Volume 2, Issue 2, February 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

ANONYMOUS CONNECTIONS and ONION ROUTING

Nilesh Madhukar Patil
Lecturer, I.T.Dept.,
Rajiv Gandhi Institute of Technology,
Mumbai,India.

Chelpa Lingam
Principal, MES's Pillai HOC College of Engg. and Technology,
Rasayani, India.

***Abstract*— Human rights workers, activists, and spies, among other groups and even ordinary citizens, may desire the ability to communicate via the Internet without revealing who they are or what they're doing. This can be accomplished through the user of anonymous network protocols, whose goals are to provide anonymity to the network user. Onion routing is a distributed P2P application that allows two peers to communicate anonymously over the network. The main focus is to have a practical network application allowing two users to have anonymous communication and at the same time be resistant to many network security attacks like denial of service attack, man in the middle attack, replay attack etc. Onion Routing is one such application which enables users to have anonymous communication and yet is so reliable from eavesdroppers and traffic analyzers. The communications in general are bi-directional and real time.**

It first securely establishes the connection. To ensure the security well known networking and public key cryptographic techniques are utilized. Here the identities of the sender and the receiver are kept hidden by an onion structure, which is cryptographically layered data structure that defines the route through the onion routing network. After the route is established by making the entries into the routing table, the data is transmitted over the channel, which is also repeatedly encrypted. Once the data is transferred the connection is destroyed. Using symmetric and asymmetric cryptosystems at different levels enhances further security.

***Keywords*— Onion routing, Crowds, Onion Proxy, dns_server, Onion, RSA, Dijkstra.**

I. OBJECTIVE

A number of solutions have been developed to have secure encrypted communication over the network and they are pretty much resistant to a number of attacks. But one thing these solutions do not offer is anonymous communication [2] i.e., we do not want the third-party to know the person with whom we are communicating. Our objective in this paper is to analyze the concept of onion routing and to implement it for LAN. Onion Routing [1] is a general-purpose infrastructure to support private and anonymous communication [3] over a public network. Preserving privacy not only means hiding messages sent, but also who is talking to whom (Traffic analysis). This paper will also discuss an experiment in which both Onion Routing and Crowds were implemented on a simulated network. They were compared to each other as well as a simple protocol using Dijkstra's algorithm, and overhead and bandwidth measurements were taken to determine performance and usability.

II. INTRODUCTION AND OVERVIEW

Imagine standing in a large, crowded room and you are handed a brown paper cylinder with your name on it. The person who hands it to you tells you to peel the paper with your name on it off of the cylinder to expose a *new* layer with a *new* name on it–your task is to deliver the cylinder to the
person named, tell him to peel that layer of paper off and pass it on to the next person named and tell him to do the same. This goes on until the very center of the cylinder is handed off to the person to whom it is addressed. The idea is that the center of the cylinder contains a message sent to the final recipient by the very first person to hand the cylinder off. But, because the cylinder travelled through so many hands, and along a random path through the crowd, anyone observing the receipt of the final message (or any of the hand-offs at any point along the way, for that matter) has no idea where it came from originally; he or she only saw the final hand-off in a relay of hand-offs. In onion routing, instead of establishing a direct connection between the two hosts that want to communicate, the connection will be routed through a set of routers called onion routers and thereby allow the communication to be anonymous. Every node will only have information about its previous hop and the next hop i.e., the person who he/she is communicating with and the person with whom he/she is supposed to communicate.

The information passed on between the routers is modified accordingly so that no other information is passed on. Thus any router does not have any idea of who is the initiator of the connection neither does he/she will have the information of the destination. Only the last node on the route which establishes a connection with the destination finally has information of the destination. Data appearing at each onion router is different and is padded at different levels to keep the length of the data constant.

Onion routing [5] is basically associated with a set of proxies which help in communication. The initiator first establish an initiating connection with Application Proxy on his/her machine through which the communications are routed to the Onion Proxy which define the route to the destination and construct the onion, the data structure that will be passed between the nodes. Onion Proxy establishes the connection with the Entry Funnel of the first node in the route and passes on the onion to it.

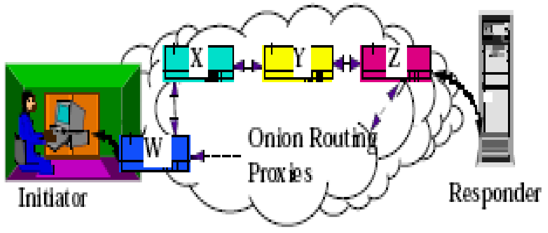


Fig.1 Onion Routing

The Entry Funnel on receiving the onion sends it to its own router, which basically strips of a layer of onion to get details about the next node on the route and accordingly modify the onion and send it to the next router. This way the onion moves in between the routers defined in the route and finally reaches the last node in the route where the onion is passed to the Exit Funnel of that particular node which establishes connection with the final destination for communicating the information. Each layer contains information about the next hop and also a key seed material which basically helps in deriving keys which are used for successive encryption of data to be transmitted.

Once the communication is established between initiator and destination, data is exchanged between the two. The data is transmitted by repeatedly encrypting it with the keys derived from the key seed material. The encryption is done with the key of the last route first and so on and finally with the key of the first router. The encrypted message moves through the nodes removing a layer of encryption at each node and finally the data reaches the destination in plain text. As the data moves through the network, it appears different at different stages and thus prevents snooping by comparing the packets. Every onion layer also has associated with it an expiration time. Thus every router stores a copy of onion till the time expires and hence if a duplicate onion appears within this period, it is ignored and also the onion is ignored if it appears after the expiration time. Thus replay attack is controlled.

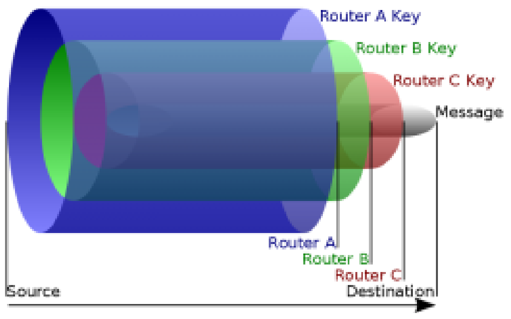


Fig. 2 Structure of Onion

The advantages of onion routing is that it is not necessary to trust each cooperating router; if one or more routers are compromised, anonymous communication can still be achieved. This is because each router in an Onion Router network accepts messages, re-encrypts them, and transmits to another onion router. An attacker with the ability to monitor every onion router in a network might be able to trace the path of a message through the network, but an attacker with more limited capabilities will have difficulty even if he controls one or more onion routers on the message's path.

Although the name related is to something taking place at the network layer of the protocol stack, it should be noted that Onion Routing operates at the Application layer of the Protocol Stack. TCP sockets are used for connection-oriented service.

Crowds [7] is another anonymous protocol, which is similar in operation to Onion Routing. The differences are in the encryption scheme and in the path selection. In Crowds, the paths are chosen dynamically as a message is set, rather than setting up a circuit as Onion Routing does. Cooperating proxies on the network are always chosen randomly on a hop-by-hop basis. When a message is received a proxy will decide to extend the path to a random proxy based on the probability of forwarding, or the proxy will become the last node in the path and communicate with the responder directly. In the figure (Fig. 3), proxy A receives the packet, calculates the probability of forwarding, which is based on a weighted coin flip, and determines that it will send the message on to another proxy. It then sends message to the randomly selected proxy B, which repeats the process and sends the message to proxy C. When proxy C flips the biased coin, it determines that it will become the last proxy in the path and will send the message directly to the receiver. The path is used for a limited amount of time, after which the path must be reformed. The main drawback to Crowds is the lack of receiver anonymity.

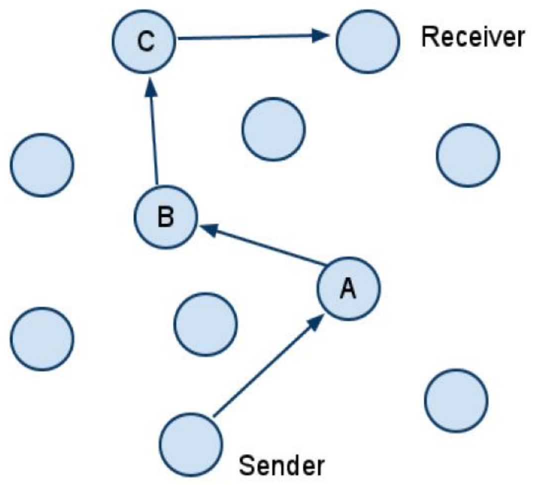


Fig. 3 Crowd Simulation

III. IMPLEMENTATION DETAILS

Our implementation of Onion Routing is in Java and consists of a client who wants to communicate certain messages with a host server and the client has the information about the server but the server does not know anything about client. The following steps are involved in our implementation -

- Network Setup: starts the Onion Router servers and establishes the longstanding connections between Onion Routers
- Starting Services: Starting the Onion Proxy, Application Proxy, host server
- Connection Setup: Client establishes anonymous connection with host server
- Data transfer: Transfer of messages from client to server

A. Network Setup

This stage involves the setting of long standing connections between various onion routers. The connections are pre-established over here to avoid latency that might arise later for setting up the connections between various routers and also to provide anonymity. The path selected for connection is the shortest one found using the Dijkstra's Shortest Path algorithm [10]. Every onion router starts a server and clients to connect with other Onion Routers. To avoid duplication of connections between onion routers, a hierarchical setup is used. All the onion routers have a list of onion routers that are running in the setup. So an entry in the list which is at the bottom just starts the server and waits for client connections from others. An entry above this starts its server and a client establishes a connection with the server below it and so on. Thus any intermediate node starts a server to accept connections from nodes above it and clients which establish connections with the servers of routers below them. There is also a looping in trying to establish the connection i.e., server will run indefinitely. So to make the configuration dynamic, whenever a new router comes in, it is added to the top of the routers list.

The connection setup stage also involves authentication and exchange of keys that will be used for encryption for further communication between the routers. This is implemented by the RSA Key exchange mechanism [4]. So at the end of establishment of connection, key exchange takes place and every two routers have a unique exchange among themselves which is known only to them and the router with which it communicates.

Once the router establishes connection with all the routers in the list, it starts its Entry Funnel server to listen to connections from Onion Proxies to receive onions and Exit Funnel server to listen from its router if it is the last node in the route of connection, so that it can establish the connection with the host server as the need arises.

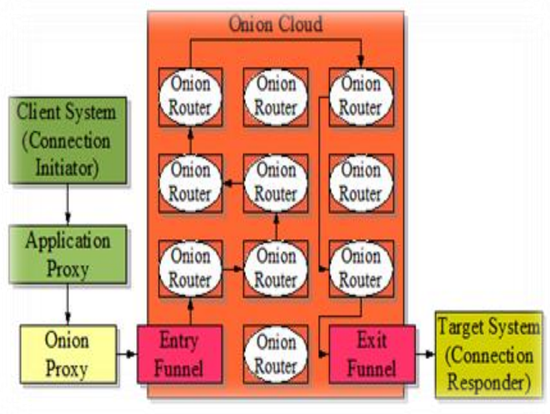


Fig. 4 Communication

B. Starting Services

This stage involves setting up of various proxies and the starting of the host service that offers the services and supports the current protocol. Later the system can be extended to support any basic TCP connection and hence we will be able to implement anonymous communication between any client and any server which runs on the internet.
Client side setup involves starting of Application Proxy and Onion Proxy. Onion Proxy gets the static configuration of all the onion routers which are active.

C. Connection Setup

The user starts the client and gives the details of the server with which it wants to communicate with. The client establishes a connection with the application proxy and sends the relevant information. The application proxy constructs a standard structure out of these details from the client and sends the standard structure to the Onion Proxy. Onion Proxy on receiving the standard structure, randomly picks up a route for communication from the list of nodes and constructs the onion by adding a onion layer for every node through which the onion passes. Having constructed the onion, the onion proxy establishes a connection with the entry funnel of the first node in the route and sends the onion. The entry funnel on receiving the onion transmits it to the onion router at the same node with which it has a pre-established connection. The onion router strips of the first layer to get the details of the key seed material and the next hop in the route. It then pads this stripped off onion with random string at end to maintain the length and send it to the next router. This continues till the onion reaches the last node in the route which realizes that it is the last node and hence the onion is sent to the exit funnel of the corresponding node.

The exit funnel gets the destination details and establishes the connection with the destination. Thus the anonymous connection is established with the destination i.e., a route is defined for the client to communicate with the host server in our case. Each layer stripped off, contains key seed material which is later used to strip off a layer of encryption from the data packets.

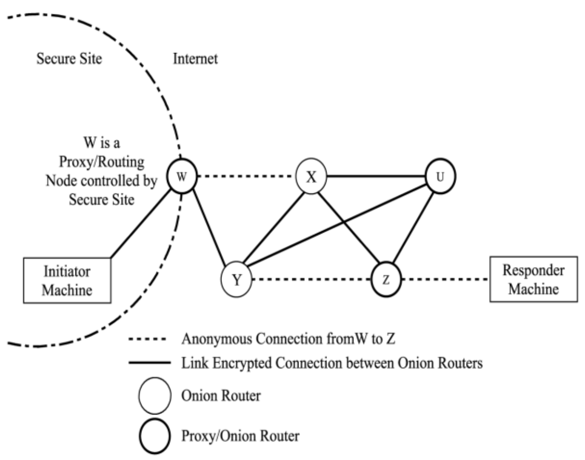

Secure Site
Internet
W is a Proxy/Routing Node controlled by Secure Site
W
X
U
Y
Z
Initiator Machine
Responder Machine
Anonymous Connection fromW to Z
Link Encrypted Connection between Onion Routers
Onion Router
Proxy/Onion Router

Fig. 5 Connection Setup

D. Data Transfer

Once the connection to the host server has been established through the route, the client can communicate with it by sending messages. The message is sent to the onion proxy through the application proxy. Onion Proxy having the knowledge of all the key seed materials encrypts the data of nodes of the route in the reverse order. The cell is passed on to the entry funnel and then to router to different routers in the node (note that at every router, a layer of encryption is removed) and finally reaches the exit funnel of the last node from which the message is sent to the host server. This process continues indefinitely to exchange a large amount of volume.

IV. MODULES DEVELOPED

A. CLIENT

After client is authenticated, it will try to connect with the dns_server. After successful connection is established, dns_server will send a file consisting of IP addresses, host names and port numbers of all the active PC's. Then client will establish connection with the desired server (Destination) and send file to it. After the file is successfully sent, Client will get a notification about it.

B. SERVER

After server is authenticated**,** it will try to connect with the dns_server. After successful connection is established, dns_server will send a file consisting of IP addresses, host names and port numbers of all the active PC's. Then when client tries to connect with it, server may accept or reject. Also, encryption and decryption of data takes place in this class.

C. DNS_SERVER

As soon as the application is started, all the active PC'S will connect with the dns_server who will accept all the connections. It will then send a file consisting of updated IP addresses, host names and port numbers of all this PC's to every other PC.

D. ONION PROXY

Client will send its data to onion proxy. Then onion proxy will form a standard structure (onion) by breaking the data into fixed size packets and encrypting it. It will then forward this structure to other nodes (where decryption takes place) on the path till the node reached is destination.
Note: In this application, every PC can act as dns_server, server (destination) and client.

V. SIMULATION IN PYTHON

Our simulation outputs statistics about the bandwidth and the overhead of each protocol for a certain sized network.
The software, when run will produce various outputs. You will see live graphs presenting simulation [8] [9] data during the run. One graph will be for bandwidth, and the other graph for overhead. The bandwidth graph represents the amount of data sent using a protocol divided by the amount of time it took to send the data. The overhead graph simply represents the amount of time it took to send the data (the bandwidth divisor). Hence, for speed, high bandwidth and low

overhead are desirable. The graphs will look as follows.

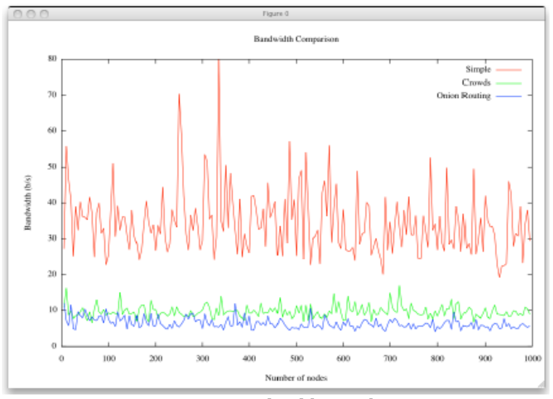


Fig. 6 Bandwidth Graph

The lines at the top of the above graph (Fig. 6) show a better bandwidth measurement, which indicates higher speeds. This graph shows that our simple protocol had very high bandwidth measurements, with Crowds and Onion Routing lingering at much lower speeds. Crowds has slightly better measurements for bandwidth than Onion Routing, which supports the prediction in our hypothesis. It is notable that the difference in performance between Onion Routing and Crowds is not nearly as significant in the jump between speeds using either anonymous protocol and a simple shortest-path protocol. From these measurements we can conclude that the anonymity provided by either Crowds or Onion Routing comes with a high cost, and is probably not for the everyday user.

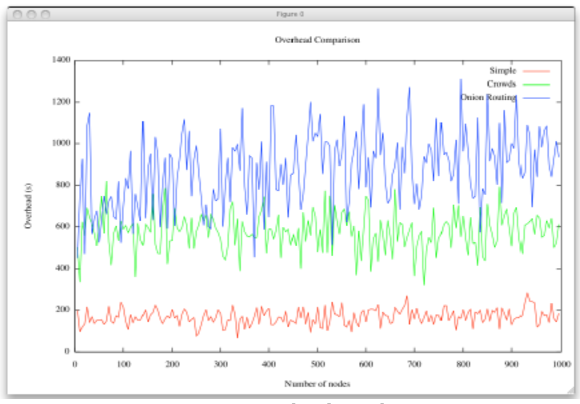


Fig. 7 Overhead Graph

The graph (Fig. 7) above shows overhead measurements for our three implemented protocols. It's clear that Onion Routing has more overhead than Crowds, especially as the network grows larger. Note the significance in time delays between the simple protocol and either anonymous protocol-- a browser running an anonymous protocol might experience two to three times more delay than a standard browser, which would be enough to cause many casual users to abandon hopes of anonymity in exchange for speed.

During the run, certain snapshots of the network will also be created. These snapshots will be saved to the current directory with filenames in the form nodes_10_protocol_simple.png. This denotes that the snapshot is for a network of 10 nodes, and the highlighted path is the path chosen by the simple protocol. The snapshots will have colored circles which represent nodes in the network, and edges between the nodes. The width of the edges represents the latency of the connection between the two nodes, and so, wide edges are slow connections (bad) while thin edges are fast connections (good.) Blue nodes and pink edges denote the path chosen by the current protocol, with the first blue node being the source, and the last blue node being the destination of the connection. The snapshots will look as follows.

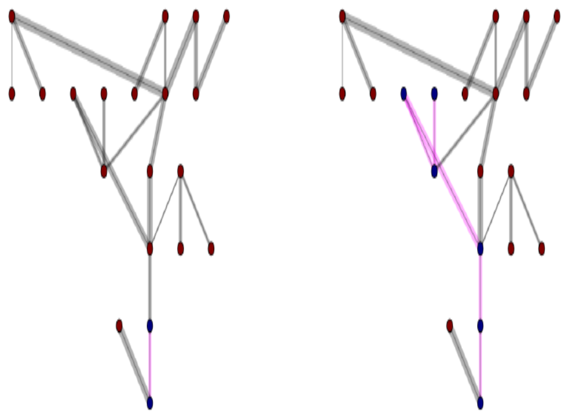


Fig. 8 Network Snapshot in Python

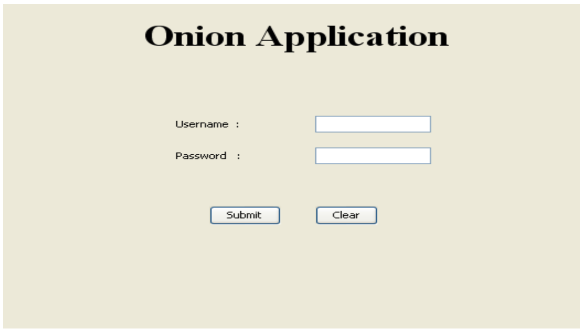
VI. CONCLUSION

Here we presented a protocol called Onion Routing. The purpose of Onion Routing is to protect the anonymity of a user who wants to communicate over a network. In particular, it will hide the destinations of all communications initiated by the user. Any outside observers will not be able to tell whom the user is communicating with and for how long. To achieve this goal, Onion Routing uses Public Key Encryption to put multiple layers of encryption around the original data packet, thus creating an object called an onion. This onion will follow a specific route through the network, and at each route a layer of encryption will be peeled off. Once the onion reaches its destination it will have been reduced to the original data packet. When a router decrypts the onion using its private key it will only get the address of the next router along the path. So no router will ever know the full path that is travelled by the onion. Since no outside observer will be able to follow an onion while it is travelling through the network, the communication is completely anonymous.

The application aimed at is developed as efficiently as possible and is tested in the live local network.

Subsequently we would like to extend this to include all the improvements that have been mentioned in the above discussion so far. Also we would like to incorporate more features to allow communications multiple clients, servers; support various encryption mechanisms and protocols etc. in the coming versions. We would also like to make this more robust so that this can be used as a commercial application.

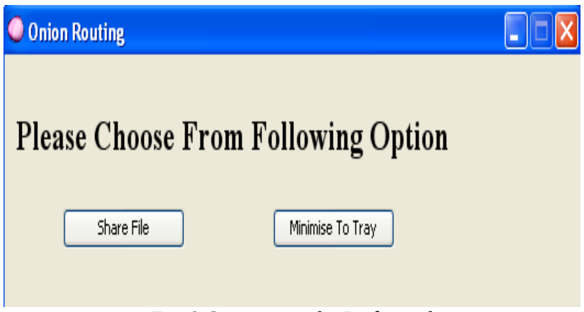
VII. RESULTS


Onion Application

Username :

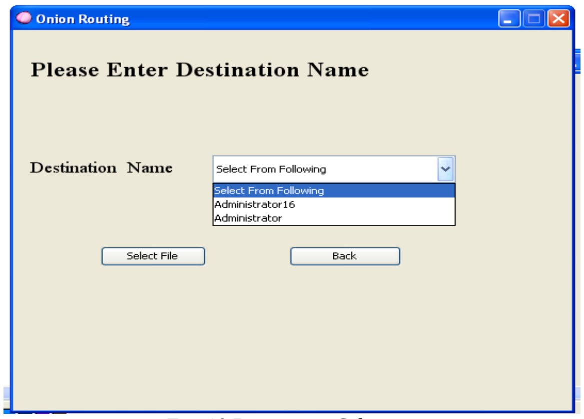
Password :

Fig. 8 Login Screen



Please Choose From Following Option

Fig. 9 Operation to be Performed



Please Enter Destination Name

Destination Name: Select From Following

Select From Following
Administrator16
Administrator

Fig. 10 Destination Selection

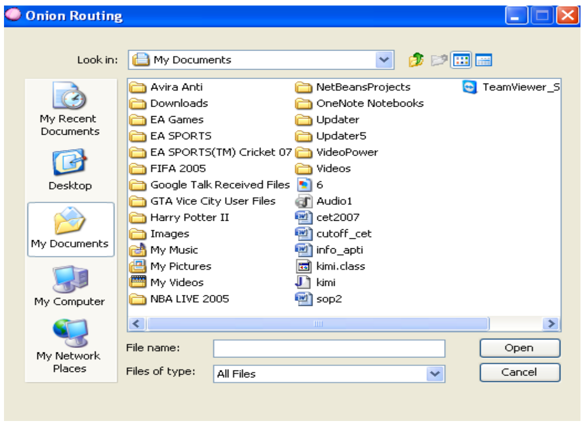
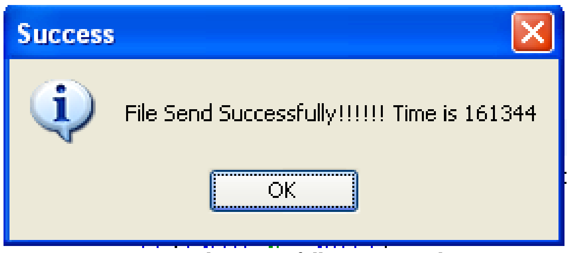


Fig. 11 File to be Shared



File Send Successfully!!!!!! Time is 161344

Fig. 12 File Successfully Transmitted

REFERENCES

[1] Michael G. Reed, Paul F. Syverson, and David M. Goldschlag *Anonymous Communication and Onion Routing*, IEEE Journal on Selected Areas in Communication Special Issue on Copyright and Privacy Protection, 1998.

[2] Roger Dingledine, Nick Mathewson and Paul Syverson, *Tor: The Second-Generation Onion Router*, Proceedings of the 13th USENIX Security Symposium, August 2004.

[3] The Anonymizer, http://www.anonymizer.com.

[4] Wikipedia *RSA*, http://en.wikipedia.org/wiki/RSA

[5] K. Kaviya *Network Security Implementation by Onion Routing*, International Conference on Information and Multimedia Technology, 2009.

[6] Public_key_cryptography http://en.wikipedia.org/wiki/Public_key_cryptography

[7] M. Wright, M. Adler, B.N. Levine and C. Shields, An analysis of the Degradation of Anonymous Protocols. In Proceedings, ISOC Network and Distributed System Security Symposium (NDSS), 2002.

[8] Gnuplot - Command line plotting software http://www.gnuplot.info

[9] Networkx - Python Graph library http://networkx.lanl.gov

[10] www.mathworld.wolfram.com/DijkstrasAlgorithm.html