TORSA

Project Documentation

Contents

INTRODUCTION 4

PROJECT OBJECTIVES 5

RESULTS SUMMARY 6

REPORT OUTLINE 6

DESIGN ASPECTS 7

THE ONION ROUTER 7

THE NETWORK COMPONENTS 9

APP-PROXY 9

CLIENT 10

ROUTER 10

SERVER 10

ROUTING TABLE 10

THE MESSAGES TRANSFERRED 12

AES ENCRYPTION 13

ENCRYTING MESSAGES 13

KEY TRANSFER 13

MESSAGE TRANSFER 13

CONCEALABILITY 14

MASKABILTY..14

ACCOUNTABILITY..14

RESULTS

PERFORMANCE BASED METRICS..18

SECURITY BASED METRICS..18

CONCLUSION..22

REFERENCES..24

Introduction

Second Generation Onion routing

Internet Anonymity

What is internet anonymity? Internet anonymity applies to any interaction a user has on the Internet that protects his or her identity from being shared with another user or with a third party. This also means to ensure that your activities cannot be traced back to your IP address.

There are several applications of Internet Anonymity, some trivial yet others excessively crucial. The vast range of applications that incorporate internet anonymity include:

- Anonymous blogging and posting (Twitter).
- Forums that allow anonymous exchange of questions and answers.
- Secure Billing; allowing users to purchase items online, without having to reveal any personal information (PayPal).
- Anonymous peer-to-peer file sharing. This is one of the main examples that has been projected in this document.
- It plays a vital role in many military applications to ensure that intercept-able data is passed along in a secure manner. In fact this is the very purpose for generation of TOR, which is the protocol that we will be exploring shortly.

Onion Routing

"Onion routing was developed in the mid-1990s at the U.S. Naval Research Laboratory by employees Paul Syverson, Michael Reed, and David Goldschlag to protect U.S. intelligence communications online. It was further developed by the Defense Advanced Research Projects Agency (DARPA) and patented by the Navy in 1998." [1]

Following shortly after the inception of the internet, anonymity and security have become one of the foremost concerns of its users. With advancements in the number of risks involved over the internet in terms of hacking, it is essential to employ finer algorithms to stay in the lead.

[1] Extracted from the article "Onion Routing" on Wikipedia

Currently “private browsing” and “proxy servers” are looked upon as the most robust source internet Anonymity. But honestly, the Internet is never truly anonymous. No matter what implementations are used to secure involved parties, there is always a way to trace back to the trace back to the user from a given activity, making the entire system vulnerable. The real solution is to make the tracing back as difficult as possible. That’s where “onion routing” comes in.

Onion routing is a technique for anonymous communication over a computer network. In an onion network, messages are encapsulated in layers of encryption, analogous to layers of the vegetable onion. The encrypted data is transmitted through a series of network nodes called onion routers, each of which "peels" away a single layer, uncovering the data's next destination. When the final layer is decrypted, the message arrives at its destination. The sender remains anonymous because each intermediary knows only the location of the immediately preceding and following nodes.

Project Objectives

The main objective of this project is to design and implement the second generation Onion Routing protocol between two users, through a TOR network. This network assists in masking the message of the sender, so that neither the message nor information regarding both the final participants of the transaction can be intercepted by any third party. In order to establish legitimacy and uniformity, the distribution of keys and routing algorithms are centralised through an authoritative proxy.

Results Summary

Based on data captured via Wireshark, we were able to successfully demonstrate the feasibility of our system. We also determined, theoretically, the shortcomings of using symmetric cryptography and its potential ramifications on the performance. Despite these shortcomings, our simulations demonstrated that we were able to successfully mask the contents of our broadcasted message and also conceal the identity of our users. Additionally, while attacking the system is still hard, it was still vulnerable to the different traffic analysis strategies employed by attackers.

Report Outline

This report aims at explaining the working details of the Onion Routing Protocol. We delve deeper into the actual implementation details, highlighting every design decisions.

Furthermore, we emphasise on the accountability of the implementation, by touching up on topics such as maskability; and proceed on with the testing portion of our project.

The final conclusions drawn will highlight both the robust aspects of our model and scopes for future improvement.

Design Aspects

The Onion Router

The Onion Routing Protocol, first presented in 1996, was designed to prevent any attacker from performing traffic analysis of a network. To help understand the need for such a protocol better, let us consider a simple analogy of a post office. Generally, a letter is placed in an envelope which is marked with the sender's and recipient's addresses. Naturally, the sender of the letter trusts that the post office does not peek inside the envelope (privacy reasons) and does not monitor the communication between it and the recipient. In the modern day context, this analogy can be applied to emails or any other form of electronic communication, thus, making it vital to protect the privacy of such messages. Granted that the communicating parties tend to reveal their identities to each other sometimes, there is no reason why the identities and the nature of the conversation between the aforementioned users should be revealed by the network to external parties. The protocol, therefore, addresses these two main problems in wireless security: eavesdropping and traffic analysis. Alternatively, this also led the researchers to devise a strategy to protect the identity of the users. This meant that not only were the contents of the message required to be protected but any identifying information attached to the message must also be protected and hidden from the external observer.

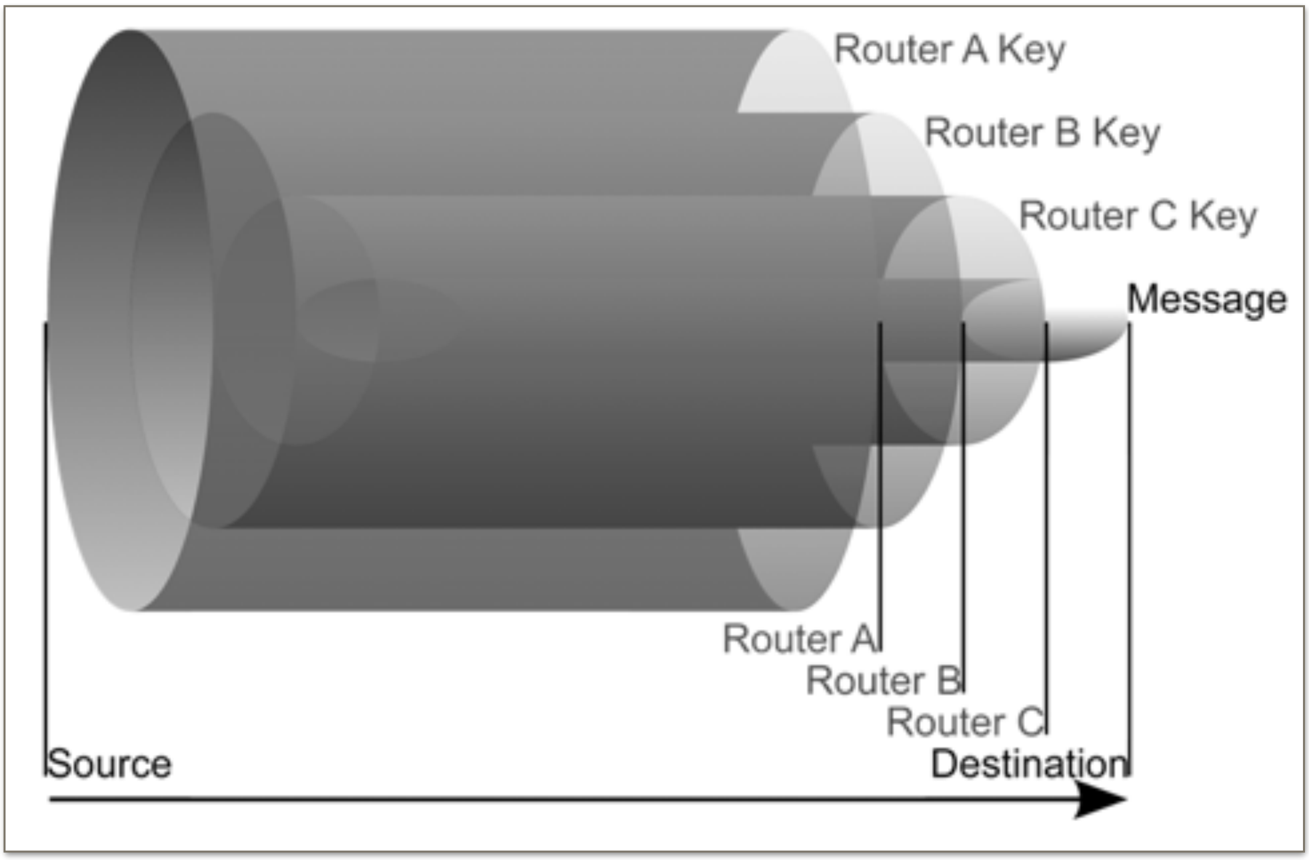
Subsequently, the researchers examined whether it was possible to devise a system whereby two parties can communicate with each other even if one or both of them do not want to be identified to the other and keep the contents of their communication private. The result of this ground-breaking research was the implementation of the first generation Onion Routing Protocol. The highlighting feature of this protocol was that even if an eavesdropper were to perform traffic analysis, the study of traffic patterns would not reveal much information about the paths of messages.

To initiate a communication, a simple plaintext message is encrypted (wrapped) with successive layers of encryption (onion) such that each layer can be decrypted (unwrapped) by one intermediary (node or router) in a succession of intermediaries in the path taken by the onion routers (circuit). The message transmission is accomplished as follows:

- The sender picks nodes from a list which provide a path for transmitting the message.

- Using asymmetric cryptography, the sender encrypts the message with the public keys of the chosen nodes which are obtained from an advertised list.

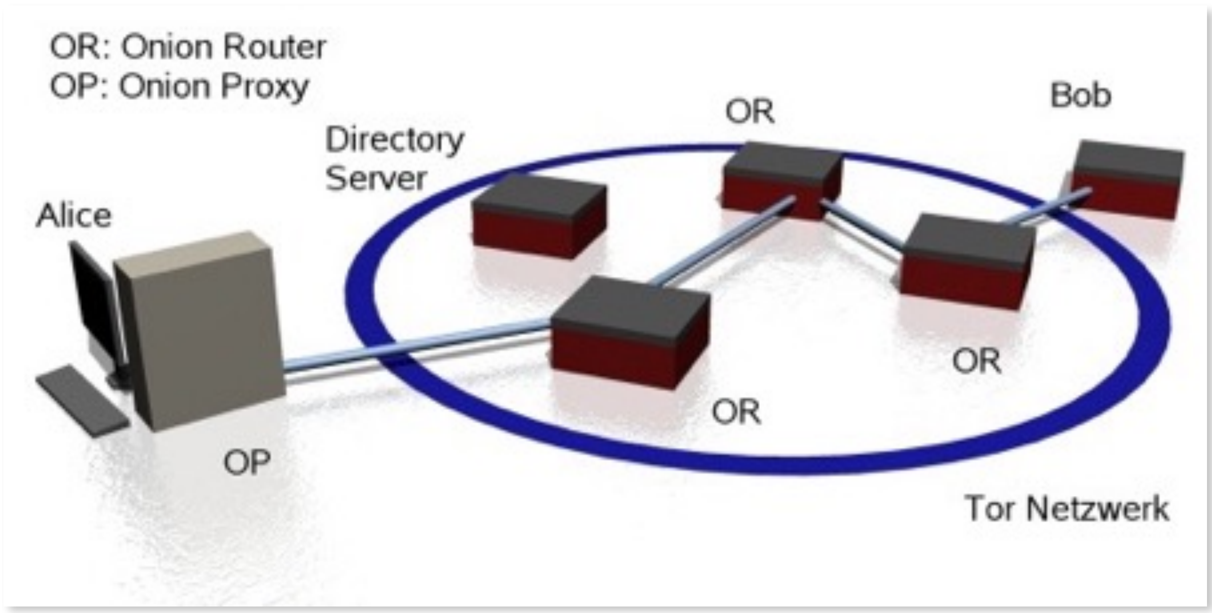
- As the message passes through each node, a layer of encryption is removed by the receiving nodes (by using their private keys) until it reaches its destination.



The Network Components

The TOR network consists of mainly 6 components. These include: the centralised App-Proxy, the Client (Let’s name her Alice), the Server (Let’s name him Bob), the intermediate routers (in this case 4), the encryption utilities, and our routing table.

The following diagram is a perfect pictorial representation of our algorithm which should be kept in mind while going through the remainder of this documentation.



App-Proxy

Application Proxy does the network setting up on part of client and acts as interface between client and onion proxy provided constitutively by onion routers, giving client freedom of not knowing the actual protocol behind the scene. At the initialisation time, the list of onion routers is available to it. It then decides an appropriate sequence of onion routers in order to transmit the message from the source to the destination. It distributes the keys to each of the routers. After this, network is set up. At the time of client to server communication, it encrypts the message whereas during server to client communication, it decrypts it and passes it to client.

Client

The client (in this case Alice), first initialises itself to and binds to its socket. After the central authority (Application proxy) has initialised all the routers and distributed their keys. It sends the message to the Application proxy along with the address of the receiver. It then waits for a certain period of time, after which it assumes that the packet is lost and retransmits the certain message till it receives an acknowledgment.

Router

The number of routers can be arbitrary. Although care has to be taken. Too few routers (say one or two) may be cracked with considerable amounts of efforts, whereas the too many routers can consume a lot of time during encryption and decryption.

There are three main things that occur in each router. During the initialising phase, each router is passed a message with the key and the router that it has to pass on the messages to. With this message it initialises itself, preparing itself for the message routing phase.

The message routing phase occurs in two parts. Message that come from its previous address have to decrypted by its key and the message has to be passed on to the next router, in the list. This is anonymous to the peeling of the onion. This message contains the actual data that has to be sent to the receiver.

The messages that come from the next address in the routing address have to be encrypted and passed on back to the client. This consists of the acknowledgment of receipt of the message from the receiver (Bob) to the sender (Alice).

Server

The first task of the server is synonymous to the rest of the routers. It initialises itself with the key it receives, which will later be used to encrypt and decrypt the messages.

In the next phase the server (in this case Bob) receives the message and decrypts it with its key. It waits for a brief period of time (5 seconds) in order to prompt for a personal Acknowledgement from the receiver. If no such message is inputted, the server encrypts and sends back a NULL acknowledgment.

Routing Table

Each entity of the network has been assigned a particular address in the default_data file. The main components of the TOR network are in fact the routers themselves.

In reality a TOR router us a volunteer router which forms a network with other TOR routers, agreeing to follow certain protocols and participating in the security of the data (through encryption and decryption).

In order for the TOR to work effectively, proper management of the TOR routers is essential. Hence we create a routing table which indicates the participants of the TOR network and the order in which the assemble.

Network Messages

There are two main forms of messages that are being passed throughout the simulation of the network. These include:

- Initialising data
- The actual messages

The format of both of these types of data are described below.

In order to send messages over the network through AES utilities, it is mandatory to maintain the size of the data to be multiples of 16 bytes, all this is taken care of as follows:

INITIALISING DATA

```
_key_XXXXXXXXXXXXXXXX_nexttip_('127.0.0.1', 6300)
  del1         key           del2          address
```

This is a 48 byte message where 14 bytes are used for delimiters to indicate the key and the next address, 16 bytes are used for the automatic generated encryption key and the remaining bytes are reserved for the address.

All this information in depicted in the diagram above.

MESSAGE FORMAT

```
$.....$_data_message
keyword        message
```

In real life messages can be of arbitrary length and care has to be taken to fit it to our requirements. In our project this is done through the padding function, which fills ip all the holes which the reserved keyword “$”.

It is the duty of the server to unpad the data and retrieve the original message.

AES Encryption

Advanced Encryption Standard (AES) has been used for encryption. AES uses a symmetric key encryption, meaning that the same key is used both for encryption as well as decryption. Needless to say, each of the router is assigned a different key. Python contains a standard module Crpyto.Cipher which contains AES encryption and decryption functions.

In addition to the normal key, the encryption and decryption modules of the recent Crypto.Cipher require an additional field, known as the Initialising Vector (IV). The same IV has to be used for both encrypting and decrypting the message, and hence for implementation purposes, the IV was attached as the first 4 bytes of each message that was transferred.

Encrypting Messages

There are two types of messages that are transmitted through the network. These are the keys and the actual messages, as discussed above. The following gives an accurate summary of how these two types of data are encrypted and decrypted while being transmitted across the network.

KEY TRANSFER

At the onset of connection, right after the client, routers and server are up, the first thing application proxy does is to distribute the keys to each of the routers. First of all, the application proxy transmits the key to the first router in an unencrypted manner. Then for distributing the next key, it is first encrypted by key of the first router and is sent to the first router which decrypts it and forwards the actual key to the second router. In this way, every subsequent key is first encrypted through multiple levels of encryption and is then routed via previous routers.

Note that the key transmitted during setting up of connection contains both the AES Encryption Key and also the address of the next router. The address of the previous router is obtained trivially. In this way, every router has hold of only its previous and next router address. Also, so as to recognise that the intended data packet is a key transfer packet not a regular data packet, it is first padded by a starting delimiter.

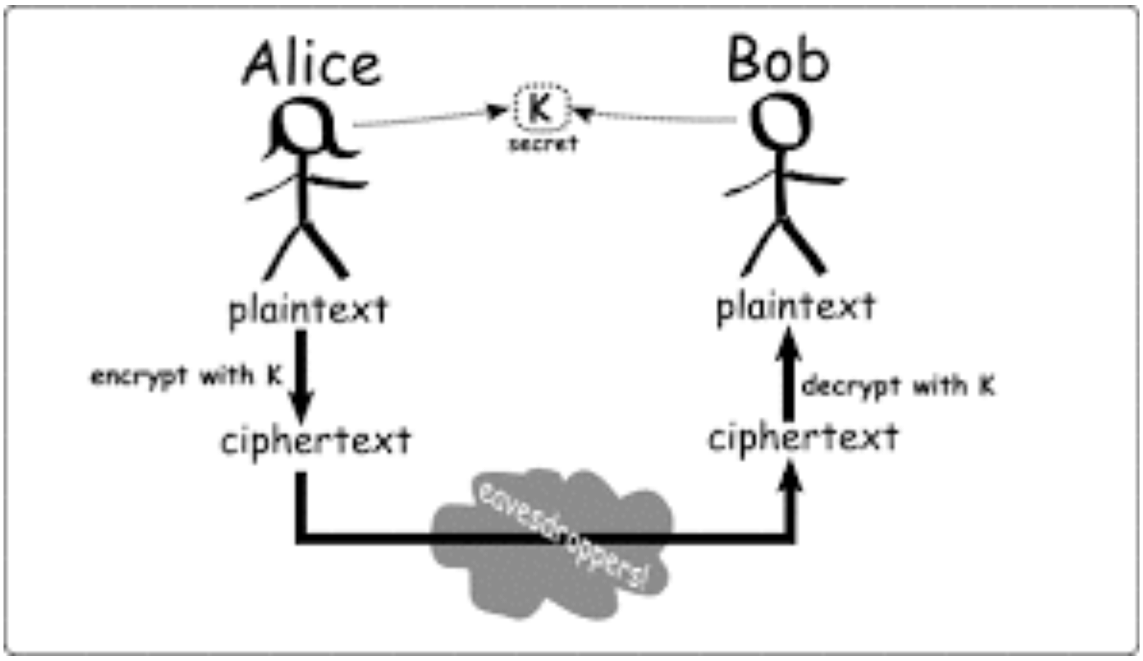
MESSAGE TRANSFER

Similar to key transfer protocol, the only difference in message transfer is that all the routers are now set up with their respective keys. It is to note that in our implementation, client waits for acknowledgement from server for certain amount of time and on timeout it resends the packet.

As evident from the figure shown on page 9, the application level proxy is on the client machine (Alice). The client communicates to router via application proxy. The figure shows a message being sent from Client to the first OR encrypted multiple times. Each of the subsequent transmission strips off encryption by one level. Finally at the last router, the entire message gets decrypted and is forwarded to the destination (Bob).

Mask-ability

To mask the contents of the messages, public-key cryptography strategies were implemented (PKI). The figure below describes the process of a typical symmetric encryption process:



Accountability

To maintain and establish trust between the nodes within the network, we designed and implemented a certification authority (CA) system. Every single node has access to the CA's public key which was later used by the nodes themselves to encrypt the messages being transmitted.

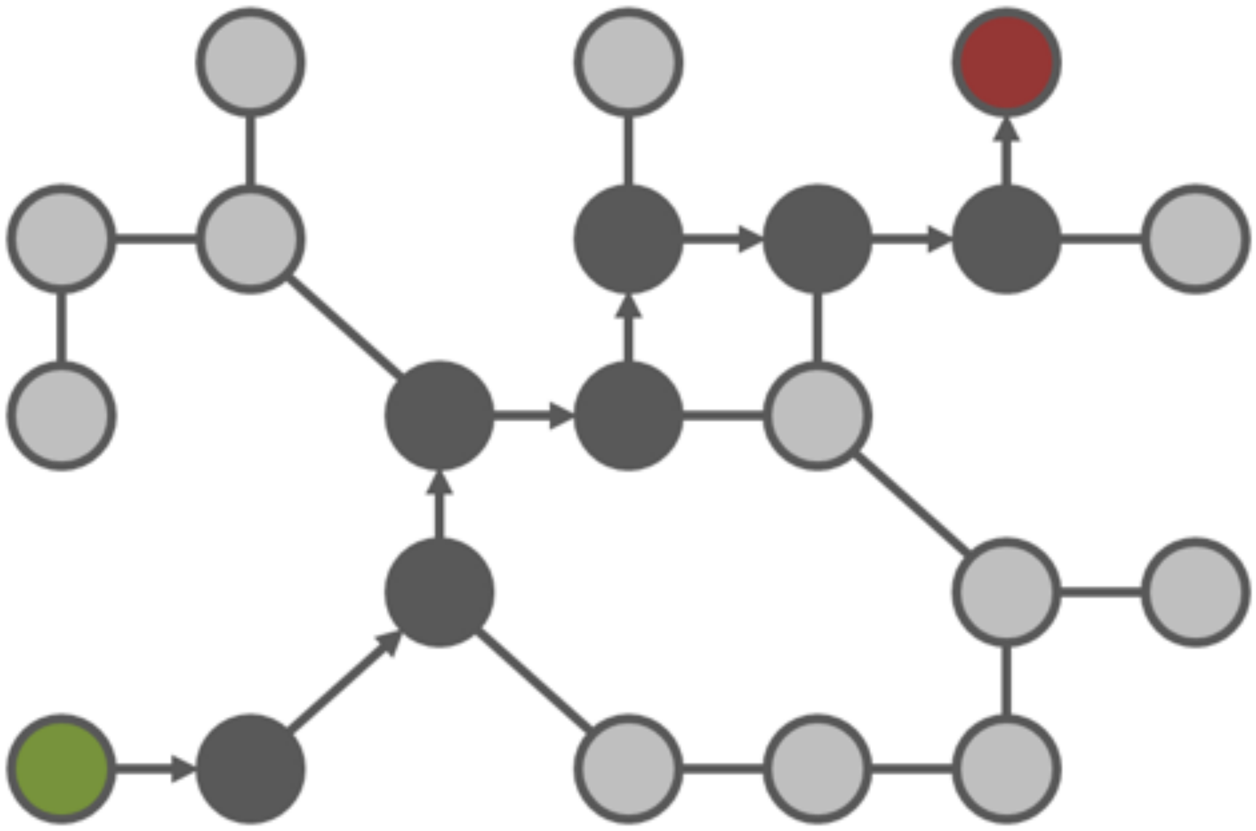
Once the CA's public key has been made available to the nodes, the only way a node can initiate a communication with another node is by signing its public key with the CA's public key. In other words, this ensures that the node is authorized to communicate in the network and its messages are deemed trustworthy. One thing to bear in mind is that all the nodes must use the same level of encryption to initiate a two-way conversation, i.e., a 256-bit encrypted node cannot initiate a conversation with a 128-bit node and vice-versa. This is due to the encryption-decryption process utilized by the RSA algorithm.

Finally, once all the nodes have been signed and authorized, they are free to communicate with each other.

Concealability

The main purpose of this criterion is to obscure the path taken by nodes to transmit the message. Ideally, this feature helps discourage attackers from gaining any useful information by performing traffic analysis. Moreover, it also makes eaves- dropping an infeasible solution to determine the nature of communication between multiple parties.

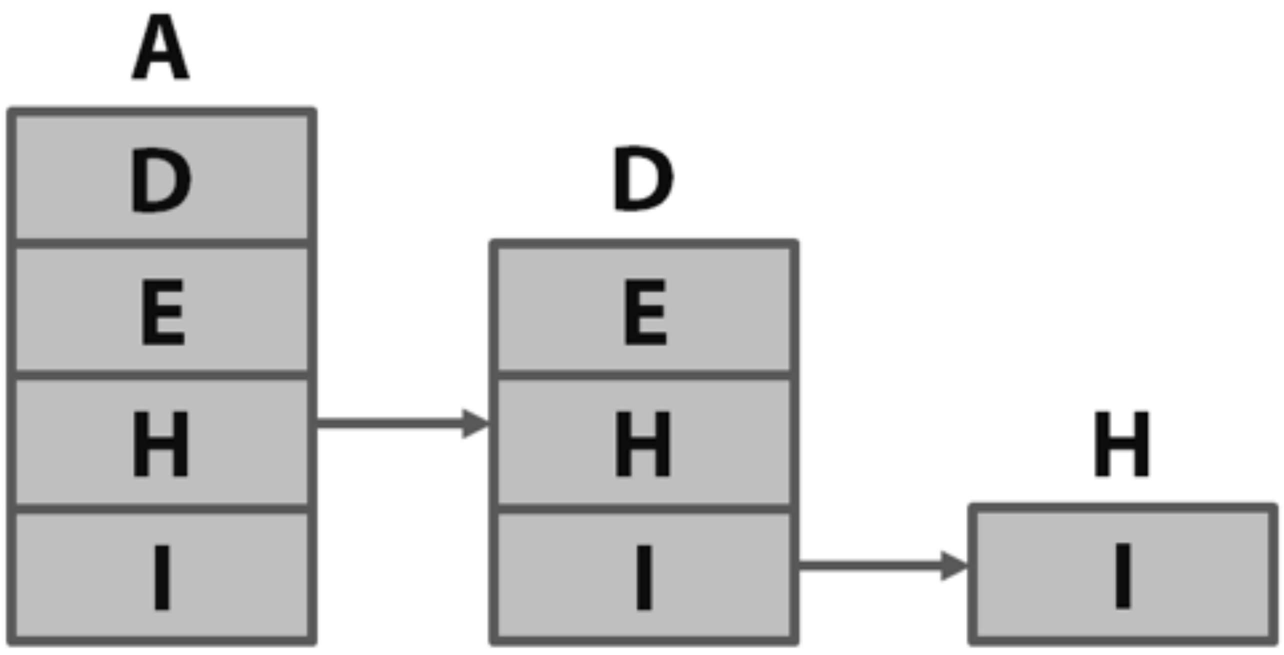
To better understand the nature of this problem, let us consider a simple scenario. Let us assume that Alice wants to send a message to Bob. Under normal circumstances, Alice can transmit a message to Bob by using any standard routing protocol which will simply broadcast its message to its neighbours until it reaches its target (multi-hopping):



10

To address this problem, we decided to encrypt the message with multiple layers of encryption. In any given network, every authorized node can simply display a list of active nodes in the zone (NDP), which is all the information it needs to transmit a message to any existing node in the network. Ideally, the node then builds a “circuit” to transmit its message to its destination, while encrypting the message (in order) with the public keys of the nodes in the “circuit”. This process is described as depicted below:

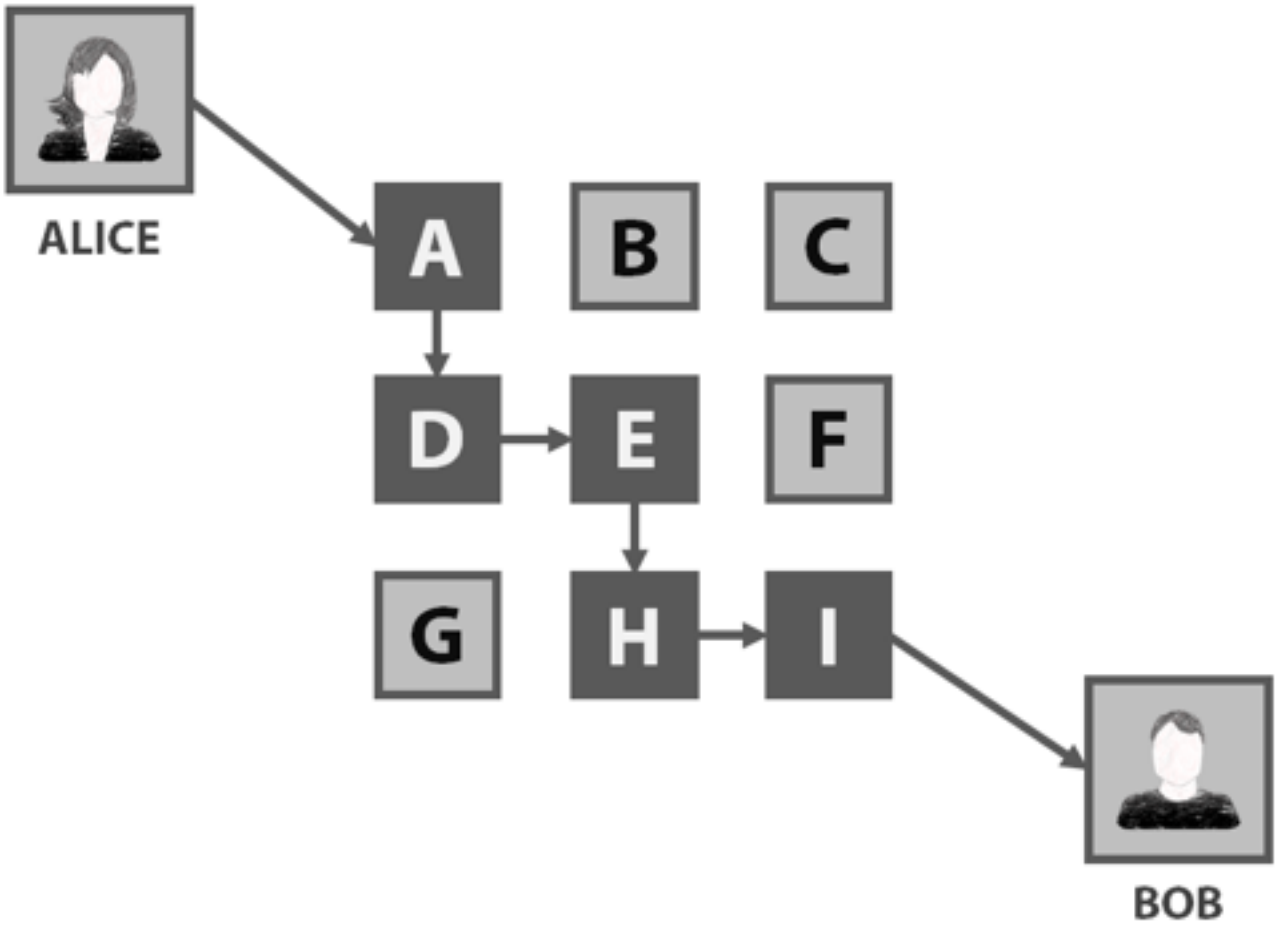
The encryption is performed by the following nested loop algorithm as depicted below:



```
$ Send [Destination] [Message ...]
```

```
Find Destination Address in the Table
    Adjust Message to be of Same length as Key
    Encrypt Message with Destination Key
    Loop Though Destination Address Path
        Find Address in Address Path
            Encrypt Message with Address Key
```

Once the message has been encrypted, Alice can communicate freely with Bob, anonymously. Our approach also conceals the identity of the users themselves (unless they choose to reveal their identities):



Results

Based on our implementation strategies, we decided to use both performance based and security based metrics to analyse and test the feasibility of our system.

We made extensive use of Wireshark in order to monitor and capture the flow of packets being transmitted and to analyze the structure of each packet to determine if any useful information could be obtained.

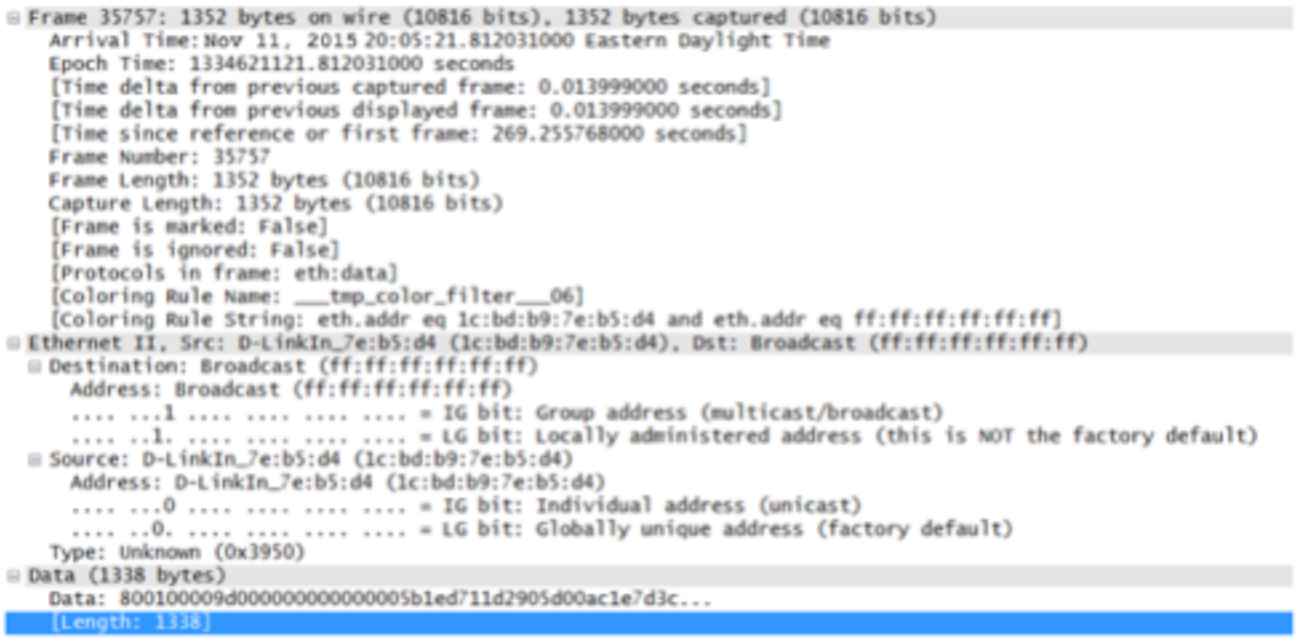
PERFORMANCE BASED METRICS

Our packet essentially consists of a simple message wrapped in successive layers of encryption. RSA encryption and decryption can be a tedious process and may influence the performance of the network. To successfully apply multiple layers of encryption to our message, our “encryption layering” algorithm made use of nested loops. This unfortunately yielded an O (N^3) algorithm, which is deemed poor by our performance requirements. During a live demonstration, there was little noticeable difference in performance as we chose a direct path to transmit a message. This problem would most certainly be noticeable if we were to employ a multi-path route. In addition to this, the extra overhead incurred due to the encryption and decryption at various stages, we hypothesize that this would most certainly have a severe impact on the performance within the network. Wireshark, unfortunately, was unable to provide us with any useful data to analyze the performance of packet delivery. However during our simulations, we did not notice any issues with performance since the application was deployed on an extremely small scale capacity and the lab environment only contained a handful of nodes for establishing our routing protocol.

We were, however, able to procure enough data by using Wireshark. By inspecting each, individual packet and analyzing the information presented by Wireshark, we could perform a small scale traffic analysis to try and determine the viability of the protocol itself. This naturally, leads us to perform a security metrics analysis with the help of packet analyzers built into Wireshark.

SECURITY BASED METRICS

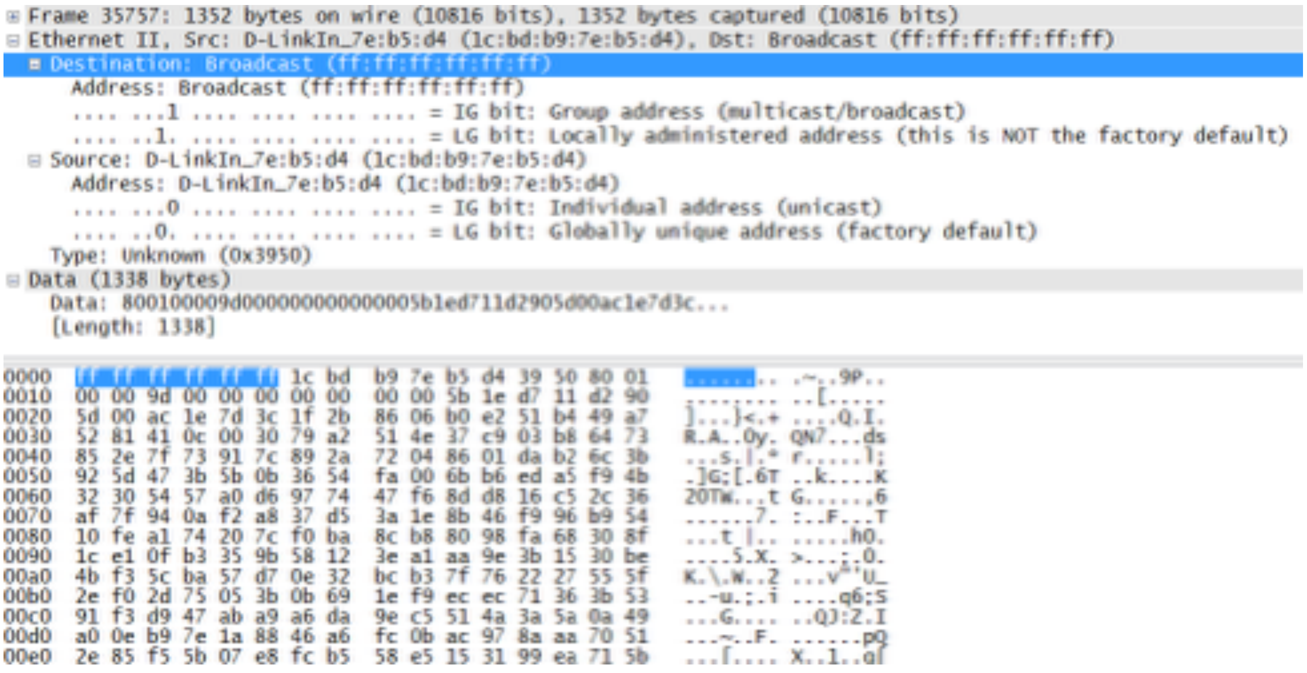
Below is a representation of what a typical broadcast looks like in Wireshark:



```
⊟ Frame 35757: 1352 bytes on wire (10816 bits), 1352 bytes captured (10816 bits)
    Arrival Time:Nov 11, 2015 20:05:21.812031000 Eastern Daylight Time
    Epoch Time: 1334621121.812031000 seconds
    [Time delta from previous captured frame: 0.013999000 seconds]
    [Time delta from previous displayed frame: 0.013999000 seconds]
    [Time since reference or first frame: 269.255768000 seconds]
    Frame Number: 35757
    Frame Length: 1352 bytes (10816 bits)
    Capture Length: 1352 bytes (10816 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:data]
    [Coloring Rule Name: ___tmp_color_filter___06]
    [Coloring Rule String: eth.addr eq 1c:bd:b9:7e:b5:d4 and eth.addr eq ff:ff:ff:ff:ff:ff]
⊟ Ethernet II, Src: D-LinkIn_7e:b5:d4 (1c:bd:b9:7e:b5:d4), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ⊟ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
      Address: Broadcast (ff:ff:ff:ff:ff:ff)
      .... ...1 .... .... .... .... = IG bit: Group address (multicast/broadcast)
      .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factory default)
  ⊟ Source: D-LinkIn_7e:b5:d4 (1c:bd:b9:7e:b5:d4)
      Address: D-LinkIn_7e:b5:d4 (1c:bd:b9:7e:b5:d4)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    Type: Unknown (0x3950)
⊟ Data (1338 bytes)
    Data: 800100009d000000000000005b1ed711d2905d00ac1e7d3c...
    [Length: 1338]
```

The captured packet reveals the destination address, source address, data length and message type. The destination of a broadcast packet is depicted below:

Nov 11, 2015

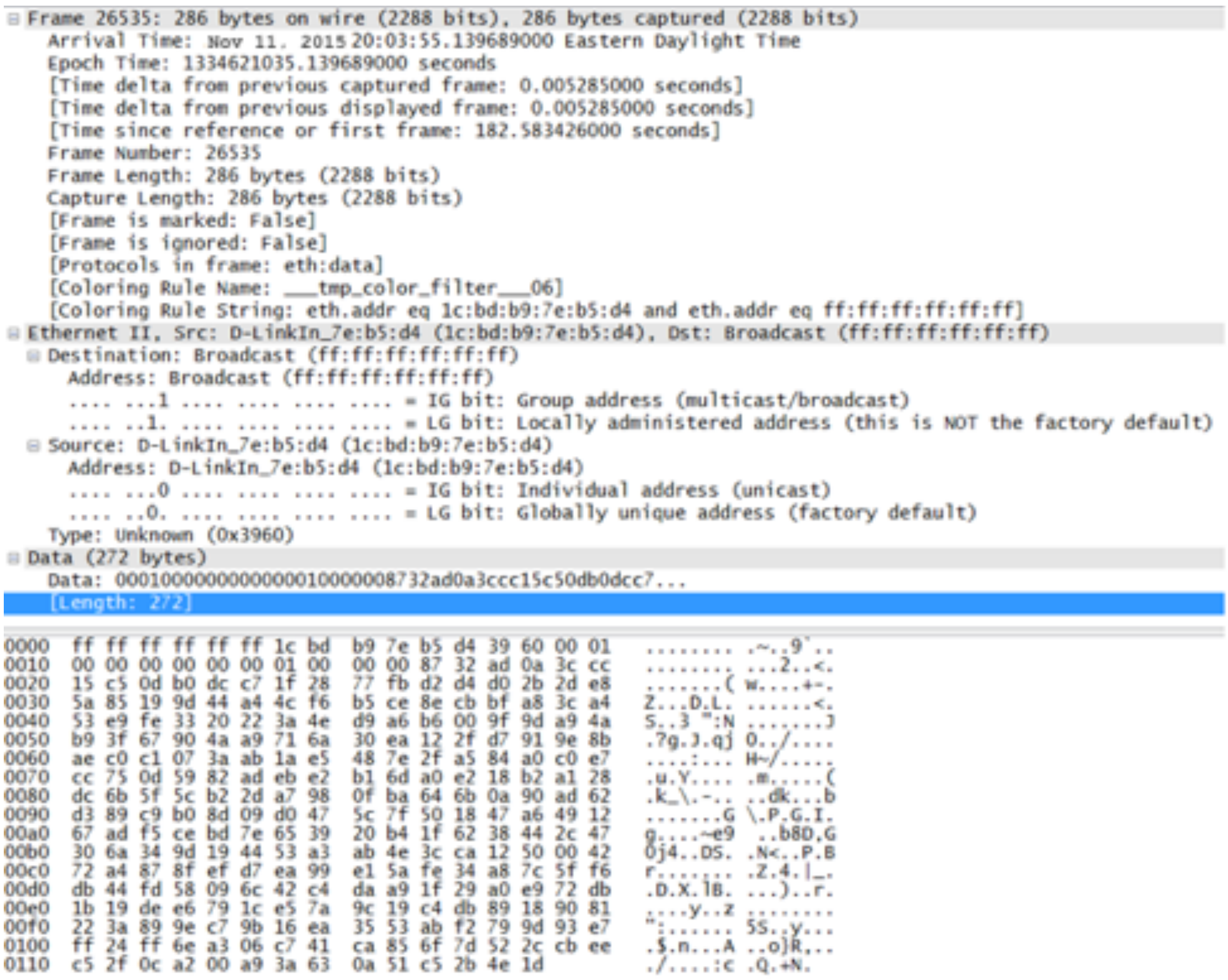


```
⊞ Frame 35757: 1352 bytes on wire (10816 bits), 1352 bytes captured (10816 bits)
⊟ Ethernet II, Src: D-LinkIn_7e:b5:d4 (1c:bd:b9:7e:b5:d4), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ⊟ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
      Address: Broadcast (ff:ff:ff:ff:ff:ff)
      .... ...1 .... .... .... .... = IG bit: Group address (multicast/broadcast)
      .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factory default)
  ⊟ Source: D-LinkIn_7e:b5:d4 (1c:bd:b9:7e:b5:d4)
      Address: D-LinkIn_7e:b5:d4 (1c:bd:b9:7e:b5:d4)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    Type: Unknown (0x3950)
⊟ Data (1338 bytes)
    Data: 800100009d000000000000005b1ed711d2905d00ac1e7d3c...
    [Length: 1338]

0000  ff ff ff ff ff ff 1c bd  b9 7e b5 d4 39 50 80 01   ........ .~..9P..
0010  00 00 9d 00 00 00 00 00  00 00 5b 1e d7 11 d2 90   ........ ..[.....
0020  5d 00 ac 1e 7d 3c 1f 2b  86 06 b0 e2 51 b4 49 a7   ]...}<.+ ....Q.I.
0030  52 81 41 0c 00 30 79 a2  51 4e 37 c9 03 b8 64 73   R.A..0y. QN7...ds
0040  85 2e 7f 73 91 7c 89 2a  72 04 86 01 da b2 6c 3b   ...s.|.* r.....l;
0050  92 5d 47 3b 5b 0b 36 54  fa 00 6b b6 ed a5 f9 4b   .]G;[.6T ..k....K
0060  32 30 54 57 a0 d6 97 74  47 f6 8d d8 16 c5 2c 36   20TW...t G.....,6
0070  af 7f 94 0a f2 a8 37 d5  3a 1e 8b 46 f9 96 b9 54   ......7. :..F...T
0080  10 fe a1 74 20 7c f0 ba  8c b8 80 98 fa 68 30 8f   ...t |.. .....h0.
0090  1c e1 0f b3 35 9b 58 12  3e a1 aa 9e 3b 15 30 be   ....5.X. >...;.0.
00a0  4b f3 5c ba 57 d7 0e 32  bc b3 7f 76 22 27 55 5f   K.\.W..2 ...v"'U_
00b0  2e f0 2d 75 05 3b 0b 69  1e f9 ec ec 71 36 3b 53   ..-u.;.i ....q6;S
00c0  91 f3 d9 47 ab a9 a6 da  9e c5 51 4a 3a 5a 0a 49   ...G.... ..QJ:Z.I
00d0  a0 0e b9 7e 1a 88 46 a6  fc 0b ac 97 8a aa 70 51   ...~..F. ......pQ
00e0  2e 85 f5 5b 07 e8 fc b5  58 e5 15 31 99 ea 71 5b   ...[.... X..1..q[
```

Meanwhile, a user attempts to send a message to a known destination address. Upon successful completion of task, the application lists the address to which the message was delivered to with a message indicating successful or unsuccessful transmission. Correspondingly this information is only available to the sender of the message and hidden from the view of an external observer.

Once a message has been transmitted, an external observer can attempt to analyse the transmitted information by inspecting the packet below. Though it is pretty clear that the destination address is obscured to the external observer, even though our application tells us otherwise; since the message is also encrypted, an eavesdropper cannot perform any man in the middle attacks to steal any crucial piece of information to gain access to the network. Essentially, this means that both the message and the destination have been made anonymous.



```
⊟ Frame 26535: 286 bytes on wire (2288 bits), 286 bytes captured (2288 bits)
    Arrival Time: Nov 11. 2015 20:03:55.139689000 Eastern Daylight Time
    Epoch Time: 1334621035.139689000 seconds
    [Time delta from previous captured frame: 0.005285000 seconds]
    [Time delta from previous displayed frame: 0.005285000 seconds]
    [Time since reference or first frame: 182.583426000 seconds]
    Frame Number: 26535
    Frame Length: 286 bytes (2288 bits)
    Capture Length: 286 bytes (2288 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:data]
    [Coloring Rule Name: __tmp_color_filter___06]
    [Coloring Rule String: eth.addr eq 1c:bd:b9:7e:b5:d4 and eth.addr eq ff:ff:ff:ff:ff:ff]
⊟ Ethernet II, Src: D-LinkIn_7e:b5:d4 (1c:bd:b9:7e:b5:d4), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ⊟ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
      Address: Broadcast (ff:ff:ff:ff:ff:ff)
      .... ...1 .... .... .... .... = IG bit: Group address (multicast/broadcast)
      .... ..1. .... .... .... .... = LG bit: Locally administered address (this is NOT the factory default)
  ⊟ Source: D-LinkIn_7e:b5:d4 (1c:bd:b9:7e:b5:d4)
      Address: D-LinkIn_7e:b5:d4 (1c:bd:b9:7e:b5:d4)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    Type: Unknown (0x3960)
⊟ Data (272 bytes)
    Data: 000100000000000000010000008732ad0a3ccc15c50db0dcc7...
    [Length: 272]

0000  ff ff ff ff ff ff 1c bd  b9 7e b5 d4 39 60 00 01   ........ .~..9`..
0010  00 00 00 00 00 00 01 00  00 00 87 32 ad 0a 3c cc   ........ ...2..<.
0020  15 c5 0d b0 dc c7 1f 28  77 fb d2 d4 d0 2b 2d e8   .......( w....+-.
0030  5a 85 19 9d 44 a4 4c f6  b5 ce 8e cb bf a8 3c a4   Z...D.L. ......<.
0040  53 e9 fe 33 20 22 3a 4e  d9 a6 b6 00 9f 9d a9 4a   S..3 ":N .......J
0050  b9 3f 67 90 4a a9 71 6a  30 ea 12 2f d7 91 9e 8b   .?g.J.qj 0../....
0060  ae c0 c1 07 3a ab 1a e5  48 7e 2f a5 84 a0 c0 e7   ....:... H~/.....
0070  cc 75 0d 59 82 ad eb e2  b1 6d a0 e2 18 b2 a1 28   .u.Y.... .m.....(
0080  dc 6b 5f 5c b2 2d a7 98  0f ba 64 6b 0a 90 ad 62   .k_\.-.. ..dk...b
0090  d3 89 c9 b0 8d 09 d0 47  5c 7f 50 18 47 a6 49 12   .......G \.P.G.I.
00a0  67 ad f5 ce bd 7e 65 39  20 b4 1f 62 38 44 2c 47   g....~e9  ..b8D,G
00b0  30 6a 34 9d 19 44 53 a3  ab 4e 3c ca 12 50 00 42   0j4..DS. .N<..P.B
00c0  72 a4 87 8f ef d7 ea 99  e1 5a fe 34 a8 7c 5f f6   r....... .Z.4.|_.
00d0  db 44 fd 58 09 6c 42 c4  da a9 1f 29 a0 e9 72 db   .D.X.lB. ...)..r.
00e0  1b 19 de e6 79 1c e5 7a  9c 19 c4 db 89 18 90 81   ....y..z ........
00f0  22 3a 89 9e c7 9b 16 ea  35 53 ab f2 79 9d 93 e7   ":...... 5S..y...
0100  ff 24 ff 6e a3 06 c7 41  ca 85 6f 7d 52 2c cb ee   .$.n...A ..o}R,..
0110  c5 2f 0c a2 00 a9 3a 63  0a 51 c5 2b 4e 1d         ./....:c .Q.+N.
```

Conclusion

From our simulations and demonstrations, our routing protocol was able to mask the contents of a transmitted packet, conceal the identity of the users (senders and users) while ensuring that the messages were only sent via authorized nodes. However, the application was plagued by a number of issues during the implementation phase. The complexity of implementing such a protocol is extremely high due to the multiple layers of encryption involved. This proved to be an additional challenge to our routing strategy as RSA algorithms are time consuming and hence, having an impact on the reliability of the network while trying to pick a route.

The Tor network solves this problem by deploying dedicated Tor Routers/ Nodes around the world which act as relay nodes and help the Tor client pick a random path to the destination address. Since the former approach is more feasible for an infrastructure based network, such an approach is not feasible for a mobile ad hoc network due to the dynamic nature and fluctuating topology of the network infrastructure.

Onion Routing (in general), however, is not invulnerable to such attacks. If an attacker has enough data and has the right set of skills and an advanced understanding of statistical analysis techniques, it is possible to analyze usage patterns and hence, make an educated guess about the routing of messages. Likewise, it is definitely possible to perform “exit node sniffing” attacks at either end of the nodes (sender and/or receiver). This form of attack is extremely dangerous since an exit node has complete access to the content being transmitted between the sender and the receiver. Generally, this problem can be made much harder by employing end-to-end encryption.

References

. [1] Tor project: Overview. (2012, February 26). Retrieved from: www.torproject.org/about/overview.html.en

. [2] "Tor: The Second-Generation Onion Router", in Proceedings of the 13th USENIX Security Symposium, August 2004.

. [3] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding Routing Information, Workshop on Information Hiding, Cambridge, UK, May, 1996.

. [4] Yih-Chun Hu, Adrian Perrig, "A Survey of Secure Wireless Ad Hoc Routing," IEEE Security and Privacy, pp. 28-39, May-June, 2004

. [5] Barbeau, M. (2010, January 11). The zone routing protocol and neighbour discovery protocol. Retrieved from: people.scs.carleton.ca/~claurend/Courses/COMP4203/W12/Resources/NDP.pdf

. [6] Z.J. Haas and M.R. Pearlman, ZRP a hybrid framework for routing in ad hoc networks, Ad Hoc Networking (Charles E. Perkins, ed.), Addison-Wesley, 2001, pp. 221-253.

. [7] Laurendeau, C. (2006, April). Wireless networks and protocols: Lecture review. Retrieved from: people.scs.carleton.ca/~claurend/Courses/COMP4203/W12/Resources/ChristineLectRev.pdf

. [8] Dingledine, R., Mathewson, N., & Syverson, P. (2004, August). Tor: The second- generation onion router. Paper presented at Proceedings of the 13th USENIX Security Symposium, San Diego.

. [9] Neal, Harrison (Wikimedia, March 2008) SVG Diagram of the Onion Routing Principal, Online [World Wide Web] Available From: en.wikipedia.org/wiki/File:Onion_diagram.svg