Cleveland State University
EngagedScholarship@CSU

ETD Archive

2012

A Coding Enabled Anonymity Network

Saikrishna Gumudavally
Cleveland State University

Follow this and additional works at: https://engagedscholarship.csuohio.edu/etdarchive

 Part of the Electrical and Computer Engineering Commons

How does access to this work benefit you? Let us know!

Recommended Citation
Gumudavally, Saikrishna, "A Coding Enabled Anonymity Network" (2012). *ETD Archive*. 804.
https://engagedscholarship.csuohio.edu/etdarchive/804

This Thesis is brought to you for free and open access by EngagedScholarship@CSU. It has been accepted for inclusion in ETD Archive by an authorized administrator of EngagedScholarship@CSU. For more information, please contact library.es@csuohio.edu.

A CODING ENABLED ANONYMITY NETWORK

SAIKRISHNA GUMUDAVALLY

Bachelor of Technology (B.Tech)

Electronics and Communication Engineering(E.C.E)

Jawaharlal Nehru Technological University,India

May, 2007

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

at the

CLEVELAND STATE UNIVERSITY

January 2012

This thesis has been approved for the

Department of **ELECTRICAL AND COMPUTER ENGINEERING**

and the College of Graduate Studies by

Thesis Committee Chairperson, Dr. Ye Zhu

Department/Date

Dr. Chansu Yu

Department/Date

Dr. Wenbing Zhao

Department/Date

To my Parents and Friends

ACKNOWLEDGMENTS

I would like to thank committee members Dr. Chansu Yu, Dr. Wenbing Zhao for their support. I am thankful to Dr. Zhu for giving me an opportunity to work in Network Security and Privacy research group.

I would like to thank my research group members for their support. Special thanks to my family and friends for their moral support.

A CODING ENABLED ANONYMITY NETWORK

SAIKRISHNA GUMUDAVALLY

ABSTRACT

An onion routing based anonymous communication system is developed to address timing analysis attacks, a common limitation of many contemporary anonymous systems including Tor. Timing analysis based attacks gained importance because simple payload check and packet inspection attacks are avoided by encrypting packets. Timing information gathered at one part of the network is correlated with information gathered at other parts to break the anonymity. Network coding, a recently developed packet forwarding technique, is used to disrupt timing attacks. The system uses a multicast tree of onion routers (OR) through which the packets are relayed to desired destinations. Packets from different users are grouped and linearly transformed over a finite field before forwarding them into the multicast tree. Encoding/transforming the packets evenly spreads the information among all encoded output packets making them equally important and informative. The system creates similar traffic pattern on all the links of the tree. Since the traffic pattern for all the ORs in the tree is similar, it becomes difficult to launch timing attacks. Extensive experiments are carried out for TCP communications using the Network Simulator-2 for different sizes of the multicast tree and probability of detecting a communication is equal to the probability detection through a random guess, equal to $1/n$, where n is number of ORs in last layer of the multicast tree. By increasing the number of leaf ORs in the tree decreases the detection probability and increase the degree of anonymity.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Most contemporary anonymous systems are built on the concept of onion routing. Onion routing has been widely accepted from the time that was initially introduced by Chaum [5]. In onion routing, first the data is repeatedly encrypted with the public keys of intermmediate nodes called the mixes then when the traffic is routed through these mixes, each mix decrypts a layer of encryption and delays the traffic before forwarding the data to the next hop. For applications which require lower latency for better QoS such as web browsing, anonymous sytems such as TOR are often used. Since oinon layer of encrytion hide the destination and payload, many flow based attacks are developed. These attacks use timing information gathered at different parts of the network to break anonymity. Contemporary systems are vulnerable to timing attacks where the adversary tries to relate/link the traffic by collecting packet timings or by inducing specific delay patterns or even counting the packets on both ends of the communication [10]. Even systems like Tor are vulnerable to such a strong adversary.

We use the novel concept of network coding to defend against timing attacks.

A node forwards a linear combination of the incoming packets on the outgoing links. In network coding, the packets are linearly transformed over a finite field [32]. In network coding routing scheme, the intermediate nodes in the route to the final destination have more importance and responsibilities when compared to the traditional store and forward routing schemes. The intermediate nodes not only buffer the incoming packets as usual but also combine/encode them with existing packets before forwarding. An encoded packet contains a trace of information from all the original packets used during the encoding process. A node has to solve the linearly encoded data to get actual information. It has been applied to applications such as P2P [6,12], wireless network [17,31] and unicast communication [34]. Network coding is more robust to link failures and node failures, because the information is evenly spread throughout the encoded packets making them equally important [12]. Figure 1 is an example of network coding. The incoming packets denoted by X, Y and Z are linearly encoded to produce output O_1, O_2 and O_3. The set of coefficients used for linear transformations is called encoding vector. For output O_1, the encoding vector is [2 3 2].

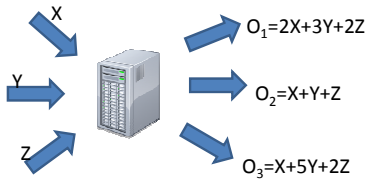


Figure 1: Network Coding Example

The destination has to collect sufficient encoded packets in order to begin decoding. For pratical implementation of this scheme, the original data packets are batched into groups called generations, and only packets belonging to same generation are encoded and decoded together [33]. In the proposed system, packets from different users are combined and transmitted over a multicast tree containing layers of onion routers (OR). The root OR of the multicast tree groups the available packets into

generations and forms the random linear combinations. The encoded packets are repeatedly encrypted by the root and then transferred from one layer to another in the multicast tree. On receiving a packet, an intermediate OR decrypts a layer of encryption and randomly selects an OR in the next layer to forward the packet. An encoded packet contains a trace of information from all the original packets, so it cannot be linked to any particular user, and all the ORs in the tree receive similar amount of traffic making it difficult for an adversary to launch timing attacks. The users run a program called onion proxy (OP). An OP helps in tree contruction and relaying data.

The rest of the paper is organized as follows. In Chapter 2 gives a review on the related works to the proposed system. In Chapter 3 goals are summurized. Chapter 4 describes the threat model and assumptions made. Chapter 5 deals with the design and implementation of the system. Theoretical analysis of performance evaluation parameters are explained in 6. Performance of the proposed system is evaluated in Chapter 7 and conclude with Chapter 9.

CHAPTER II

RELATED WORK

2.1 Anonymous Communication Systems

Chaum [5] initially introduced the idea of using public key cryptography to hide the source and destination of a communication. The messages are repeatedly encrypted with the public keys of the mixes in the path to the destination. When the encrypted messages are routed through these mixes, each mix will decrypts a layer of encryption and introduce a random delay before relaying to the next hop. Any intermediate node (Mix) has no knowledge about the complete route except the previous and next hop. Since its introduction many anonymous systems have been proposed that use the concept of onion routing. Tor [7], the second generation onion router, is a low latency anonymous system which incrementaly builds circuits along the network and the traffic is routed through these circuits. Tor provides perfect forward secrecy by using TLS encrytion and also provides additional features such as congestion control and integrity checking. Anonymous systems like Maxminion [10] and Mixmaster [29] are built on the concept of onion routing, but provide anonymity using

timed dynamic pool batching which introduce large latencies to disrupt timing analysis attacks. Such large delays are not viable for some applications like web browsing where delay decreases the QoS. Tarzan [9] and Morphmix [23] provide anonymous communication for P2P overlays by trying to hide the origin of the traffic. Every node acts as a relay as well as generates its own traffic. Both these methods are based on onion type layered encryption. The concept of crowds introduced in [22], concentrates on hiding the origin of the traffic. Each member of the crowd is equally probable to be the initiator of the traffic. The larger the crowd, the greater the degree of anonymity. In crowds, more control is needed over the browser in order to provide a higher degree of anonymity. Anonymizer [1] and JAP [3] provide anonymity by aggregating the traffic from different users. Anonymizer uses a single hop proxy where as JAP uses a distributed trust system. JAP builds fixed circuits involving series of mixes called the cascade, through which traffic is relayed. In [16], a method of anonymous communication has been proposed which makes use of network coding technique to scramble the message and transfer the pieces of the message along disjoint paths. Here the anonymity and the confidentially provided are slightly weaker than in systems that use public key cryptography.

2.2 Possible Attacks on Current Systems

Since the development of the onion routing technique which avoids simple payload comparison attacks, many flow based anonymity attacks have been proposed. These attacks make use of timing information to relate the traffic flows. An adversary who is observing the traffic, gathers the timing information and uses various techniques to relate the traces obtained [24]. A more powerful adversary can induce distinct delays into the traffic and observes traffic pattern at other parts of the network for such distinct patterns [7]. In [35] and [36], flow separation methods are

proposed which make use of the Blind Source Separation algorithm which separates the flows from the mixed traffic of a mixer into individual flows. Once the flows are separated various techniques can be applied to relate and detect the traffic.

2.3 Network Coding Techniques

We use the technique of network coding to overcome the above mentioned possible attacks. Network coding provides an opportunity to fully utilize network capacity and has the potential to increase the throughput while being robust to lossy links [19] and node failures. In network coding, the intermediate nodes not only buffer the incoming packets as usual but also combine/encode them with existing packets before forwarding. Network coding requires nodes to have high computational power to perform the tasks like encoding and decoding [8]. In a network, the broadcast capacity is limited by the minimum cut between the sender and the receiver. This maximum capacity is not attainable by traditional forwarding schemes however with network coding this maximum can be reached [2]. In [32], a network coding algorithm is developed for time varying cyclic networks based on the max-flow min-cut theorem produces minimum possible delay while achieving broadcast capacity. Information is considered as a vector over a finite field on which each node applies a linear transformation before forwarding and it has been proven that for the linear transformation a finite symbol size is sufficient to have linear coding for multicast purpose [32]. A framework is developed in [18] to apply network coding to arbitrary networks to achieve network capacity. Network capacity can be achieved with high probability by randomly selecting the linear mapping of inputs to form the output [15, 18, 32]. The upper bound on the failure to decode is in the order of the inverse of the finite field size of random variables used for coding. In conclusion, using a larger field size for encoding significantly reduces the probability of failure [14].

"Avalanche" is a network coding mechanism for large scale content distribution [12]. It solves almost all of the issues in the current file swarming mechanisms such as rare blocks, load on servers and eliminates the need to increase network resources in order to provide better file sharing services. This system takes advantage of the uniform distribution of information among all the encoded packets to eliminate the rare blocks problem. In P2P networks, network coding helps in faster file transfer with little CPU overhead and is able to provide better service to all the peers in a topology including the the peers that are behind NAT and firewalls [11]. A practical implementation of the network coding technique is provided in [33]. Here the packets are grouped into generations and only the packets from the same generation can be encoded and decoded together. By buffering the packets, it deals with random link loss and synchronization problems caused by congestion delay. Attaching the encoding vector along with the block sent out makes it robust and helps in extraction of the original blocks. Encoding vecor is a series of coefficients used during encoding process. Network coding by default provides some light weight security because the adversary needs to obtain sufficient encoded packets before retrieving actual data which in some cases might be difficult for an adversary who has limited access and control over the network. Confidentiality can be provided by simply encrypting the encoding vector to counter malicious modifications done by the intermediate nodes [30]. Secure network coding methods have been proposed in [4] and [28], to tackle wiretappers while providing perfect forward secrecy.

CHAPTER III

GOALS

As described in Section 2, many timing-based anonymity attacks have been proposed to compromise existing anonymity networks. Clearly, a novel network architecture for anonymous communication is needed to defend against these attacks. In addition to the goal of providing improved anonymity, the new network architecture should satisfy the following requirements: (1) It must provide sender anonymity and receiver anonymity [21]. (2) It is capable of providing low-latency communication services. Because of the popularity of Internet applications which require low latency such as web browsing and VoIP, low latency is a must for usable anonymity networks. (3) Simple and proven anonymizing techniques should be included in the new anonymity network architecture to avoid previous pitfalls. In this paper, the onion-based layered encryption previously implemented and verified in Onion Routing [13, 26, 27] and Tor [7] are extended for the coding-enabled anonymity network architecture,[1] since the onion-based layered encryption can hide source and destination address information from intermediate nodes to protect anonymity even when

[1]We believe the new architecture without the onion-based layered encryption is possible. We plan to investigate in our future work.

some intermediate nodes are compromised.

We do not include defense against end-to-end timing-based traffic analysis as one of our design goals. End-to-end timing-based traffic analysis, such as correlating traffic flows entering and leaving an anonymity network, can effectively link communication parties. These end-to-end traffic analysis attacks are not considered as in previous studies [7], because these attacks are based on traffic flow information collected outside an anonymity network. In Section 8, we introduce a possible countermeasure for these end-to-end traffic analysis attacks.

In this paper, we do not base the new anonymity network architecture on peer-to-peer network structure such as Tarzan [9] because of known problems of peer-to-peer structure used for anonymity [9, 23].

CHAPTER IV

THREAT MODEL

4.1 Capability of Adversary

The main goal of the system is to provide anonymous communication for both source and the destination pair. It is assumed that the adversary has control over some parts of the network. The adversary here has access to some parts of network and can observe and modify the traffic on all the links in those parts. All the links between OR's are TLS encrypted which provides perfect forward secrecy, thus any modifications to the data transferred can be detected. The adversary can launch traffic analysis attacks by correlating the timing information of the packets. The adversary can match the traffic patterns by causing distinct delays or disrupting the traffic [7]. The adversary has the capacity to build the traffic pattern of the packets emerging from a particular user and try to match with the traffic patterns collected from other parts of the network, thus identifying the communication. The adversary might also run an OR that is corrupted or malicious. The attacker can launch Denial of Service attack on trustworthy OR's and divert the traffic to a group

of OR controlled by it [7]. The malicious OR can also give false information like available bandwidth to the directory servers so that it is included in the multicast tree. By doing so, it can launch the attacks where it drops the packets and not forward as instructed and to see where in the network the traffic halts or packets are sent to a group controlled by the adversary when the key has been compromised. Here a faulty OR can disrupt only the traffic that pass through it and cannot effect traffic on other links. By performing the above mention activities the adversary can only disrupt the traffic but cannot decrease the anonymity of the system.

4.2 Assumptions

Not all nodes can be compromised at the same time and the adversary cannot have access to all parts of the network thus end to end timing analysis cannot be performed. All the packets are end to end encrypted for higer secrecy is needed. It is also assumed that all the OR's provide correct information about their public keys as they are used during the tree construction phase. An OR in the tree has only knowledge of previous layer ORs and next layer ORs. The root OR has access to multiple IP. This is a reasonable assumption considering the popularity of VPN [16].

CHAPTER V

DESIGN

Onion routing is the most commonly used technique in anonymous communication. The proposed system is built on the widely accepted low latency anonymous system called Tor [7]. The proposed packet forwarding system combines the concept of onion routing with the recently developed network coding technique to avoid timing analysis attacks, while inheriting the onion routing immunity towards many other attacks.

5.1 Design Overview

Users run a program called the Onion Proxy which helps in communicating with the onion routers. A group of Onion Proxies/users (OPs) come together to build a multicast tree of ORs. The multicast tree is used by all the OPs involved in tree construction to anonymize their traffic. The multicast tree contains layers of onion routers. The multicast tree of ORs is constructed layer by layer and each OP chooses an OR of choice to be included in a layer. An OR in a layer is connected to every

OR in the next layer. The leaf ORs of the tree are called the exit ORs and the last layer is called the exit layer. OPs connect to the tree through the root of the tree.

After the multicast tree construction, packets from different users (OPs) are collected and encoded by the root of the tree before forwarding them in the tree. The group of packets that are encoded together is given a name denoted by *Generation Sequence Number*. The number of packets in this group is given by the *Generation Size* gs. The root OR in Figure 2 encodes the packets from user A and user B. The

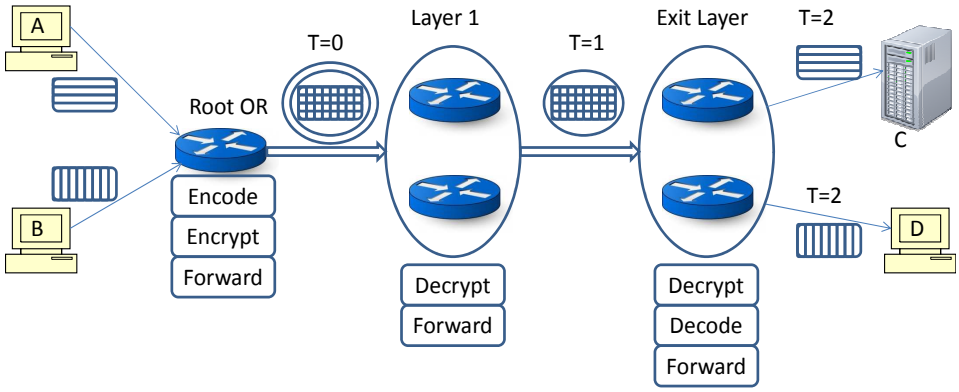


Figure 2: Overview

root repeatedly encrypts the encoded packets with the keys it shares with the ORs in each layer of the multicast tree. The keys used by the root OR for layer encryption are formed and shared during the tree construction process. The hashed box represents the encoded packet and circles around it represent layer encryption. The root forwards the encoded and encrypted packets to first layer of the multicast tree. Packets are forwarded from one layer to the next. When an OR receives a packet, it decrypt a layer and then select an OR in the next layer to send the packet. A layer of encryption is decrypted at each layer of the multicast tree. Last layer of encryption is decrypted by exit ORs. Since all layers of encryption are now removed, the exit ORs gets the encoded packets. The exit ORs then decode the encoded packets that belong to the same generation. The decoding process starts only if exit OR has received sufficient encoded packets of that generation or gs encoded packets. The decoded packets are

forwarded to actual destination by the exit ORs, here server C and server D. It can be seen that all the exit ORs receive the packets at the similar time and also receive similar amount of traffic. Thus making it difficult for an adversery to link an encoded packet to a either user A or user B. Depending on the context, OP and the root OR use different types of packets for the tree construction. The packet format is a modified version of the packet format used in Tor. Additional header fields are added to accommodate the needs of encoding and decoding involved in network coding. The tree construction is based on Tor circuit formation. Like in Tor each onion router maintains an identity key for signing the TLS certificates and other onion keys to decrypt packets from the root OR of the tree.

5.2 Packet Format

The ORs and OPs communicate with each other over TLS links using packets of length determined by the OPs. There are two types of packets used in this scheme of data forwarding. They are control packets and relay packets.

5.2.1 Control Packet

A control packet carries different types of commands. When an OR receives a control packet it would act according to the command in the packet. The packet format for the control packet of length 512 bytes is shown in the Figure 3 and explanantion of each field follows.

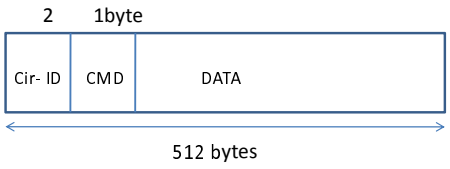


Figure 3: Control Packet

Circuit identifier is a local value; it changes from one hop to another. It is the mutually agreed name given to the link between the sender and the receiver of this packet. The *CMD* field indicates the command a packet carries. The commands in a control packet are *create, created, group, grouped, change* and *changed*. A *create* packet is sent out to build a circuit between the sending OR and the receiver of the packet. A *created* packet is used as an acknowledgment for a *create* packet. The receiver of a *group* packet recognizes the sender as an OR in the previous layer of the tree. A *grouped* packet is used as an acknowledgment for a received *group* packet. A *change* packet is used to transfer keys from the root to a newly formed layer of ORs. A *changed* packet is used as an acknowledgement for a received *change* packet.

5.2.2 Relay Packet

The relay packets are used to transfer end to end information [7]. The relay packets have additional header fields when compared to the control packets. The format is shown in Figure 4.

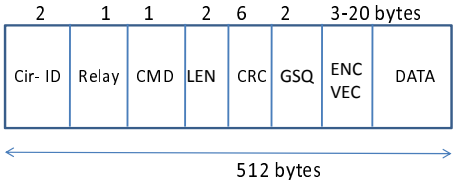


Figure 4: Relay Packet

Circuit identifier field serves the same purpose as mentioned before. The relay header field *length* denotes the length of the packet, *CRC* is a checksum used to check packet integrity and *CMD* byte is used to indicate the type of the relay command. The relay commands are *extend* (used to extend an existing connection), *extended* (sent out when an extension is succesfully completed), *keychange* (used to broadcast the layer key for a particular layer), *keychanged* (reply to keychange packet sent after successful transfer of the layer key). *GSQ* denotes the *Generation Sequence Number*

Table I: Key Notation

Notation	Owner	Purpose
PK_i	OR_i	Public key of node i. It is used to send encrypt packets to OR_i. Exampe: PK_{11} denotes public key of the onion router OR_{11}
PK_i^T	OR_i	Public Key of OR_i, used for tree construction and key sharing. T denotes that the key is specific to this tree. Also referred to as tree public key of OR_i.
$PK_{S_{Li}}^T$	Root (S)	Public Key of the root OR used for key sharing with the i^{th} layer of the tree. This key is used by the root to transfer layer key to the i^{th} layer of the multicast tree. Also referred to as tree public key of S for i^{th} layer. Li denotes i^{th} layer.
K_{Li}^S	Root (S)	Symmetric key S shares with all the ORs in the i^{th} layer. Used for layer encryption by the root OR. Also referred to as the root layer key for the i^{th} layer.
K_{Li}^{OP}	Onion Proxy (OP)	Symmetric key all the OPs shares with all the OR in the i^{th} layer. Used to encrypt information to be sent to the i^{th} layer of the tree from the OPs. Also referred to as OP layer key for i^{th} layer.
$K_{OP_i}^S$	OP and S	Symmetric key shared between the i^{th} OP denoted by OP_i and the root S.
$PK_{E_i}^T$	Exit OR of OR_i	Public key of the exit OR on the tree whose root OR is OR_i

to which the packet belongs and *encoding vector* is the series of linear coefficients that were used during the encoding process by the root to generate that packet. The relay cells are iteratively encrypted by the root with the keys it shares with the ORs in the intermediate layers of the tree.

The OPs, the root and the intermmediate ORs communicate with each other using the relay and the control packets. Different keys are used to encrypt these packets depending upon the source, destination and other parties invloved in the communication. Various keys used and their notations are given in the Table I.

5.3 Tree Construction

The proposed system makes use of a multicast tree of onion routers to anonymize traffic. When an OP wants to anonymize its communication, it connects to the root of the multicast tree and multiple OPs connect to the root. The root of the multicast tree will combine/encode the packets from different OPs and forward them in the tree. The root of the tree is also responsible for the layer encryption of the encoded packets. The root repeatedly encrypts the encoded packets with the symmetric keys

shared with the ORs in each layer of the tree.

To begin the multicast tree construction, each OP negotiates a symmetric key with the root. This key is used to encrypt the packets exchanged between the root and the OP.

The tree is constructed incrementally, one layer at a time. The OPs decide the ORs that are to be included in a layer of the tree. After connecting to the root, each OP chooses an OR that will be in the 1^{st} layer of the tree. Each OP sends out a relay *extend* packet to the root, instructing it to include the chosen OR in the first layer. The root forms a set of control packets for the chosen ORs to make them part of the tree. These control packets contain information about the tree and keys used for encryption. After forming the 1^{st} layer, the OPs will send relay *extend* packets to the 1^{st} layer ORs instructing them to extend the tree to the next layer. The OPs follow a similar procedure of instructing the last layer to extend the tree to a new layer until a tree of required size is formed. The root of the tree shares a symmetric key with each layer of the tree. Whenever the ith layer is added to the tree, the root S forms the root layer key K_{Li}^{S} and shares it with the ORs in the newly formed ith layer, denoted by Li. The encoded packets are layer encrypted by the the root using these shared keys.

5.3.1 Establishing Connections Between the OPs and the Root

To build a muticast tree with the OR S as the root, all the OPs need to establish a connection with the OR S. An OP sends a *create* packet to establish a connection and to negotiate a symmetric key with the root. This key is used by an OP to encrypt any packet sent to the root. This key is denoted by $K_{S}^{OP_i}$, shared between an Onion Proxy OP$_i$ and the root S.

The packets exchanged between the root S and an onion proxy OP_i during this phase are depicted in Figure 5. To create a new connection, OP_i sends a *create* packet along with the first half of Diffie-Hellman handshake P^x in the payload to the root S. The root S responds with a *created* packet. The payload contains the second half of Diffie-Hellman key exchange P^y and the hash of the key formed. After the exchange of *create* and *created* packets, OP_i and the root S have both the halves of key exchange algorithm, so they can form $K_S^{OP_i}$. In Figure 5, the packets are numbered according to the order of events.

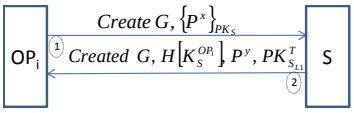


Figure 5: Creating Circuits

G is the circuit identifier. Circuit identifier is a unique name given to the link between two ORs or an OR and an OP. The payload of the *created* packet also contains $PK_{S_{L1}}^T$, the tree public key of the root S used for the key transfer from the 1^{st} layer ORs denoted by $L1$ to the root. $PK_{S_{L1}}^T$ will be sent by the OPs to the ORs that will be included in the 1^{st} layer of the tree. H is the hash function. All the OPs exchange *create* and *created* packets with the root to form the symmetric key.

5.3.2 Extending the Tree to Layer 1

After connecting with the root, the OPs extend the tree to the 1^{st} layer by sending out relay *extend* packets to the root. Each OP chooses an OR to be a part of the 1^{st} layer, so the number of ORs per layer n in the multicast tree is equal to the number of OPs connected to the root. Each OP sends to the root an *extend* packet which contains the identity of the chosen OR and the keys used for encrypting future packets from the OPs destined for the 1^{st} layer ORs. The root copies the payload of an *extend* packet into a *group* packet. The *group* packets are forwarded to all the

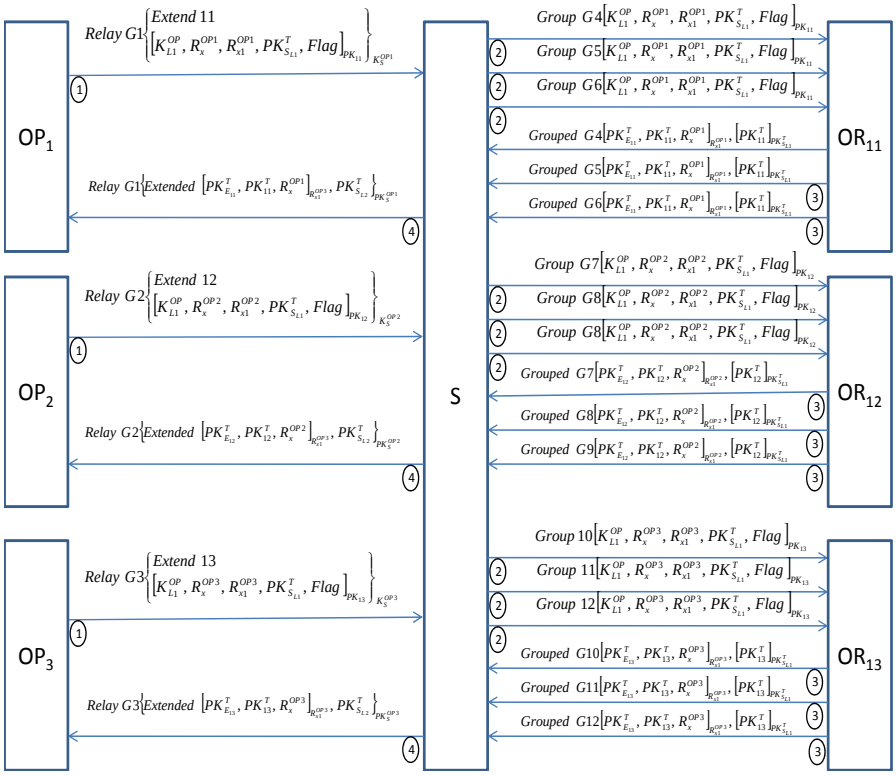


Figure 6: Multicast Tree Extending to Layer 1

ORs mentioned in the *extend* packets. An OR responds to a *group* packet with a *grouped* packet. A *grouped* packet contains confirmation of receipt by the chosen OR in the payload. A *grouped* packet is converted to a relay *extended* packet by the root and forwarded to the OPs.

For simplicity of explanation, as an example, consider the *extend* packet sent by OP_1 in the Figure 6. It is

$$Relay\ G1\left\{Extend\ OR_{11}\ \ \left[K_{L1}^{OP},\ R_1^{OP_1},\ R_2^{OP_1},\ PK_{S_{L1}}^{T},\ Flag\right]_{PK11}\right\}_{K_S^{OP_1}}.$$

$G1$ is the circuit identifier. As mentioned, $K_S^{OP_i}$ is used to encrypt any packet exchanged between OP_i and the root S. Here the *extend* packet is encrypted with $K_S^{OP_1}$. *Extend* OR_{11} indicates that OP_1 wants the onion router OR_{11} to be included in 1^{st} layer of the tree. The payload is encrypted with the public key PK_{11} of the onion router OR_{11}, so that only the onion router OR_{11} will be able to retrive the information. K_{L1}^{OP} is a symmetric key created by the OPs to encrypt all of the future

packets destined to the 1^{st} layer ORs denoted $L1$. All the OPs use the same OP layer key K_{L1}^{OP} in their *extend* packets for the 1^{st} layer. To form K_{L1}^{OP}, all the OPs exchange a series of messages that contain contributions from each OP towards the formation of K_{L1}^{OP} [25]. We assume these contributions are exchanged between the OPs before the formation of the *extend* packets. $R_1^{OP_1}$ and $R_2^{OP_1}$ are a pair of random numbers generated by OP_1. $R_1^{OP_1}$ will be a part of the reply as a confirmation of receipt by the intended OR and $R_2^{OP_1}$ will be used to encrypt the payload of the reply packet. $PK_{S_{L1}}^{T}$, the tree public key of the root is used to transfer keys between the root S and the 1^{st} layer ORs. The $flag$ value can either be 0 or 1 depending on whether OR_{11} is the intermmediate OR or an exit OR.

The root S copies the payload of the *extend* packets into the *group* packets. For the 1^{st} layer construction, each *extend* packet is converted to n *group* packets. These n *group* packets are forwarded by the root S to the chosen OR using different IP addresses. Each of these n *group* packets sent to an OR contain different circuit identifiers but the same payload. Same set of IP addresses are used to send *group* packets to each of the ORs mentioned in the *extend* packets. We make use of multiple IP addresses so that the root would appear like any other layer in the multicast tree. The number of multiple IP addresses used by the root to connect to each OR in the 1^{st} layer is equal to number of ORs per layer n. The ORs receiving the *group* packets consider the senders as the previous layer ORs. In the example given, the root S has multiple connections with the onion routers OR_{11}, OR_{12} and OR_{13}. There are three connections between the root S and each of the onion routers OR_{11}, OR_{12} and OR_{13}, giving a total of nine connections between the first layer and the root S.

An OR replies to a *group* packet with a *grouped* packet. Depending upon the flag and the keys received in the *group* packets, OR_{11} will form the payload of the *grouped* packets. The payload of the *grouped* packet sent out by OR_{11} contains

$\left[PK^T_{E_{11}},\ PK^T_{11},\ R^{OP_1}_1\right]_{R^{OP_1}_2}$, $\left[PK^T_{11}\right]_{PK^T_{S_{L1}}}$

Random number $R^{OP_1}_1$ is obtained from the *group* packets received by OR_{11}. Including $R^{OP_1}_1$ in the reply packet serves as a confirmation that the tree was extended to OR_{11}. If the flag value in the received *group* packets is set then $PK^T_{E_{11}}$ represents the public key of the exit OR in the tree for which the onion router OR_{11} is the root, E_{11} denotes the exit OR in the tree for which the onion router OR_{11} is the root. If the flag value is zero then $PK^T_{E_{11}}$ will be replaced by padding bits. $PK^T_{E_{11}}$ is needed by the OP to form the return address. Detailed explanation about the use of $PK^T_{E_{11}}$ is presented in section 5-C. PK^T_{11}, the tree public key of the onion router OR_{11}, is included twice in the *grouped* packet because only the root and OP_1 need to know this information. PK^T_{11} is used by the root to encrypt the root layer key shared between the root S and layer 1 ORs during the key exchange process. PK^T_{11} is used by OP_1 to encrypt any data sent to the onion router OR_{11}. Since the root has no information about the random number $R^{OP_1}_2$ used to encrypt the reply, PK^T_{11} has to be encrypted seperately using $PK^T_{S_{L1}}$ so that the root will be able to retrive it. If PK^T_{11} is made public then other ORs which select the OR S as their exit OR can relate the exit OR public key given to them with this information.

When the *grouped* packets are received by the root S it retrives tree public keys (example PK^T_{11}) from the payload and copies the rest into an *extended* packet and forwards it to the OPs. The *extended* packet is broadcasted to all the OPs but only OP_1 will be able to verify the *extended* packet. The *extended* packets are broadcasted because the root cannot relate an *extend* packet to the corresponding *extended* packet. Since OP_1 knows $R^{OP_1}_2$, it can decrypt the *extended* packet for confirmation and verification that the tree was extended to the onion router OR_{11}. The root S will also include $PK^T_{S_{L2}}$, the root public key for layer 2 denoted by L2, in the payload of the *extended* packet, so that this key can be included in the *extend*

packets for layer 2.

If an OP does not receive any reply packet for the *extend* packet sent out, then it would send an *Unsuccessfull* packet to the root. The root broadcasts a *resend* packet to all of the OPs. A *resend* packet indicates to the OPs that the tree extension could not be completed successfully.

5.3.3 Transferring the Root Layer Key for Layer 1

The root shares a unique symmetric key with each layer of the multicast tree. These keys are used by the root to form the onion layers of encryption. After the 1^{st} layer of the tree is formed, the root S forms a root layer key K_{L1}^{S} for layer 1 ORs denoted by L1. This layer key is forwarded to all of the ORs in the 1^{st} layer. To transfer the root layer key to layer 1 ORs, the root S forms a *change* packet. The payload contains blocks of the root layer key and a random number. Each block contains the root layer key and a random number. Each block is encrypted with the public key of an OR in the 1^{st} layer. The root OR gets the public keys of the ORs from the *grouped* packets. Figure 7 shows the packets exchanged with the onion router OR_{11} during the key transfer process. The onion router OR_{11} replies a *change* packet with a *changed* packet. $\left\{K_{L1}^{S}, R_{a}^{S}\right\}_{PK_{11}^{T}}$ is an encrypted block formed by the root for OR_{11} and can only be decrypted by the onion router OR_{11}. R_{a}^{S} is a the random number chosen by the root S. $\left[R_{a}^{S}\right]_{K_{L1}^{S}}$ can only be formed by OR_{11} thus serving as a confirmation of receipt. The payload of the *changed* packet sent by OR_{11} contains $\left[R_{a}^{S}\right]_{K_{L1}^{S}}$ confirming that layer key is received by the intended OR.

K_{L1}^{S} is shared between all of the ORs in the 1^{st} layer and the root S. All future packets from the root to the 1^{st} layer ORs are encrypted using this key and vice versa.

The root S exchanges similar packets with all the ORs in 1^{st} layer of the tree.

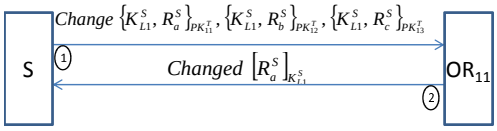


Figure 7: Packet Exchanged during Key Transfer

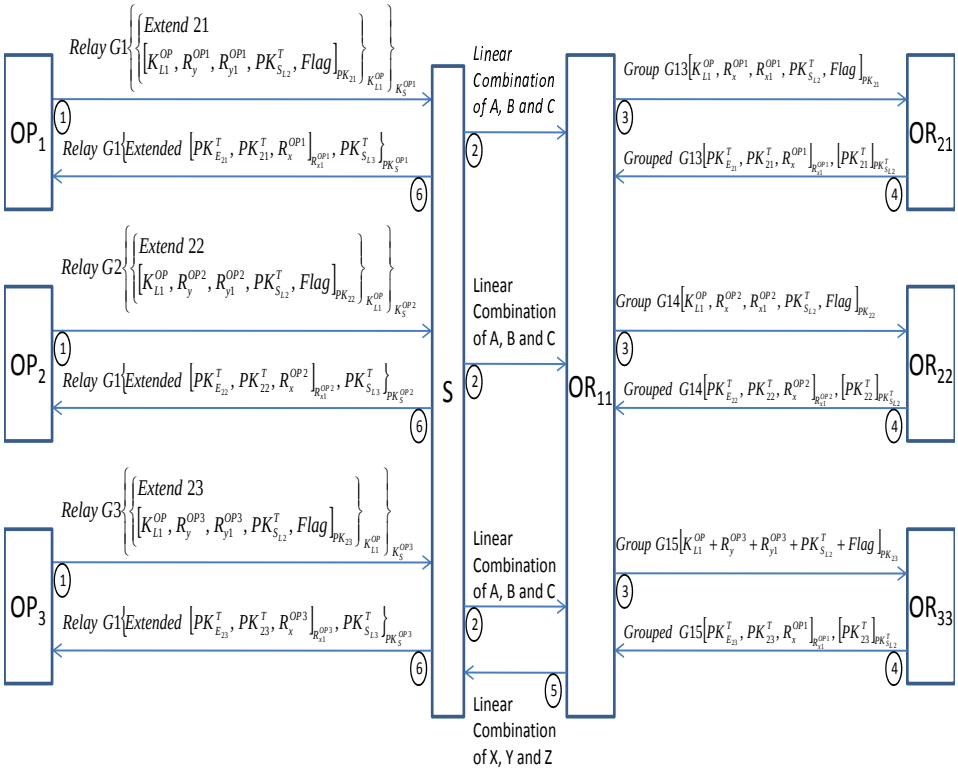


Figure 8: Multicast Tree Extending to New Layer

5.3.4 Extending the Tree to the 2^{nd} Layer

As mentioned earlier, the OPs send the *extend* packets to the last layer for extending the tree to a new layer. To build the 2^{nd} layer, the OPs send *extend* packets to the 1^{st} layer ORs. An onion proxy OP_i encrypts the *extend* packet with K_{L1}^{OP}, the key OPs share with the layer 1 ORs and with $K_S^{OP_i}$, the key it shares with the root. The OPs encrypt the *extend* packets with K_{L1}^{OP} so that the root will not be able to view its contents. The packets exchanged during this process are depicted in Figure 8 and Figure 9.

The root S will decrypt the packets it receives from the OPs with the respective

shared keys. In the example given, the decrypted packets are denoted by A, B and C

$$A = \left\{ Extend\ 21 \left[K_{L2}^{OP},\ R_3^{OP_1},\ R_4^{OP_1},\ PK_{S_{L2}}^{T},\ Flag \right]_{PK_{21}} \right\}_{K_{L1}^{OP}}$$

$$B = \left\{ Extend\ 22 \left[K_{L2}^{OP},\ R_3^{OP_2},\ R_4^{OP_2},\ PK_{S_{L2}}^{T},\ Flag \right]_{PK_{22}} \right\}_{K_{L1}^{OP}}$$

$$C = \left\{ Extend\ 23 \left[K_{L2}^{OP},\ R_3^{OP_3},\ R_4^{OP_3},\ PK_{S_{L2}}^{T},\ Flag \right]_{PK_{22}} \right\}_{K_{L1}^{OP}}$$

The root S will form the linear combinations of A, B and C. The encoded packets are layer encrypted with the layer keys which the root shares with each layer and in this case its K_{L1}^{S}. When the onion routers OR_{11}, OR_{12} and OR_{13} get three such coded packets, they can decrypt the layer encryption by the root S and then decode to get A, B and C. The ORs in the 1^{st} layer know K_{L1}^{OP}, so they can decrypt the encryption done by the OPs, to get *extend* packets. As shown in Figure 8, the onion router OR_{11} converts the *extend* packets into *group* packets and forwards them to the ORs mentioned in the *extend* packets. In the example, the onion router OR_{11} will send out the *group* packets to onion routers OR_{21}, OR_{22} and OR_{23}. OR_{12} and OR_{13} will do the same as OR_{11}. Each of these 2^{nd} layer ORs gets the required keys and other information from the *group* packets and reply with the *grouped* packets along with a confirmation of receipt in the payload. The structure of a *grouped* packet is discussed in chapter 5.3.2 and the detailed contents of the packets exchanged can be viewed in Figure 8.

When the onion router OR_{11} receives a reply from all of the 2^{nd} layer ORs, it converts the *grouped* packets to *extended* packets. As shown in Figure 9 each OR in the 1^{st} layer receives the *grouped* packets with the same payload from the onion router OR_{21}. Since the contents in the payload of the *group* packets received by the onion router OR_{21} are the same, the reply sent to the onion routers OR_{11}, OR_{12} and OR_{13} will also be the same. In other words, a *grouped* packet from the onion router OR_{21} to the onion router OR_{11} will have the same payload as a *grouped* packet from the onion router OR_{21} to the onion router OR_{12}. So all of the ORs in layer 1 will

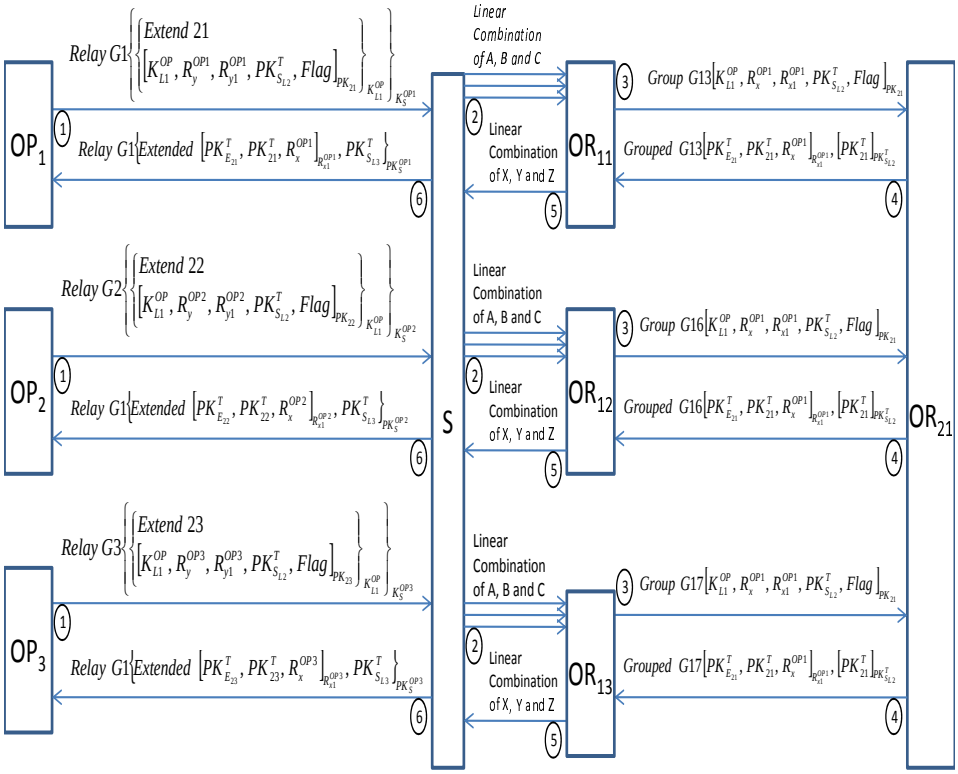


Figure 9: Multicast Tree Extending to New Layer

form identical *extended* packets. These packets are denoted as X, Y and Z.

$$X = Extended\left[PK^T_{E_{21}} + PK^T_{21} + R^{OP_1}_1\right]_{R^{OP_1}_{x1}} + \left[PK^T_{21}\right]_{PK^T_{S_{L2}}}$$

$$Y = Extended\left[PK^T_{E_{22}} + PK^T_{22} + R^{OP_2}_1\right]_{R^{OP_2}_{x1}} + \left[PK^T_{22}\right]_{PK^T_{S_{L2}}}$$

$$Z = Extended\left[PK^T_{E_{23}} + PK^T_{23} + R^{OP_3}_1\right]_{R^{OP_3}_{x1}} + \left[PK^T_{23}\right]_{PK^T_{S_{L2}}}$$

All of the ORs in layer 1 form a linear combination of X, Y and Z. The encoded packet is encrypted with K^S_{L1} before relaying back to the root on one of the links connected to previous layer. The root will receive 3 such encoded packets, one from each OR in the 1^{st} layer. The root will decrypt and decode the packets to get the *extended* packets. The *extended* packets are relayed back to the OPs as mentioned before. This is the only instance when an intermediate OR will encode data. Encoding helps in reducing the number of replies sent back to the root OR.

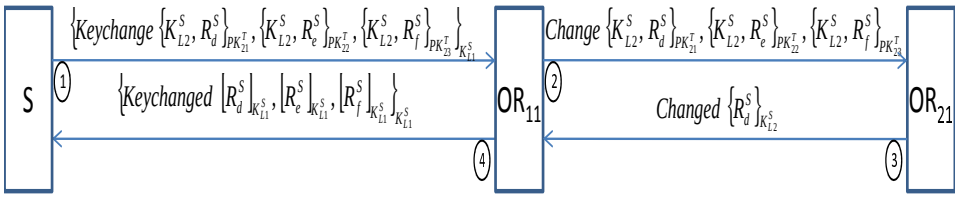


Figure 10: Transferring the root Layer Key to Layer 2

5.3.5 Transferring the Root Layer Key to the 2^{nd} Layer ORs

As explained earlier, after extending to the ith layer in the multicast tree, the root creates the root layer key K^S_{Li} for the ith layer denoted by Li. This layer key is shared with all the ORs in the ith layer. The root uses this key for layer encryption. The root will use the tree public key of each OR in the ith layer (example PK^T_{21}) to encrypt K^S_{Li} during key transfer. To transfer the key, the root forms a *keychange* packet. The payload of a *keychange* packet contains blocks of the new root layer key and a random number.

As shown in Figure 10, the payload of the *keychange* packet contains blocks of the root layer key for the 2^{nd} layer K^S_{L2} and a random number. Each block is encrypted with the tree public key of an OR in the 2^{nd} layer. The *keychange* packet is forwarded to a randomly chosen first layer OR, in example given it is OR_{11}. The onion router OR_{11} copies the payload of the *keychange* packet to a *change* packet and forwards it to all of the ORs in the 2^{nd} layer. The onion router OR_{21} replies with a *changed* packet and includes receipt confirmation in the payload. The onion router OR_{11} will receive similar confirmations from all of the ORs in the 2^{nd} layer. A *keychanged* packet is relayed back to the root after all of the confirmations are received. The payload of the *keychanged* packet contains all the confirmations received from all the 2^{nd} layer ORs.

To extend the tree to a new layer, the OPs instruct the last layer ORs to extend the tree to another layer. The root transfers the root layer key for the new

layer after the tree extension. The number of layers in the tree are decided by the OPs. The root encodes and encrypts the packets that it receives from the OPs and when an intermmediate OR receives a packet, it decrypts a layer of encryption and forwards it on a randomly picked next layer OR. The packets are forwarded from one layer to another layer and finally reach the exit layer. When an exit OR receives gs packets of a particular generation, it can decode to get the actual data. After it obtains the decoded information, an exit OR forwards the data to the destination. The path taken by reply data is discussed in chapter 5.4. In case there is only a single OP connected to the root OR, then there would be only one OR per layer. Packets cannot be encoded because there is only one source available. However the packets will be encrypted and relayed as before. The exit OR in the forward direction would still be the root in the reverse direction.

5.4 Return Path

The reply data from a destination is routed back to the OP using a different tree. The exit OR in the forward direction of traffic would be the root of the tree for the reply traffic. To use an already constructed tree for reverse traffic, an OP has to obtain the public key of an exit OR. An OP obtains this information from the *extended* packet. During the tree construction, when an OP extends the connection to an exit OR, it will set the flag in the *extend* packet to 1. When an OR receives a *group* packet with the $flag$ equal to 1, then it includes the public key of the exit OR of its own tree in the *grouped* packets. An OP forms the return address by encrypting its address with the given public key. The return address is included in the data packets sent by the OP. The structure of the packet and the path taken by a packet sent by an OP is shown in the Figure 11.

In the example shown in the Figure 11, the exit OR chosen by an onion proxy

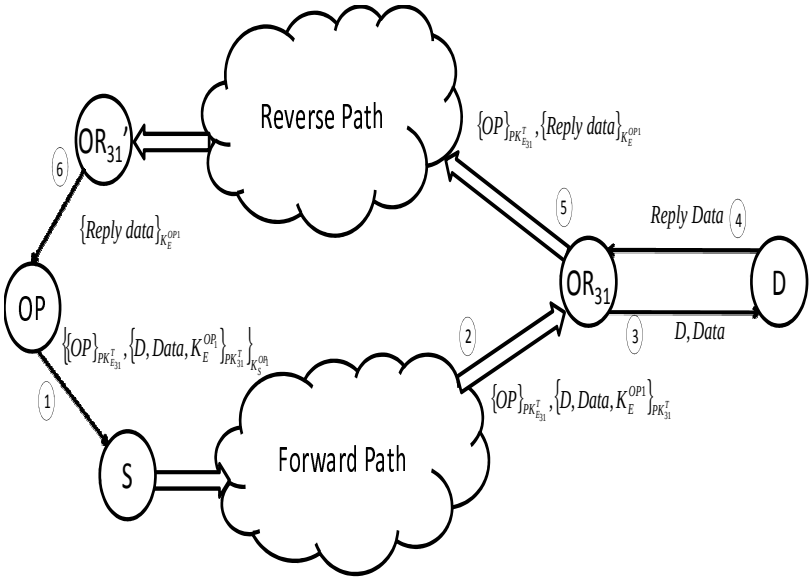


Figure 11: Reverse Traffic Path

OP_1 is OR_{31}. $OR_{31'}$ is the exit OR in the tree for which OR_{31} is the root. $PK^T_{E_{31}}$ is the public key of onion router $OR_{31'}$. OP_1 will use the public key of $OR_{31'}$ to form the return address $\{OP_1\}_{PK^T_{E_{31}}}$. An OP encrypts a data packet with the tree public key of the exit OR. In the example, a data packet from the OP_1 is encrypted with PK^T_{31}. An OP includes the address of the destination, data and a key to encrypt the reply data from the destination in the payload of a data packet. After decoding a generation, the onion router OR_{31} can decrypt $\left\{D, Data, K_E^{OP_1}\right\}_{PK^T_{31}}$ to get the final destination denoted by D. The onion router OR_{31} will forward $Data$ to the final destination D. The reply data from D is encrypted with $K_E^{OP_1}$. $K_E^{OP_1}$ is a symmetric key formed by OP_1 and it is sent to exit OR denoted by E to encrypt the reply data from a destination. The return address $\{OP\}_{PK^T_{E_{31}}}$ is attached to the *reply data*. The reply packet formed by OR_{31} is encoded along with other packets available with OR_{31} and is sent along the tree for which OR_{31} is the root and $OR_{31'}$ is one of the exit OR. The return address can be decrypted only by the onion router $OR_{31'}$. When the onion router $OR_{31'}$ decrypts the return address to get the identity of the OP, it will forward the encrypted *reply data* to the OP.

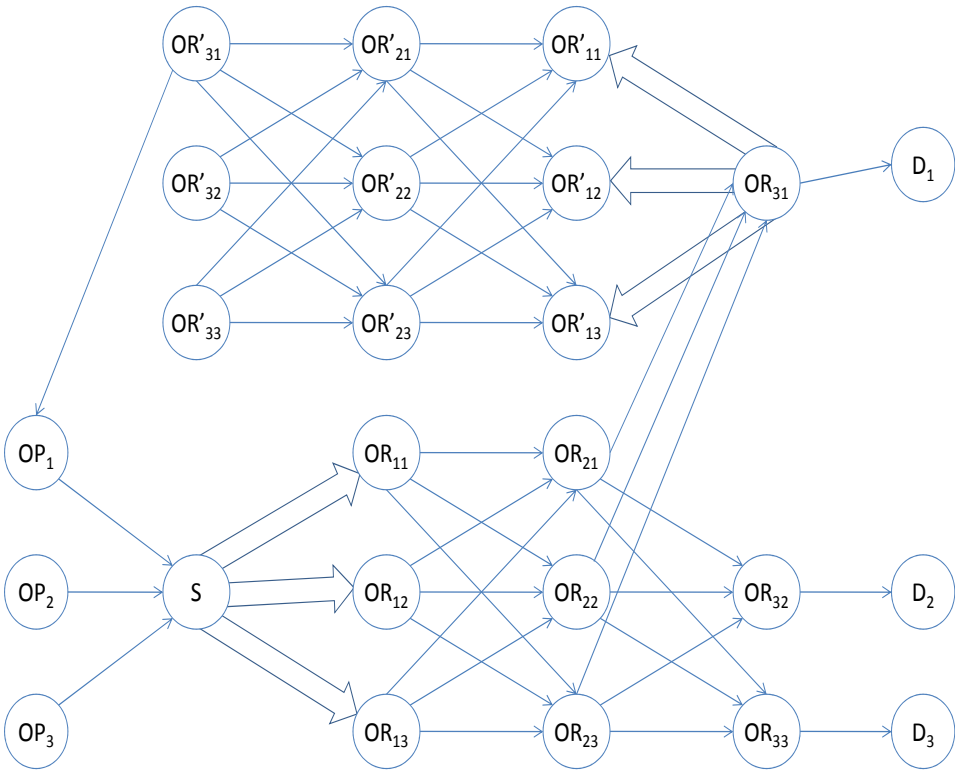


Figure 12: Multicast Tree after Grouping

An already constructed tree is used for return traffic because an OP cannot construct a new tree for the return traffic. For an OP to construct a tree, it has to connect to the root of the tree and send control packets for the tree construction. But if an OP connects to an exit OR to construct the tree, the exit OR would be able identify the OP and can relate the traffic. For example, if OP connects to the onion router OR_{31} to construct a tree with OR_{31} as the root OR for return traffic then the onion router OR_{31} knows that OP will use it as an exit OR in another tree and will be using its tree for reverse traffic. So to avoid this an already constructed tree is used for return traffic. The complete topology is as shown in the Figure 12.

5.5 Boot Up

When a system is booting up and there are no trees, the Directory Server (DS) helps in the tree construction. Every node that enters the network is supposed be the root of a tree. If there are no OPs available, the DS would randomly select a few ORs already in the network which send control packets to the currently entering OR and help in tree construction. When the OPs are available and ready to send packets, the initially constructed tree is torn down and a new tree is constructed.

5.6 Addition of a New OP

When a new OP opts to join the group of OPs using a tree then a new tree needs to be constructed. Since the addition of a new OP requires changes to all of the previous session keys and a new OR needs to be added to each layer, it is easier to form a new tree rather than adding ORs to the old tree. The old tree will not be torn down until the new tree is completely constructed. To join a tree, an OP sends a *create* packet to the root. The root will reply with a *created* packet. It sends a notification of new tree construction to all other OPs through a *new* packet. The root OR broadcasts this packet to all the OPs. On receiving *new* packet all the OPs reply with the *create* packets. And the new tree is constructed as already discussed.

5.7 Forwarding Techniques

The root waits for gs packets to be received from the OPs before it starts encoding the incoming packets (gs packets). The root OR forms the onion layers of encryption after forming the random linear combination of the packets received. Every OR when it receives a packet peels one layer of the onion encryption and forwards it to an OR in the next layer of the tree. The key used for layer decryption

is the key it shares with root and is same for all the ORs in that layer. The next layer OR is selected based on the algorithm being followed to select the next layer OR. We propose two algorithms; they are packet based and link based.

5.7.1 Packet Based Forwarding

In packet based forwarding, whenever a packet arrives at the intermediate OR it will remove a layer of encryption and forward it to a randomly selection OR in the next layer of the tree. There is no duplication of the packets at the intermediate nodes. The exit node will only be able to decode if it receives gs encoded packets from the previous layer. Because of random forwarding not all exit layer nodes will receive gs packets. There is a possibility that an exit node will receive less than gs packets of a particular generation. The probability that an exit layer node will be able to decode a generation is given by Equation 5.1

$$\frac{\sum_{i=0}^{N-gs} C_{gs+i}^{N} (n-1)^{N-gs-i}}{n^N} \tag{5.1}$$

Where value of n denotes the number of OR's per layer, gs denotes generation size and N denotes the number of unique and linearly independent packets formed by the root for the generation.

5.7.2 Link Based Forwarding

In link based forwarding when the packets of a particular generation arrive at an OR, it will forward the packet to next layer OR such that it will not send two packets of same generation to the same OR and no two ORs will recieve same packet. An OR on receiving a packet first decrypts a layer of the onion encryption and the forwards the packet to next layer while satisfying the following conditions

- A packet cannot be sent to more than one OR. There will not be any duplicate copy at any time in the multicast tree.

- If there are n OR's per layer in the tree then every OR will receive n packets from previous layer, one from each OR in previous layer for a particular generation. In this scheme of data forwarding the decoding probability is one because the exit layer will have at least gs packets, if it is assumed that $n >= gs$. Always number of ORs in each layer (n) must be greater than or equal to gs size so that each exit layer OR will receive at least gs packets for decoding a generation.

In link based forwarding, a seperate onion is formed for the header field GSQ because the intermediate ORs need to know the changes in GSQ.

5.7.3 Binary Coding Based Forwarding

In this scheme, each forwarding OR in the network will generate a random binary vector. The forwarding OR will combine received packets using the binary vector and forward combined packets. Since the coding vectors are generated randomly, there may be duplicate packets with the same encoding vectors. Because of these duplicate packets, the exit nodes may not have enough innovative packets for decoding. Decoding probability for binary coding based forwarding in discussed in Section 6.2.2

CHAPTER VI

THEORETICAL ANALYSIS

6.1 Detection Rate

Many of the existing anonymous communication systems suffer from timing analysis attack. In timing attacks, an adversery collects timings information from one part of the network and correlates with the information gathered at another part. To emulate such attacks packet timing information of a flow of interest is matched with the timing information collected at the exit layer ORs. To match the timing information collected mutual information concept is used. mutual information calculated between the input flow and the actual output flow is higher than mutual information calculated between the input flow and all other output flows then we can link the traffic thus detect the communication. Entropy calculated using mutual information is used as an indicator of communication detection. As mentioned, the detection rate should be equal to a random guess which is $1/n$ and entropy should be equal to $log_2 n$ In network coding, any encoded packet cannot linked to a particular original packet. An encoded packet contains a trace from all the original packets.

Since the root of the multicast tree encodes packets from different OPs, the resultant coded traffic cannot be linked to a particular OP and this combined with random packet forwarding beacuse of which all the ORs receive similar traffic makes it difficult to launch timings attacks.

6.2 Decoding Probability

6.2.1 Packet Based Forwarding

In a link based packet forwarding, the probability that a particular exit node will decode is 1 if $gs <= n$. Every OR will receive n packets from previous layer, one packet from each OR in the previous layer. The exit layer OR will decode once it receives gs packets belonging to a particular generation. But in the packet based forwarding, the decoding probability might be a value less than 1. On receiving a packet, an intermmediate OR forwards it to a randomly selected next layer OR. The random selection is independent of previous selections for packets belonging to the same or a different generation. Due to random forwarding there is a possibility that an exit OR might not receive gs packets that belong to the same generation.

Theorem 1 *The probability that an exit OR will decode a particular generation is given by*

$$\frac{\sum_{i=0}^{N-gs} C_{gs+i}^{N} (n-1)^{N-gs-i}}{n^N} \tag{6.1}$$

Proof N is the total number of encoded packets of a generation produced by root OR. Total number of ways N different packets distributed among n exit nodes is given by n^N, which is represented by T. The chances that a particular exit layer OR

will receive at least gs packets of a generation from N is given by F. So

$$F = \sum_{i=0}^{N-gs} C_{gs+i}^{N} \times (n-1)^{N-gs-i} \quad ,$$

and

$$T = n^{N} \quad .$$

The probability that a exit layer OR will decode a generation is given by $P = F/T$. Therefore

$$P = \frac{\sum_{i=0}^{N-gs} C_{gs+i}^{N} (n-1)^{N-gs-i}}{n^{N}} \quad .$$

6.2.2 Binary Coding Based Forwarding

An exit OR has to receive gs linearly independent packets of a generation to start decoding. Since the linear cofficients are choosen from GF(2) field the total number of linear combinations of gs original packets is given by 2^{gs}. An encoded packet is termed dependent if the encoding vector in that packet is a linear combination of already received linearly independent encoding vectors. In other words when an encoding vector in a packet can be formed by linear combination of encoding vectors previous received then the packet does not bring any new information or is not innovative thus does not contribute to decoding the generation.

Every exit OR receives n packets for a generation. For successfull decoding an exit OR should recive gs independent encoding vectors or should not receive more than $n - gs$ dependent packets. So the threshold on number of dependent packets an exit OR can receive is $n - gs$. If an exit OR receives more dependent packets than this value then it will not be able to decode that generation. The probability of an exit OR receiving gs independent encoding vectors is the sum of all probabilities where the number of received dependent packets is less than or to equal $n - gs$.

Theorem 1 *The probability that an exit OR will be able to decode a generation is given by*

$$P = \left(\left(1 - \frac{2^0}{2^{gs}}\right)\left(1 - \frac{2^1}{2^{gs}}\right)...\left(1 - \frac{2^{gs-1}}{2^{gs}}\right)\right) \left(1 + \left(\sum_{i=0}^{gs-1} \frac{2^i}{2^{gs}}\right) + \left(\sum_{i=0}^{gs-1} \frac{2^i}{2^{gs}} * \left(\sum_{j=0}^{gs-1} \frac{2^j}{2^{gs}}\right)\right) + ...(n-gs)terms\right) \tag{6.2}$$

Proof The probability that an encoding vector in an incoming packet is linearly independent depends on number of previously received linearly independent encoding vectors of the same generation. If m is the number of already received linearly independent encoding vectors then the number of linearly dependent vectors that can be formed from these m vectors is 2^m. So probability that an incoming packet has a linearly dependent encoding vector or an encoding vector from these 2^m dependent vectors is $\frac{2^m}{2^{gs}}$. As an example, the probability that the first packet received by an exit OR is linearly independent is $\left(1 - \frac{2^0}{2^{gs}}\right)$. By continuing, the probability that even the second packet received by an exit is also linearly independent is $\left(1 - \frac{2^0}{2^{gs}}\right)\left(1 - \frac{2^1}{2^{gs}}\right)$ and probability that the second packet is dependent on the first packet is $\left(1 - \frac{2^0}{2^{gs}}\right)\left(\frac{2^1}{2^{gs}}\right)$. Probability that gs out of gs packet received are linearly independent is $\left(1 - \frac{2^0}{2^{gs}}\right)\left(1 - \frac{2^1}{2^{gs}}\right)....\left(1 - \frac{2^{gs-1}}{2^{gs}}\right)$.

The above probability is the perfect case. But we can receive upto $n - gs$ dependent packets. As an example, the probability for the first packet to be linearly dependent is $\frac{2^0}{2^{gs}}$. By continuing further, probability that even the second packet received is dependent is $\left(\frac{2^0}{2^{gs}}\right)\left(\frac{2^0}{2^{gs}}\right)$ and probability that the second packet received by an exit is linearly independent is $\left(\frac{2^0}{2^{gs}}\right)\left(1 - \frac{2^0}{2^{gs}}\right)$. Probability that first packet received is linearly dependent and the next gs packets received are linearly independent is $\left(\frac{2^0}{2^{gs}}\right)\left(1 - \frac{2^0}{2^{gs}}\right)\left(1 - \frac{2^1}{2^{gs}}\right).....\left(1 - \frac{2^{gs-1}}{2^{gs}}\right)$. Probability that first and second

packet received are linearly dependent and the rest of gs packets received are linearly independent is $\left(\frac{2^0}{2^{gs}}\right)\left(\frac{2^0}{2^{gs}}\right)\left(1-\frac{2^0}{2^{gs}}\right)\left(1-\frac{2^1}{2^{gs}}\right).....\left(1-\frac{2^{gs-1}}{2^{gs}}\right)$. Extending the example even further, the probability that first packet received is linearly independent but the second and third packets received are linearly dependent and the rest of $gs-1$ packets received are linearly independent is
$\left(1-\frac{2^0}{2^{gs}}\right)\left(\frac{2^1}{2^{gs}}\right)\left(1-\frac{2^1}{2^{gs}}\right)\left(\frac{2^2}{2^{gs}}\right)\left(1-\frac{2^2}{2^{gs}}\right).....\left(1-\frac{2^{gs-1}}{2^{gs}}\right)$. Thus when we generalise the above example for n ORs per layer, then the probability of decoding a generation at an exit OR is given by summation of all the probabilities where number of received linearly dependent packets is less than $n-gs$, which is
$\left(\left(1-\frac{2^0}{2^{gs}}\right)\left(1-\frac{2^1}{2^{gs}}\right)...\left(1-\frac{2^{gs-1}}{2^{gs}}\right)\right)$
$\left(1+\left(\sum_{i=0}^{gs-1}\frac{2^i}{2^{gs}}\right)+\left(\sum_{i=0}^{gs-1}\frac{2^i}{2^{gs}}*\left(\sum_{j=0}^{gs-1}\frac{2^j}{2^{gs}}\right)\right)+...(n-gs)terms\right)$

6.3 Through-put Calculation

In [20] the authors have developed an analytical formula for the steady state throughput calculation of a TCP communication. It is widely accepted and used. The TCP throughput is determined by the probability of a packet being lost, the round trip time and time out value. The value of the throughput is limited by the maximum possible congestion window size. The formula is given in Equation 6.3.

$$B(p) \approx \min\left(\frac{W\max}{RTT}, \left(\frac{1}{RTT\sqrt{\frac{2bp}{3}}+T0\min\left(1,3\sqrt{\frac{3bp}{8}}\right)p\left(1+32p^2\right)}\right)\right) \quad (6.3)$$

RTT is the round trip time, W max is the maximum size of the congestion window, To is the timeout value, b is the number packets that are acknowledged by an ACK and p is the probability that a packet will be lost during the commu-

nication. We compare the theatrically calculated values of through-put $B(p)$ using Equation 6.3 against the practically obtained value. The throughput model developed in [20] for a TCP communication takes into account congestion avoidance behavior, retransmissions, timeouts and is valid for entire range of p.

CHAPTER VII

PERFORMANCE EVALUATION

We implement the proposed scheme using Network Simulator-2, which is an event based simulator built using C++ blocks. Extensive experiments are performed for various values of generation size (gs), number of ORs per layer (n) and number of layers (nl) in the multicast tree. By varying these parameters, we calculated delay, throughput and detection rate of a TCP communication. We studied the influence of varying the generation size and size of multicast tree on the performance metrics discussed in Chapter 6. The delay, throughput and detection rate of the communication calculated from the experiments are compared to the theoretically calculated values as explained in Chapter 6. The experiments are conducted over TCP connections. A Galois field of size 2^8 is used by the root for the random linear transformation of the incoming packets from the OPs. All the links in the topology are of 10 Mb capacity and delay on each link is 10 ms unless otherwise specified. Each simulation is carried out for 60 minutes. All the links between the intermediate OR's have on/off cross traffic with burst rate 5 Mbit/s, average burst time 500 ms and average idle time 500 ms. The size and other parameters of the multicast tree

for reverse traffic is same as multicast tree in the forward direction.

7.1 Packet Based Forwarding

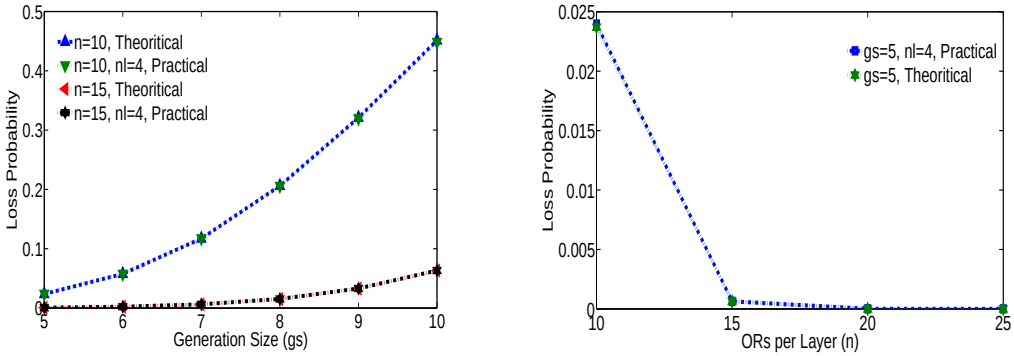
7.1.1 Loss Probability

Decoding a generation of encoded packets takes place only after an exit OR has collected gs packets of that generation. Since the packets are randomly forwarded from one OR to another in the next layer, there is a possibility that an exit OR might not receive gs packets of a generation. If the generation cannot be decoded, the exit OR is not be able to retrieve the original packet and it is lost.

Figure 13(a) shows the variations of loss probability by varying the generation size from 5 to 10. The decoding probability decreases with increase in gs with other parameters kept constant because more number of encoded packets are needed by an exit OR to start decoding. When decoding probability is high, the loss probability is less. Figure 13(a) indicates that as gs increases, the loss probability increases. Experimently obtained values are compared to the values calculated from equation 5.1. The loss probability in the experiments is obtained by the ratio of number of generations that could be decoded to number of different generations that were received by an exit OR.

The decoding probability increases with increase in number of ORs per layer n because there are more number of encoded packets of a generation in the tree, there is a greater chance that an exit OR will receive atleast gs encoded packets of a generation, thus decreasing the loss probability. This explains the decrease in loss probability with increase in n, as shown in Figure 13(b). Number of layers in the multicast tree do not effect the decoding probability because the packets are forwarded from one layer to another and decoding probability depends on distribution

of encoded packets among the exit layer ORs. So even if number of layers are varied the decoding probability remains the same.



(a) $layers$ =4 (b) gs =5

Figure 13: Comparison of Theoretical and Practical Values of Loss Probability

7.1.2 Throughput Variations

By Varying Generation Size

The experimently calculated throughput values are compared to theoritical values calculated from Equation 6.3. Figure 14(a) shows throughput variations when gs is increased from 5 to 10. The throughput decreases with increase in generation size because as gs increases, more number of encoded packets are required by an exit ORs to decode. As gs increases with other parameters kept constant, the decoding probability decreases as suggested by Equation 5.1. This will lead to increase in loss probability, which decreases the throughput as suggested in Equation 6.3. It can be seen that rate of throughput decrease is less for $n = 20$ than $n = 15$ because more number of encoded packets of a generation are present in tree when n is greater.

The effectiveness of an anonymous communication system is determined by the rate at which the communication is detected. The detection scheme used here is

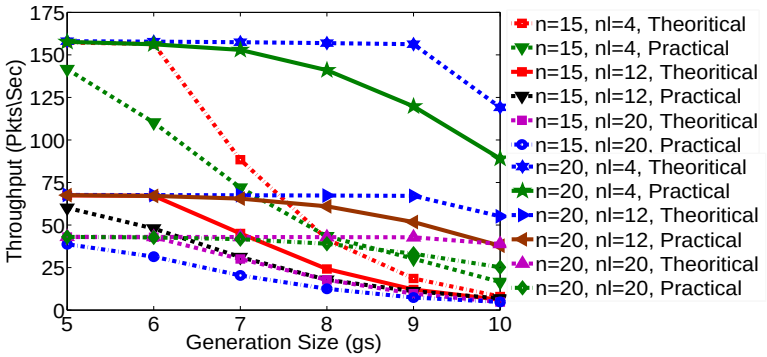
based on timing analysis and is discussed in Section 6.1. The detection rate of a communication using the proposed system is obtained using entropy which is calculated by applying the concept of mutual information and correlation on the packet timings gathered at the sender and the exit ORs. Figure 14(b) and Figure 14(c) show that the entropy values calculated from correlation and entropy values calculated from mutual information technique. The timing information of a flow of interest is compared with the timing information of aggregate flow at each exit OR. It can be seen that entropy calculated from experiments are almost equal to the entropy of equal distribution indicating that all the exit ORs received similar amount of traffic. So the probability of detecting the traffic is equal to probability of a random guess. As the number of exit ORs increases, the entropy increases which decreases the probability of detection. The probability of detection is less for $n = 20$ compared to $n = 15$.

By Varying Number of Layers

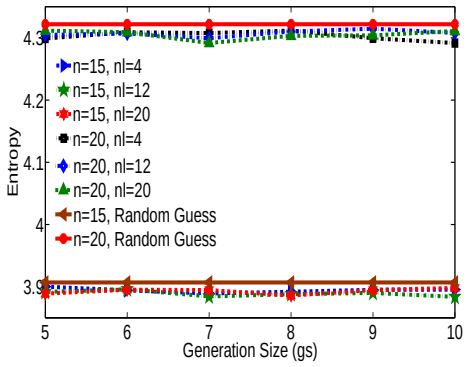
The round trip time increases with number of layers because the hop count increases with layers in the multicast tree. When the number of hops (delay) increases, the throughput of a TCP communication decreases as suggested by the Equation 6.3 and this explains the dip in the value of throughput in the Figure 15(a) for increase in number of layers (nl) from 4 to 20. In Figure 15(b) and Figure 15(c) it can be seen that varying nl in the multicast tree does not change the detection rate because the detection rate depends on the number of ORs in the layer rather than the number of layers itself. So n =15 has a different detection rate from n =20.

By Varying Number of ORs Per Layer (n)

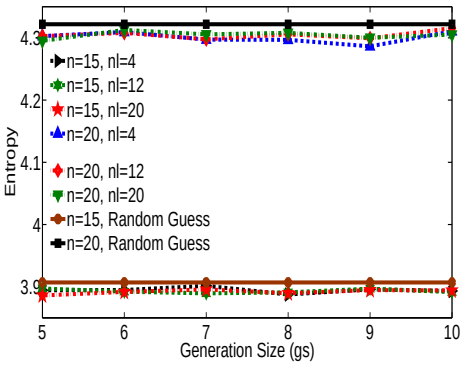
In Figure 16(a), n is varied from 10 to 25 for 4, 8 and 12 layers in multicast tree and gs =5. The throughput value increases with increase in the number of ORs



(a) Throughput Variations for n =15, 20 and $layers$ = 4, 12, 20

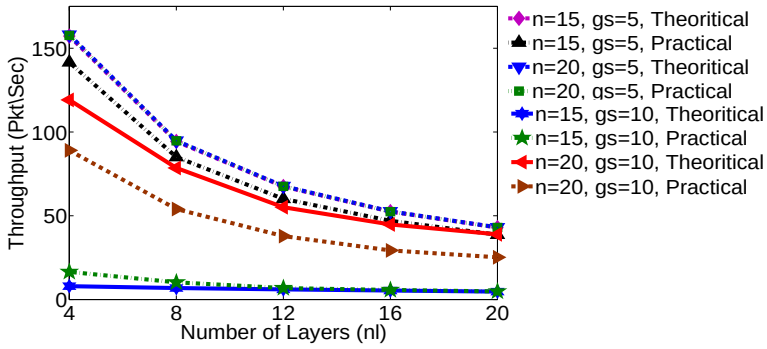


(b) Mutual Information Detection Rate

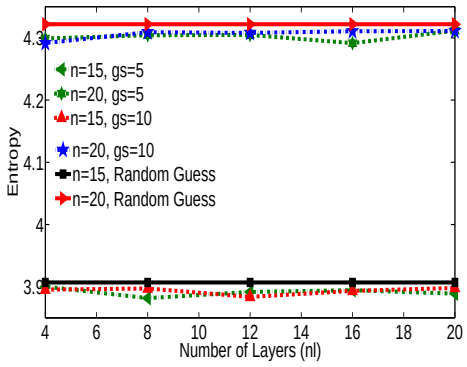


(c) Correlation Detection Rate

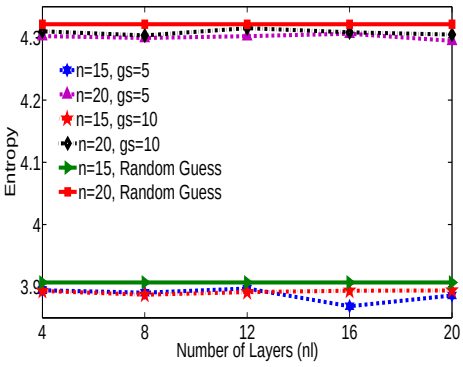
Figure 14: Comparison of Theoretical and Practical Values of Throughput and Detection Rate for n =15, 20 and $layers$ = 4, 12, 20



(a) Throughput Variations for n =15, 20 and gs = 5, 10



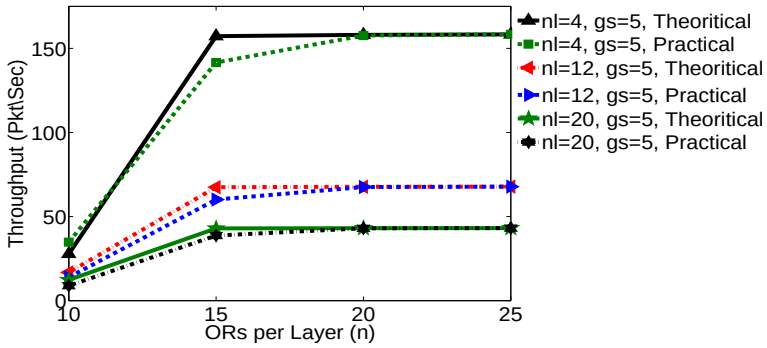
(b) Mutual Information Detection Rate



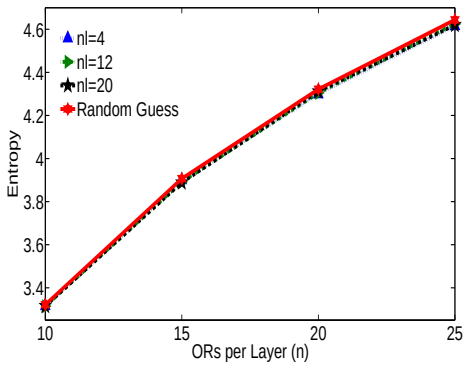
(c) Correlation Detection Rate

Figure 15: Comparison of Theoretical and Practical Values of Throughput and Detection Rate for n =15, 20 and gs = 5, 10

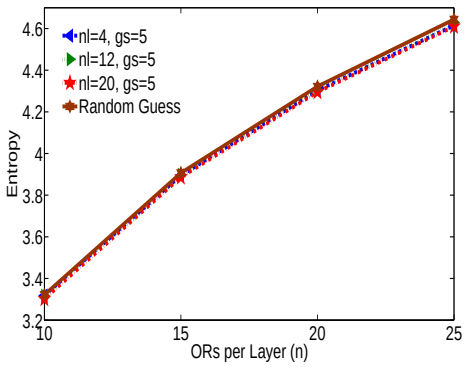
per layer. This is because as n increases there will be more encoded packets in the tree and so the loss probability decreases. Increase in decoding probability will lead to decrease in loss probability, thus increasing the throughput. So n =25 has greater throughput than n =10 as indicated in the Figure 16(a). In Figure 16(b) and Figure 16(c), the entropy for n =25 is greater than n =10 because the traffic is equally distributed among larger group of exit ORs.



(a) Throughput Variations for gs =5 and $layers$ = 4, 12, 20



(b) Mutual Information Detection Rate



(c) Correlation Detection Rate

Figure 16: Comparison of Theoretical and Practical Values of Throughput and Detection Rate for gs =5 and $layers$ = 4, 12, 20

7.2 Link Based Forwarding

The link based forwarding is discussed in the section 5 and as mention the decoding probability of a link based forwarding technique is almost equal to 1.

7.2.1 By Varying Generation Size

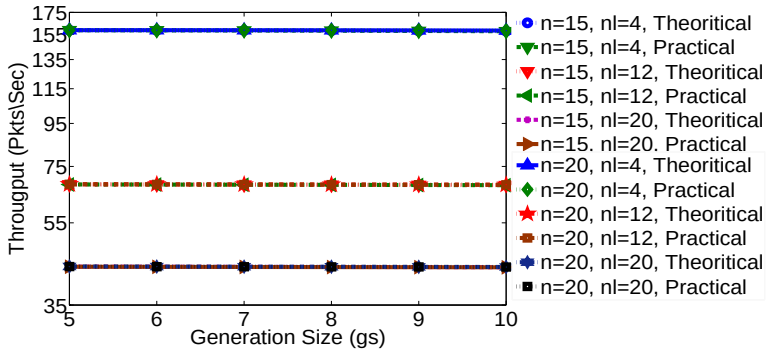
Figure 17 shows throughput variations for link based forwarding scheme. It can be observed that throughput values are uneffected by variation in gs because when the decoding probability is 1 and loss probability is zero, then the throughput value is the maximum possible value determined by RTT as suggested by Equation 6.3. Throughput for $layers$ =4 is more than $layer$ =12 because as number of layers increase, the RTT value increase resulting in decrease of throughput. Figure 17(b) and Figure 17(c) show entropy values for varying gs in the tree. Since all the ORs in the exit layer receive similar traffic, the entropy values are almost equal to entropy of a random guess, indicating that traffic cannot be detected.

7.2.2 By Varying Number of Layers

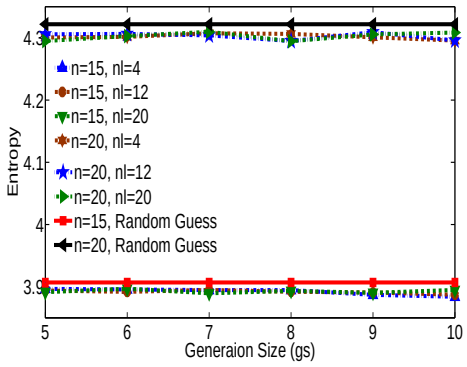
In Figure 18(a) it can be seen that as number of layers are increasing, the throughput is decreasing because of increase in RTT with increase in number of hops to final destination. The entropy values is same for $nl = 4$ and $nl = 8$ as shown in Figure 18(b) and Figure 18(c) because the number of layers do not effect the traffic distribution among the exit layer ORs and it is effected only by n.

7.2.3 By Varying Number of ORs per Layer(n)

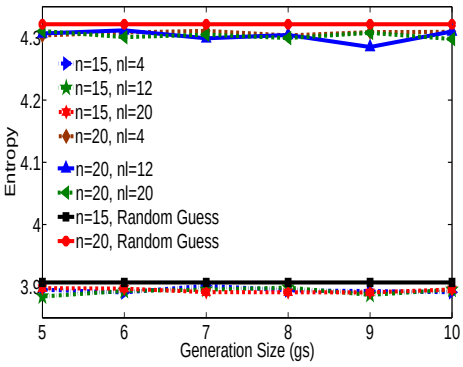
In Figure 19(a) it can been seen that number of ORs per layer does not effect the throughput. It is because increasing n does not effect the loss probability as it



(a) Throughput Variations for n =15, 20 and $layers$ = 4, 12, 20

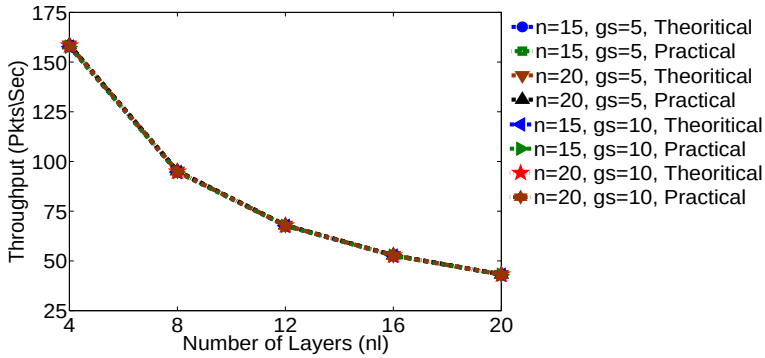


(b) Mutual Information Detection Rate

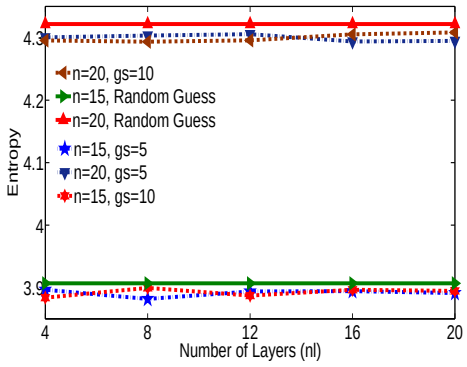


(c) Correlation Detection Rate

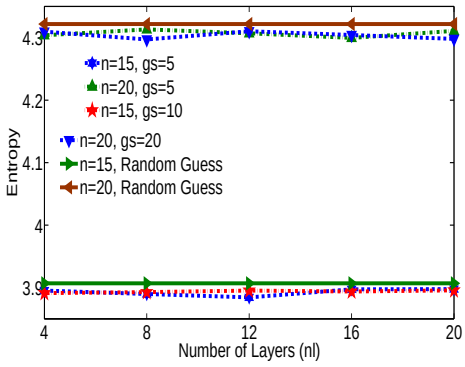
Figure 17: Comparison of Theoretical and Practical Values of Throughput and Detection Rate for n =15, 20 and $layers$ = 4, 12, 20



(a) Throughput Variations for n =15, 20 and gs = 5, 10



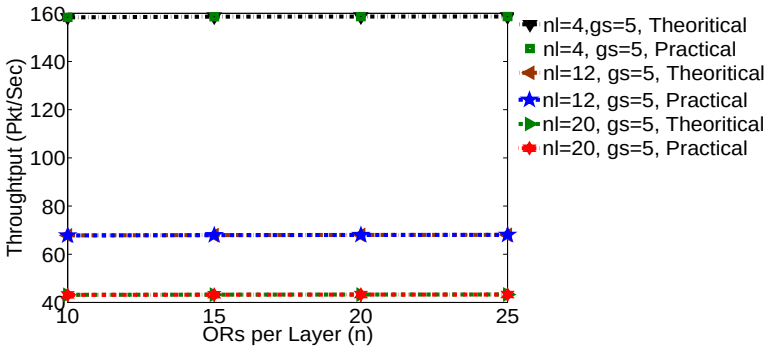
(b) Mutual Information Detection Rate



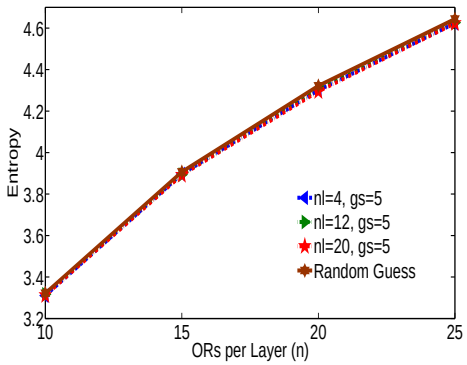
(c) Correlation Detection Rate

Figure 18: Comparison of Theoretical and Practical Values of Throughput and Detection Rate for n =15, 20 and gs = 5, 10

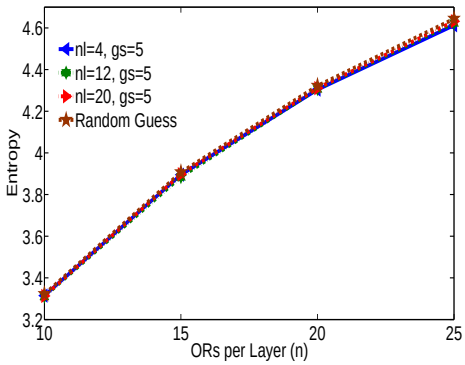
will in case of packet forwarding technique and when the loss probability is zero then throughput will reach the maximum possible value determined by RTT. In Figure 19(b) and Figure 19(c) the entropy value increases with n, indicating that probability of detection decreases with increase in the number of ORs per layer because the traffic is distributed among more ORs.



(a) Throughput Variations for gs =5 and $layers$ = 4, 12, 20



(b) Mutual Information Detection Rate



(c) Correlation Detection Rate

Figure 19: Comparison of Theoretical and Practical Values of Throughput and Detection Rate for gs =5 and $layers$ = 4, 12, 20

7.3 Binary Coding Based Forwarding

7.3.1 Loss Probability

Decoding a generation of encoded packets takes place only after an exit OR has collected gs linearly independent encoding vectors of that generation. As suggested by Equation 6.2, the probability that a generation is decoded by an exit OR depends on the generation size gs and number of ORs per layer n in the multicast tree.

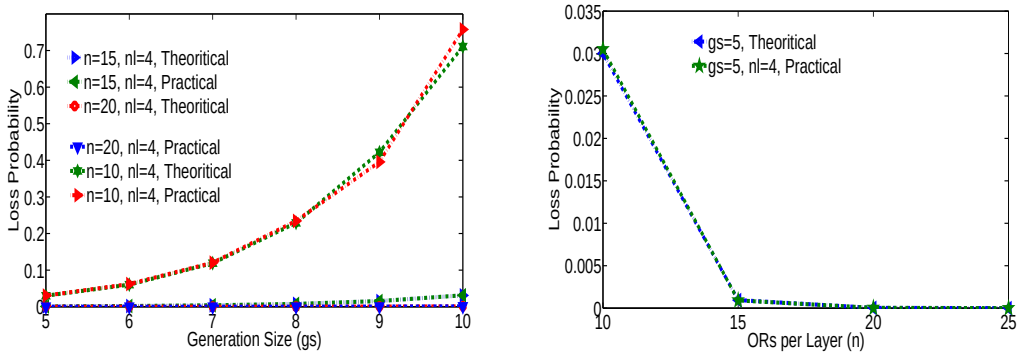
Figure 20(a) shows the variations of loss probability when generation size increases. The decoding probability decreases with increase in gs because more number of linearly independent packets are needed by an exit OR to start decoding. When decoding probability is high, the loss probability is less. Figure 20(a) indicates that as gs increase the loss probability increases. Experimently obtained values are compared to the values calculated from equation 6.2.

Since each exit OR receives n packets for a generation, the loss probability decreases with increase in number of ORs per layer n, as there is a greater chance that an exit OR will receive gs linearly independent packets. This explains the decrease in loss probability with increase in n, as shown in Figure 20(b). Number of layers in the multicast tree do not effect the decoding probability because the packets are forwarded from one layer to another and decoding probability depends on number of encoded packets received by the exit layer ORs and generation size. So even if number of layers are varied the decoding probability remains the same.

7.3.2 Throughput Variations

By Varying Generation Size

The experimently calculated throughput values are compared to theoritical values calculated from Equation 6.3. Figure 21(a) shows throughput variations when

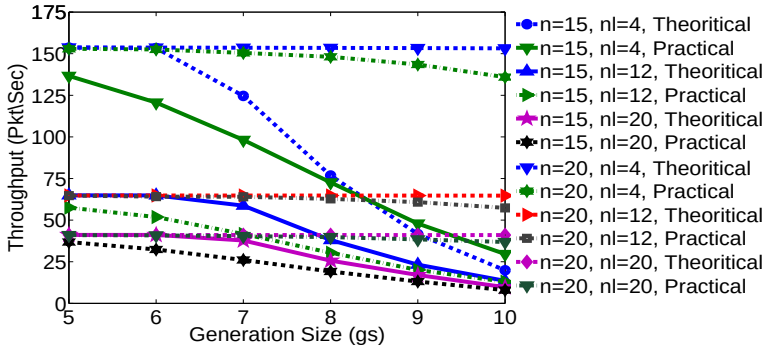


(a) $layers$ =4 (b) gs =5

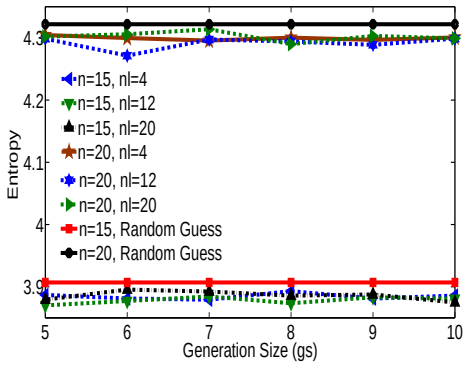
Figure 20: Comparison of Theoretical and Practical Values of Loss Probability

gs is increased from 5 to 10. When other parameters are kept constant, the throughput decreases with increase in generation size because as gs increases, more number of linearly independent encoded packets are required by the exit ORs to decode. As gs increases, the loss probability increases leading to a decrease in throughput as suggested in Equation 6.3. It can be seen that rate of throughput decrease is less for $n = 20$ than $n = 15$ because loss probability is less for higher value of n.

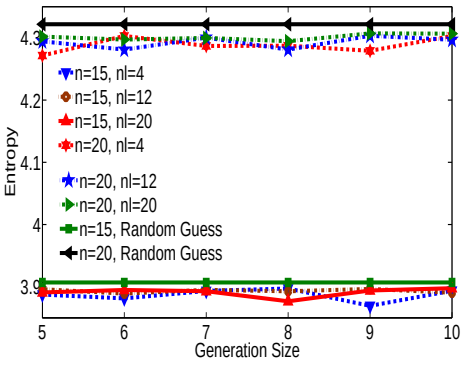
The detection rate of a communication using the proposed system is obtained using entropy which is calculated by applying the concept of mutual information and correlation on the packet timings gathered at the sender and all of the the exit ORs. Figure 21(b) and Figure 21(c) show the entropy values calculated from entropy values calculated from mutual information technique and correlation. It can be seen that entropy calculated from experiments are almost equal to the entropy of equal distribution indicating that all the exit ORs received similar amount of traffic. So the probability of detecting the traffic is equal to probability of a random guess which is $1/n$. As the number of exit ORs increases, the entropy increases which decreases the probability of detection. The probability of detection is less for $n = 20$ compared to $n = 15$.



(a) Throughput Variations for n =15, 20 and $layers$ = 4, 12, 20



(b) Mutual Information Detection Rate

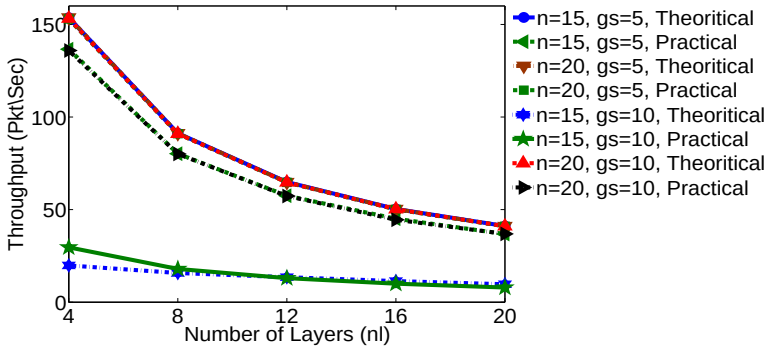


(c) Correlation Detection Rate

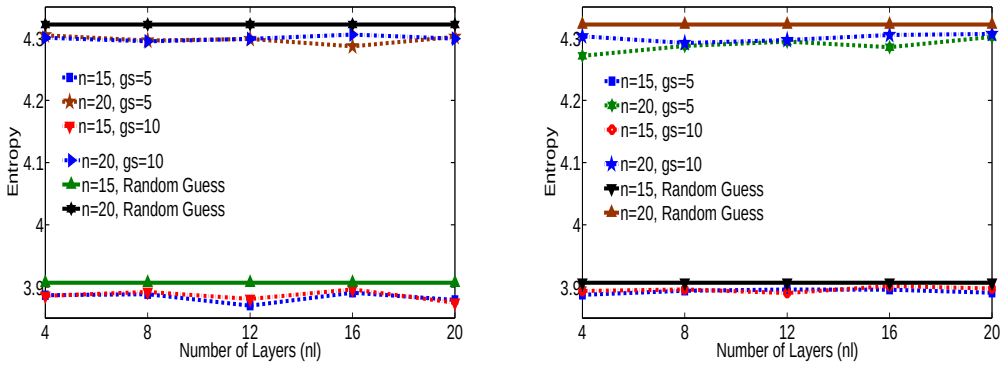
Figure 21: Comparison of Theoretical and Practical Values of Throughput and Detection Rate for n =15, 20 and $layers$ = 4, 12, 20

By Varying Number of Layers

The round trip time increases with number of layers because the hop count increases with layers. When the number of hops (delay) increases, the throughput of a TCP communication decreases as suggested by the Equation 6.3 and this explains the dip in the value of throughput in the Figure 22(a) for increase in number of layers (nl) from 4 to 20 in steps of 4. In Figure 22(b) and Figure 22(c) it can be seen that varying nl in the multicast tree does not change the detection rate because the detection rate depends on packet timings and among the ORs in a layer rather than the number of layers.



(a) Throughput Variations for n =15, 20 and gs = 5, 10

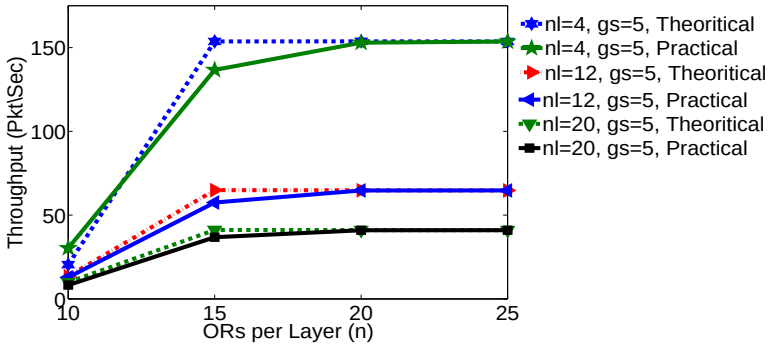


(b) Mutual Information Detection Rate (c) Correlation Detection Rate

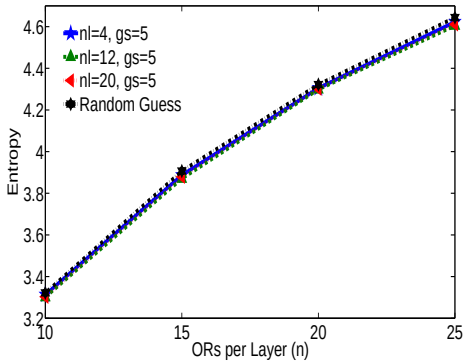
Figure 22: Comparison of Theoretical and Practical Values of Throughput and Detection Rate for n =15, 20 and gs = 5, 10

By Varying Number of ORs Per Layer (n)

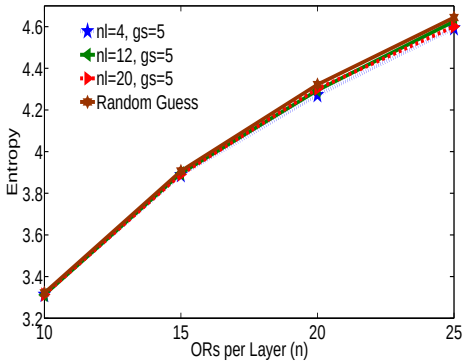
In Figure 23(a), n is varied from 10 to 25 in steps of 5. The throughput value increases with increase in the number of ORs per layer because an exit OR will receive n encoded packets and so the probability of receiving gs independent packets is higher. A tree with n =25 has greater throughput than n =10 as indicated in the Figure 23(a). In Figure 23(b) and Figure 23(c) the entropy for n =25 is greater than n =10 because the traffic is equally distributed among larger group of exit ORs.



(a) Throughput Variations for gs =5 and $layers$ = 4, 12, 20



(b) Mutual Information Detection Rate



(c) Correlation Detection Rate

Figure 23: Comparison of Theoretical and Practical Values of Throughput and Detection Rate for gs =5 and $layers$ = 4, 12, 20

CHAPTER VIII

POSSIBLE ATTACKS

We merge the idea of network coding with the concept of onion routing to disrupt timing attacks while still preserving the important qualities of onion routing to disrupt many other attacks.

- *Packet counting*: Using network coding we combine packets from different users to form a batch/generation and these set of packets are encoded/decoded together. So simple packet count technique will not reveal any information as destination will receive more packets as it has to decode after collecting gs packets to get the actual information.

- *Payload Check*: Since we are using onion routing, a layer of onion is decrypted at every hop. So the input packet at a node is not identical to output packet. The adversary cannot map the input traffic to output traffic just depending on the payload.

- *Finger printing*: In order to launch this attack the adversary first creates a traffic pattern for well known servers depending on the file transfer and compares

the traffic pattern of a user against database of known server traffic pattern. Since forward traffic uses different multicast tree from the multicast tree of reverse traffic, it will be difficult to build the incoming traffic pattern because of encoding/decoding process and collection of gs packets.

- *Compromise the keys*: The packets are iteratively encrypted with the session keys the source OR has established with the intermediate ORs. Even if the session key of a layer is compromised the adversary cannot get any information due to other layers of encryption which are not yet decrypted. The effect of these attacks can be reduced further by periodically changing the session keys. If the TLS key is compromised then the effect will last only for the life time of that particular key [7].

- *Running a hostile OR*: By using network coding we are spreading the information equally in all the encoded packets and the destination has to collect gs packets to decode. Even if the adversary is dropping few packets it will not affect the communication any anonymity but it will affect the loss probability.

- *Introducing timing patterns*: The adversary can introduce timing patterns into the traffic but it will not affect the communication because the destination has to get gs packets of a generation and does not care about the order or the sender of those packets (as long as it is previous layer OR). If the packets on a link are delayed then the destination will decode with the packets it received from other OR and carry on with the communication.

- *Tagging Attacks*: By integrity checks we can avoid such attacks where the packet contents are altered [7].We also use TLS encryption that would identify and alteration to the encode packets.

- *End to End timing Correlation* These attacks are not a part of design goals. Such attacks can be avoided by being a part of the anonymous communication system rather than just a user because the traffic inside the system cannot be detected.

CHAPTER IX

CONCLUSION

An onion routing based anonymous communication system is presented which disrupts the timing analysis attacks. The detection rate for different sizes of the multicast tree is calculated. The probability detecting a communication is equal to probability of detection through a random guess which is $1/n$. Number of OR's in the leaf layer can be increased to increase the degree of anonymity. The information spreading quality of network coding helps to produce similar traffic on all the links. Since packets from multiple communications can be combined to produce an encoded output traffic that evenly spreads in the tree, detecting the traffic based on timing information becomes very difficult. An exit OR just needs to collect any but sufficient encoded packets to decode and get actual information. The TCP throughput of a communication using the proposed system is compared with the value obtained from Equation 6.3. The loss probability in packet based forwarding technique and binary code based technique can be decreased by increasing the number ORs per layer of the multicast tree.

BIBLIOGRAPHY

[1] The anonymizer. `http://www.anonymizer.com`.

[2] R. Ahlswede, N. Cai, S. yen Robert Li, R. W. Yeung, S. Member, and S. Member. Network information flow. *IEEE Transactions on Information Theory*, 46:1204–1216, 2000.

[3] O. Berthold, H. Federrath, and S. Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 115–129. Springer-Verlag, LNCS 2009, July 2000.

[4] N. Cai and R. Yeung. Secure network coding. In *IEEE International Symposium on Information Theory*, July 2002.

[5] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, 1981.

[6] R. G. Dimakis, P. B. Godfrey, Y. Wu, M. O. Wainwright, and K. Ramch. Network coding for distributed storage systems. In *In Proc. of IEEE INFOCOM*, May 2007.

[7] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proc. of the 13th USENIX Security Symposium*, pages 303–320, San Diego, CA, August 2004.

[8] C. Fragouli. Abstract network coding: An instant primer.

[9] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network

layer. In *In Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002*, pages 193–206, 2002.

[10] R. D. G. Danezis and N. Mathewson. maxminion: Design of typeiii anonymous remailer. In *IEEE Symposium on Security and Privacy*, pages 12–15. IEEE CS, May 2003.

[11] C. Gkantsidis, J. Miller, and P. Rodriguez. Comprehensive view of a live network coding p2p system. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 177–188, New York, NY, USA, 2006. ACM.

[12] C. Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. 2005.

[13] D. M. Goldschlag, M. G. Reed, and P. F. Syverson. Hiding Routing Information. In R. Anderson, editor, *Proceedings of Information Hiding: First International Workshop*, pages 137–150. Springer-Verlag, LNCS 1174, May 1996.

[14] T. Ho, R. Koetter, M. Mdard, D. R. Karger, and M. Effros. The benefits of coding over routing in a randomized setting. In *In Proceedings of 2003 IEEE International Symposium on Information Theory*, 2003.

[15] T. Ho, M. Mdard, R. Koetter, D. R. Karger, A. Member, M. Effros, S. Member, S. Member, S. Member, J. Shi, and B. Leong. A random linear network coding approach to multicast. *IEEE Trans. Inform. Theory*, 52:4413–4430, 2006.

[16] S. Katti, J. Cohen, and D. Katabi. Information slicing: Anonymity using unreliable overlays. In *Proceedings of the 4th USENIX Symposium on Network Systems Design and Implementation (NSDI)*, April 2007.

[17] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. Xors in the air: practical wireless network coding. *SIGCOMM Comput. Commun. Rev.*, 36(4):243–254, 2006.

[18] R. Koetter, M. Mdard, and S. Member. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, 11:782–795, 2003.

[19] D. S. Lun, M. Medard, R. Koetter, and M. Effros. Further Results on Coding for Reliable Communication over Packet Networks. *ArXiv Computer Science e-prints*, Aug. 2005.

[20] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling tcp throughput: A simple model and its empirical validation. Technical report, Amherst, MA, USA, 1998.

[21] A. Pfitzmann and M. Waidner. Networks without user observability. *Comput. Secur.*, 6(2):158–166, 1987.

[22] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for web transactions. Technical report, 1997.

[23] M. Rennhard and B. Plattner. Practical anonymity for the masses with morphmix. In A. Juels, editor, *Proceedings of Financial Cryptography (FC '04)*, pages 233–250. Springer-Verlag, LNCS 3110, February 2004.

[24] A. Serjantov and P. Sewell. Passive attack analysis for connection-based anonymity systems. In *In Proceedings of European Symposium on Research in Computer Security (ESORICS*, pages 116–131, 2003.

[25] M. Steiner, G. Tsudik, and M. Waidner. Key agreement in dynamic peer groups. *Parallel and Distributed Systems, IEEE Transactions on*, 11(8):769 –780, aug 2000.

[26] P. Syverson, M. Reed, and D. Goldschlag. Onion Routing access configurations. In *Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX 2000)*, volume 1, pages 34–40. IEEE CS Press, 2000.

[27] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr. Towards an Analysis of Onion Routing Security. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 96–114. Springer-Verlag, LNCS 2009, July 2000.

[28] J. Tan and M. Medrad. Secure network coding with a cost criterion. In *Proc. 4th International Symposium on Modeling and Optimization in Mobile Ad Hoc and Wireless Networks*, April 2006.

[29] P. P. U. Moller, L. Cottrell and L. Sassaman. Mixmaster protocol – version 2. draft, 2003.

[30] J. P. Vilela, L. Lima, and J. Barros. Lightweight security for network coding. *CoRR*, abs/0807.0610, 2008.

[31] J. Widmer and J.-Y. Le Boudec. Network coding for efficient communication in extreme networks. In *WDTN '05: Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 284–291, New York, NY, USA, 2005. ACM.

[32] S. yen Robert Li, S. Member, R. W. Yeung, and N. Cai. Linear network coding. *IEEE Transactions on Information Theory*, 49:371–381, 2003.

[33] P. C. Yunnan, P. A. Chou, Y. Wu, and K. Jain. Practical network coding, 2003.

[34] X. Zhang, G. Neglia, J. Kurose, and D. Towsley. On the benefits of random linear coding for unicast applications in disruption tolerant networks. pages 1 – 7, april 2006.

[35] Y. Zhu and R. Bettati. Unmixing mix traffic. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2005)*, pages 110–127, May 2005.

[36] Y. Zhu and R. Bettati. Compromising anonymous communication systems using blind source separation. *ACM Trans. Inf. Syst. Secur.*, 13(1):1–31, 2009.