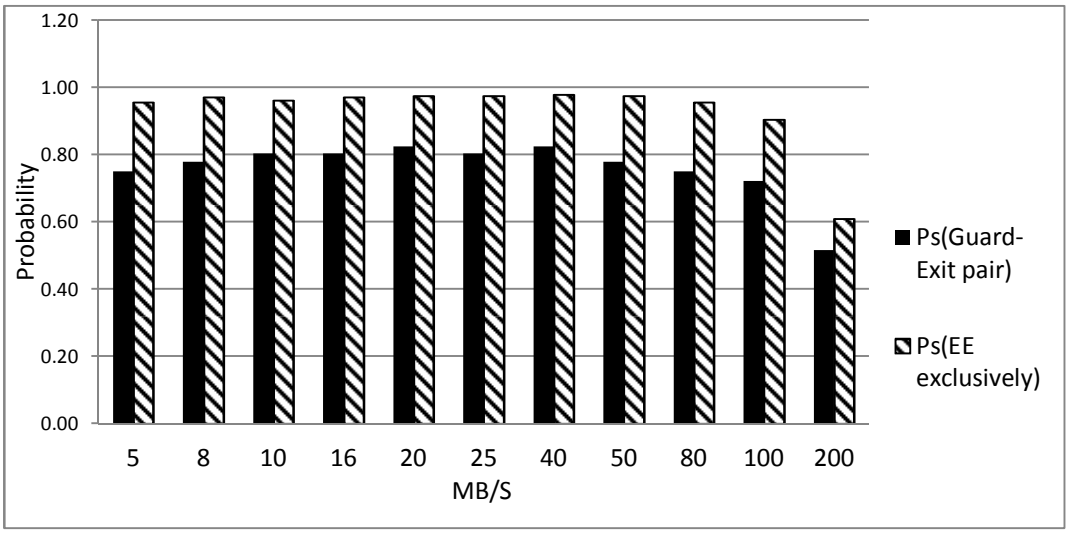

Fig. 14. End-to-End Selection Probability for n=10.

Fig. 14, 15 and 16 shows the end-to-end selection probability for 10, 20 and 40 number of circuits respectively. It is observed that increasing the number of circuits increases the selection probability.

For n=20

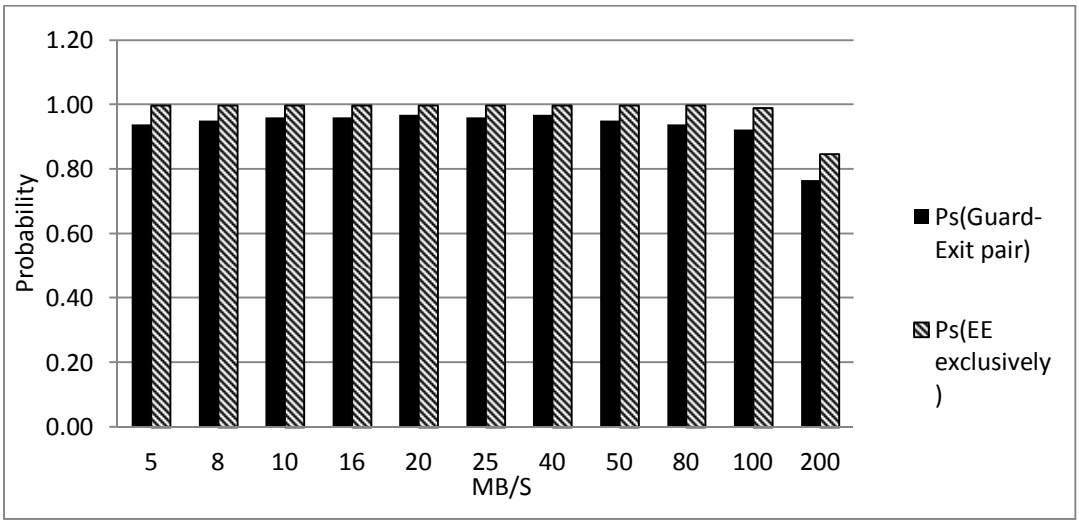

Fig. 15. End-to-End Selection Probability for n=20.

For n=40

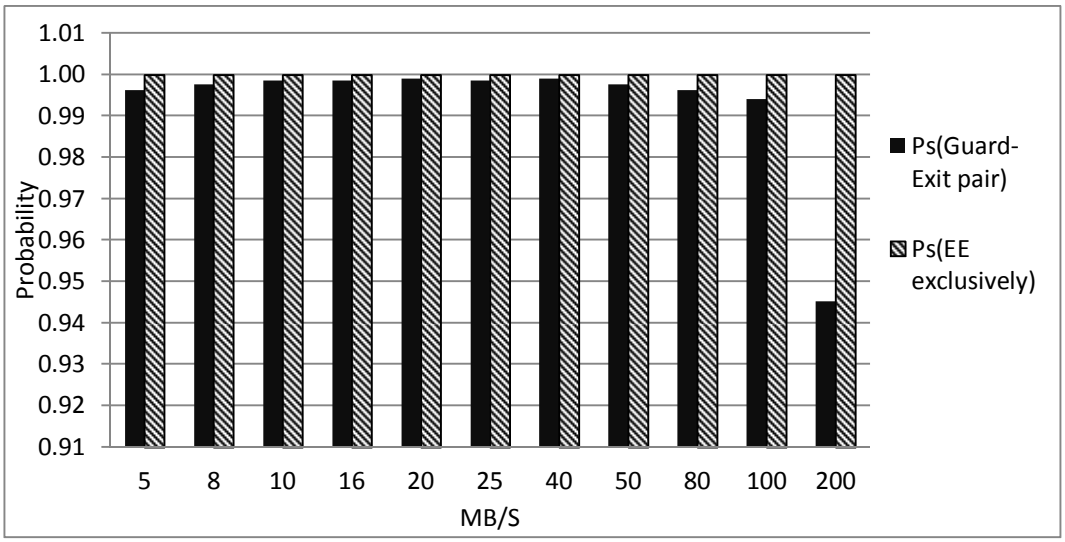

Fig. 16. End-to-End Selection Probability for n=40.

Fig. 17 shows the bar graph showing the probability of selection of compromised nodes in both guard and exit position, when EE nodes, placed in different geographical locations are added to the Tor network. It is observed that the selection probability doesn't depend upon the location of the compromised nodes.

In simulator, Director Authority (DA) does not measure the node Bandwidth. The bandwidths of the nodes are defined initially in the simulator, which appears on the consensus document. But in live Tor network, DA measures the BW of the node and if the distance between node and DA is longer, actual BW may not appear in the consensus. This affects the selection probability.

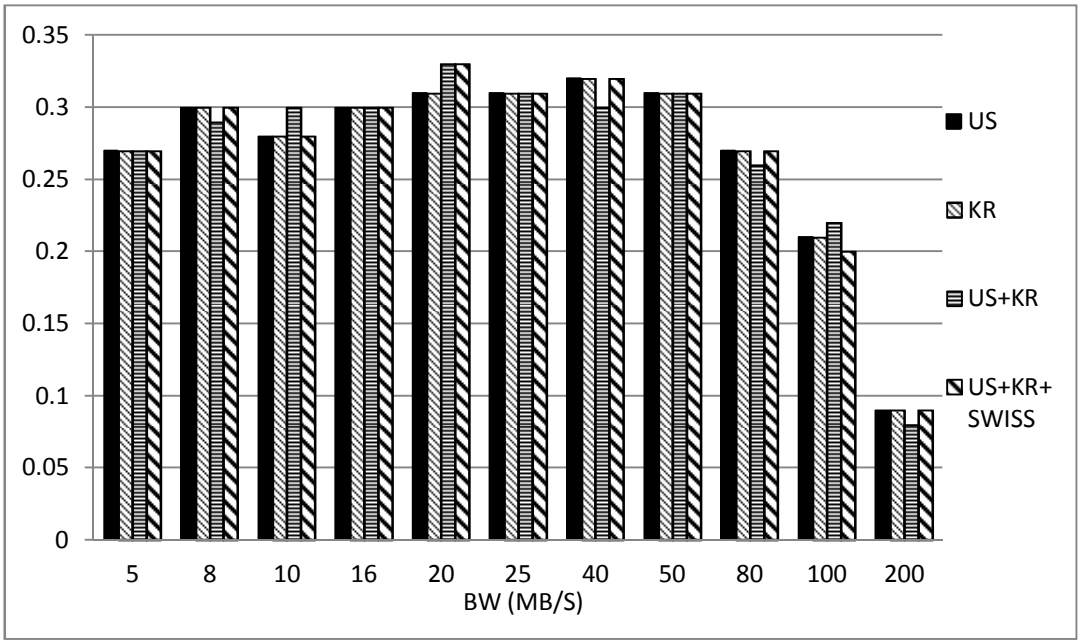

Fig. 17. End-to-End Selection Probability when EE nodes, in different locations, are added to the Tor network.

2. Guard Selection Probability

For n=1

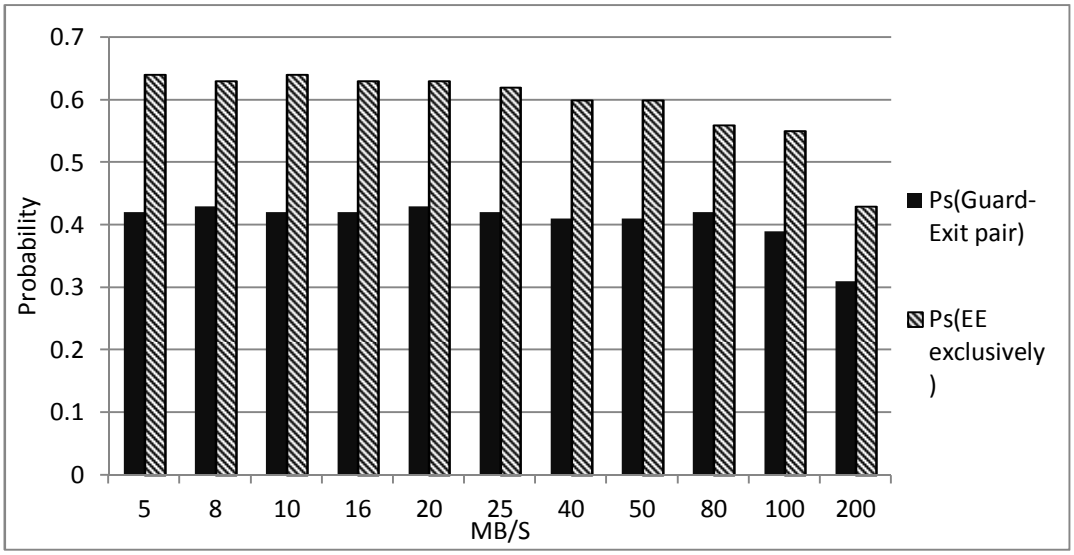

Fig. 18. Guard selection Probability when guard-exit pair and EE node are added to the Tor network for n=1.

Fig. 18 shows the bar graph of node selection that a client chooses compromised node in guard position only while creating a circuit. The figure shows two graphs showing selection probability for two cases, which are G-E pair and EE exclusively. From the graph, it is seen that adding EE exclusively nodes only in the existing Tor network yields higher selection probability of compromised nodes in guard position than G and E nodes separately. The probability of guard node selection is important because during discovery of hidden service, the targeted hidden service must use our compromised guard during circuit creation to the client.

For n=10

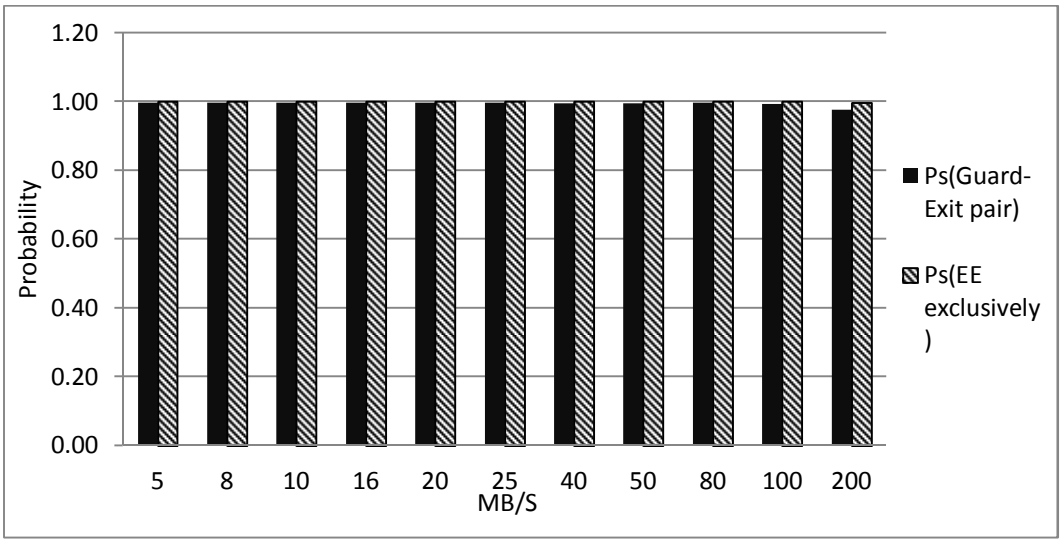

Fig. 19. Guard selection Probability for n=10.

For n=20

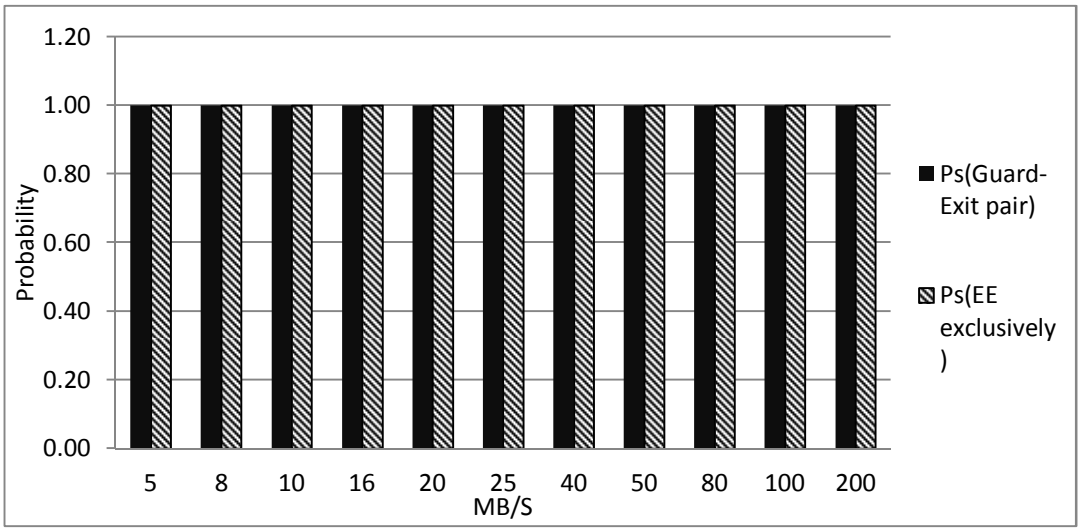

Fig. 20. Guard selection Probability when guard-exit pair and EE node are added to the Tor network for n=20.

Fig. 19 and 20 shows the guard selection probability for 10 and 20 numbers of circuits respectively. It is observed that increasing the number of circuits increases the selection probability.

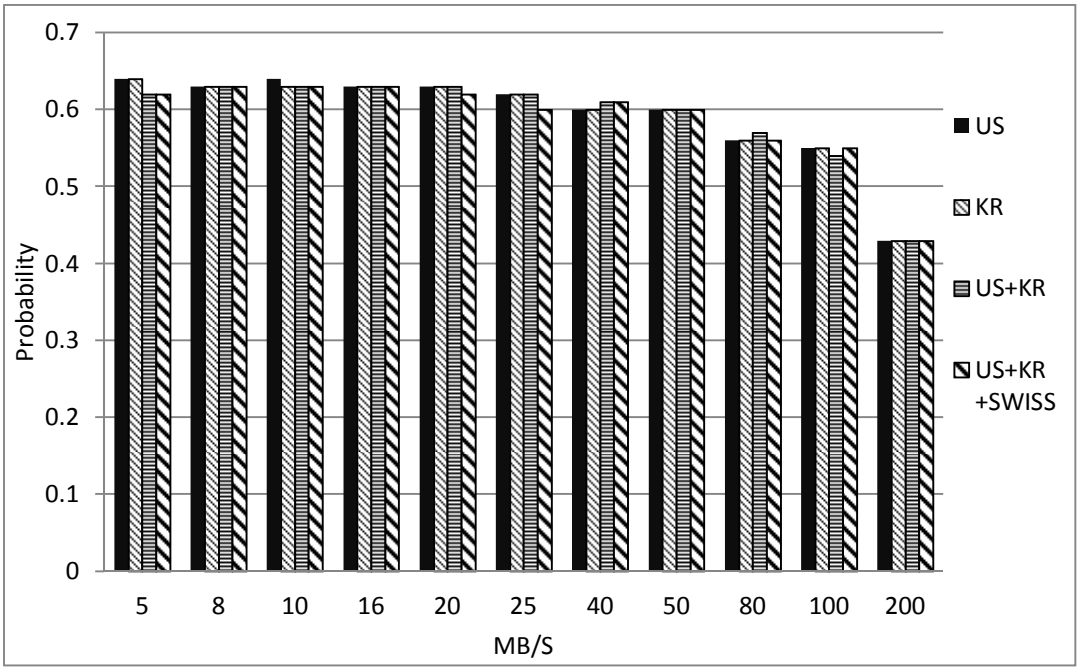

Fig. 21. Guard selection Probability when EE nodes, in different locations, are added to the Tor network.

Fig. 21 shows the guard selection probability when EE exclusively nodes, in different geographical locations, are added to the Tor network. Unlike Fig. 15, it can be observed that the guard selection probability doesn't depend on the geographical locations of the compromised nodes.

From all of the figures, it is observed that inserting a single or very few numbers of compromised nodes having very high bandwidth yields less selection probability than inserting large number of nodes with smaller bandwidth. It is seen that if just two compromised nodes with a very high bandwidth of 200 Mbytes/s are

contributed, selection probability is low in comparison to adding more numbers of compromised nodes with a smaller bandwidth of 10 Mbytes/s to 40 Mbytes/s, provided that total bandwidth contributed by the compromised nodes to the existing Tor network remains equal. It can be also observed that if the attacker inserts EE exclusively node only, then the attacker can control more than 60% of the total entry traffic going through Tor network, which is a significant threat to the anonymity provided by Tor.

3. Prediction of number of nodes required for certain probability

Table 3. Prediction for number of nodes

Entry-Exit Pair

		Target end-to-end node selection, Ps (%)		
		50%	**80%**	**99%**
# of nodes required of 5 MB/S	n=1	308	492	609
	n=10	40	92	228
	n=20	22	47	129
	n=40	11	25	68

		Target end-to-end node selection, Ps (%)		
		50%	**80%**	**99%**
# of nodes required of 10 MB/S	n=1	133	213	264
	n=10	17	40	99
	n=20	9	21	56
	n=40	5	11	29

		Target end-to-end node selection, Ps (%)		
		50%	**80%**	**99%**
# of nodes required of 20 MB/S	n=1	63	100	124
	n=10	8	19	47
	n=20	5	10	27
	n=40	3	5	14

EE Node

		Target end-to-end node selection, Ps (%)		
		50%	**80%**	**99%**
# of nodes required of 5 MB/S	n=1	148	237	294
	n=10	20	45	110
	n=20	11	24	63
	n=40	5	12	33

		Target end-to-end node selection, Ps (%)		
		50%	**80%**	**99%**
# of nodes required of 10 MB/S	n=1	72	115	142
	n=10	10	22	53
	n=20	5	12	30
	n=40	3	6	16

		Target end-to-end node selection, Ps (%)		
		50%	**80%**	**99%**
# of nodes required of 20 MB/S	n=1	33	52	64
	n=10	5	10	24
	n=20	3	5	14
	n=40	2	3	8

4. Prediction of number of nodes required for certain probability

If the target selection is desired as 90%, then following table shows the Percentage of BW required of all compromised nodes for end-to-end selection probability:

Table 4. Prediction for bandwidth required

Entry-Exit Pair

MB/S / No. of circuits	5	10	20	100
n=1	462.00%	390.00%	409.50%	572.00%
n=10	126.00%	109.20%	102.38%	136.5%
n=20	66.00%	57.2%	53.63%	78.50%
n=40	33.00%	28.6%	26.81%	35.75%

EE Node

MB/S / No. of circuits	5	10	20	100
n=1	210.89%	200.57%	173.62%	293.43%
n=10	60.67%	58.50%	52.84%	78.00%
n=20	31.78%	30.64%	27.68%	40.86%
n=40	15.89%	15.32%	13.84%	20.42%

The above table shows the prediction for bandwidth required to achieve 90% selection probability for G-E pair and EE exclusively. The table data can be explained as follows: from the simulation, it is seen that if we insert a G-E pair node of 5 MB/S, with total compromised bandwidth of 400 MB/S, then only 0.13 catch probability is obtained. So, to increase this catch probability up to 90%, the total bandwidth of compromised 5 MB/S nodes should be increased by nearly 4.5 times. It can be seen that, on increasing the bandwidth the total bandwidth required decreases but for higher bandwidth, the total compromised bandwidth increases.

VI. CONCLUSIONS

In this thesis, an analysis on catch probability i.e. the probability that a client chooses the compromised nodes while making a circuit in Tor network was presented. Tor is a low latency anonymous communication system that supports TCP application over the internet. From the Shadow based simulation, the thesis outlined that end-to-end node selection probability of EE nodes is higher than that of individual Guard and Exit nodes. It is also noteworthy that the selection probability of the node doesn't depend upon the geographical location of the node. In terms of BW, ranging from 10Mbytes/s to 50Mbytes/s as per a compromised node's BW could be a good choice for better selection probability. Along with the end-to-end selection probability, the guard node selection probability of the compromised nodes was also stated. Finally, a prediction for the number of nodes that are required for certain percentage of catch probability and prediction for the percentage of bandwidth required for obtaining 90% selection probability were evaluated.

BIBLIOGRAPHY

[1] D. Chaum, “Untraceable electronic mail, return addresses, and digital pseudonyms,” ACM Communication, Vol. 24, No. 2, pp. 84-88, 1981.

[2] A. Serjantov and R. E. Newman, "On the anonymity of timed pool mixes," Security and Privacy in the Age of Uncertainty, Springer US, pp. 427-434, 2003.

[3] A. Serjantov, R. Dingledine, and P. Syverson, "From a trickle to a flood: Active attacks on several mix types," Information Hiding, Springer Berlin Heidelberg, pp. 36-52, 2003.

[4] C. Diaz, and A. Serjantov, "Generalising mixes," Privacy Enhancing Technologies, Springer Berlin Heidelberg, pp. 18-31, 2003.

[5] D. Kesdogan, J. Egner, and R. Büschkes, "Stop-and-go-mixes providing probabilistic anonymity in an open system," Information Hiding, Springer Berlin Heidelberg, pp. 83-98, 1998.

[6] C. Diaz, and A. Serjantov, "Generalising mixes," Privacy Enhancing Technologies, Springer Berlin Heidelberg, pp. 18-31, 2003.

[7] G. Danezis, and L. Sassaman, "Heartbeat traffic to counter (n-1) attacks: red-green-black mixes," Proceedings of ACM workshop on Privacy in the electronic society, pp. 89-93, 2003.

[8] “Tor Project, Inc. The Tor Project,” https://www.torproject.org.

[9] R. Dingledine, N. Mathewson, and P. Syverson, “Tor: The Second-Generation Onion Router,” In USENIX Security Symposium (USENIX), vol.13, pp.21-21, 2004.

[10] S. Hahn and K. Loesing, “Privacy-preserving Ways to Estimate the Number of Tor Users,” 2010. Available at https://metrics.torproject.org /papers/countingusers-2010-11-30.pdf.

[11] “The Tor Metrics Portal,” http://metrics.torproject.org.

[12] D. Foo Kune, T. Malchow, J. Tyra, N. Hopper, and Y. Kim, “The Distributed Virtual Network for High Fidelity Large Scale Peer to Peer Network Simulation,” Technical Report 10-029, University of Minnesota, 2010.

[13] M.Reed, P. Syverson, and D. Goldschlag, “Anonymous connections and onion routing,” IEEE Journal on Selected Areas in Communications, vol.16, no.4, pp.482–494, 1998.

[14] R. Dingledine and N. Mathewson, “Tor directory protocol, version 3,” https://gitweb.torproject.org /torspec.git /blob/ HEAD:/dir-spec.txt

[15] M. Perry, “Torflow: Tor network analysis,” Symposium on Privacy Enhancing Technologies (HotPETS), 2009.

[16] L. Øverlier and P. Syverson, “Locating Hidden Servers,” IEEE Symposium on Security and Privacy (Oakland), 2006.

[17] M. Wright, M. Adler, B. N. Levine, and C. Shields, “The Predecessor Attack: An Analysis of a Threat to Anonymous Communications Systems,” ACM Transactions on Information and System Security (TISSEC), vol.4, no.7, pp.489–522, 2004.

[18] R. Dingledine and N. Mathewson, “Tor protocol specification,” Available at https://gitweb.torproject.org/torspec.git?a=blob_plain; hb=HEAD;f=tor-spec.txt.

[19] Z. Ling, J. Luo, W. Yu, M. Yang, X. Fu, "Extensive analysis and large-scale empirical evaluation of tor bridge discovery," IEEE Proceedings of INFOCOM, pp.2381- 2389, 2012.

[20] Z. Ling, J. Luo, W. Yu, X. Fu, D. Xuan and W. Jia, "A New Cell-Counting-Based Attack Against Tor," IEEE/ACM Transactions on Networking, Vol.20, No.4, pp.1245-1261, 2012.

[21] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, "Lowresource routing attacks against tor," in Proceedings of the Workshop on Privacy in the Electronic Society (WPES), 2007.

[22] M. Edman and B. Yener, "On anonymity in an electronic society: A survey of anonymous communication systems," ACM Computing Surveys (CSUR), Vol. 42, No. 1, 2009.

[23] R. Jansen and N. Hopper, “Shadow: Running Tor in a Box for Accurate and Efficient Experimentation,” Symposium on Network and Distributed System Security (NDSS), 2012.

[24] ″The shadow simulator,” http://shadow.github.io/

[25] R. Dingledine and N. Mathewson, “Tor path specification,” Available at https://gitweb.torproject.org/torspec.git?a=blob_plain/HEAD:/path-spec.txt.

[26] M. AlSabah, K. Bauer, T. Elahi and I. Goldberg, "Tempura: Improved tor performance with multipath routing," Centre for Cryptographic Research, University of Waterloo, Tech. Rep, 2011.

[27] R. Snader and N. Borisov, "A Tune-up for Tor: Improving Security and Performance in the Tor Network," Network and Distributed System Security, Vol. 8, pp. 127, 2008.

[28] I. Goldberg, “Privacy-enhancing Technologies for the Internet, II: Five Years Later,” Springer Berlin Heidelberg, pp. 1-12, 2003. http://dx.doi.org/10.1007/3-540-36467-6_1

[29] I. Goldberg, “Privacy-Enhancing Technologies for the Internet III: Ten Years Later,” chapter 1 of Digital Privacy: Theory, Technologies, and Practices, pp. 3–18, 2007.

[30] I. Goldberg, D. Wagner and E. A. Brewer, “Privacy-enhancing Technologies for the Internet,” IEEE Proceedings of COMPCON, Vol. 103, No. 109, pp. 23-26, 1997.

[31] N. Borisov, I. Goldberg and E. Brewer, “Off-the-record communication, or, why not to use PGP,” Proceedings of the ACM workshop on Privacy in the electronic society, pp. 77-84, 2004. http://doi.acm.org/10.1145/1029179.1029200.

[32] “Adium,” A free instant messaging application for Mac OS X. http://www.adiumx.com/

[33] A. Serjantov and P. Sewell, "Passive attack analysis for connection-based anonymity systems," Computer Security–ESORICS, pp. 116-131, 2003.

[34] J. Boyan, "The Anonymizer-Protecting User Privacy on the Web," 1997.

[35] W. Dai, “Pipenet 1.0. Post to Cypherpunks mailing list,” 1998.

[36] M. Reiter and A. RUBIN, “Crowds: Anonymity for Web transactions,” ACM Transactions on Information and System Security, Vol. 1, No. 1, pp. 66–92., 1998.

[37] O. Berthold, H. Federrath, and S. Köpsell, "Web MIXes: A system for anonymous and unobservable Internet access," Designing Privacy Enhancing Technologies, pp. 115-129, 2001.

[38] M. J. Freedman, and R. Morris, "Tarzan: A peer-to-peer anonymizing network layer," Proceedings of the 9th ACM conference on Computer and communications security, pp. 193-206, 2002.

[39] M. Rennhard B. Plattner, "Introducing MorphMix: peer-to-peer based anonymous Internet usage with collusion detection," Proceedings of the ACM workshop on Privacy in the Electronic Society, pp. 91-102, 2002.

[40] S. Dahal, J. Lee, J. Kang and S. Shin, "Analysis on end-to-end node selection probability in Tor network," IEEE International Conference on Information Networking (ICOIN), pp. 46-50, 2015.

[41] S. Nepal, S. Dahal and S. Shin, “Deanonymizing schemes of hidden services in tor network: A survey,” IEEE International Conference on Information Networking (ICOIN), pp. 468-473, 2015.

[42] N. G. Duffield, A. G. Greenberg, P. Goyal, P. P. Mishra, R. K. Kadangode and J. E. Van der Merwe, "Virtual private network," U.S. Patent No. 6912232, 2005.

[43] E. Wustrow, S. Wolchok, I. Goldberg and J. A. Halderman, “Telex: Anticensorship in the Network Infrastructure,” Proceedings of 20th Usenix Security Symposium (Usenix Security), 2011.

[44] B. Zantout and R. Haraty, “I2P Data Communication System,” Proceedings of 10th International Conference in Networks, pp. 401-409, 2011.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my utmost gratitude to my advisor, Professor Seokjoo Shin for all his valuable guidance, advice and support during my years as graduate student at Chosun University.

I would like to show appreciation to the members of the thesis examining committee members Prof. Sangman Moh and Prof. Moonsoo Kang, for their comments and expertise feedback to complete my thesis. Likewise, I would like to thank the Department of Computer Engineering, Chosun University, to provide me an opportunity to pursue my Master's degree.

I was a great experience for me to work in Wireless Communication and Networking Lab (WHYNET) with other members. I would like to thank Mr. Seong Yong Jeon, who always helped me to deal the matters related to Korean language. Similarly, I would like to thank all the members of WHYNET Lab for their warm friendship and kind assistance.

I have been supported by my Nepali senior brothers and sisters during these two years. So I would like to thank them all.

Finally, I would like to thank my family for their unconditional support and caring throughout the degree. In particular, I want to thank my grandparents, parents, my sister and most importantly my wife Sabita, for all their love and support. Thank you all!!!!

Saurav Dahal