Network security using onion routing

S.Kiruthika, K.Chandramohan
Gnanamani College of Engineering,
Namakkal, Tamil Nadu,
India.

Abstract-Communication is the god given gift that enables intellectual and cultural exchange and builds up our competence in social behavior. The internet has taken communication to unimaginable attitudes. But many questions arise when we think of security. Is Internet communication private? Most security concerns focuses on preventing eavesdropping that is outsiders listening in on electronic conversions. But encrypted messages can still be tracked revealing who is talking to whom. This tracking is called as traffic analysis and may reveal sensitive information. As an initial step towards ensuring a secured communication we have chosen our paper as Onion Routing in LAN. Onion routing is a flexible communication infrastructure that is resistant to both eavesdropping and traffic analysis.The paper is two fold. First it 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 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 crypto systems at different levels enhances further security. Though we have many alternative solutions such as anonymizer and crowds, Onion routing provides the efficient way of protection, which we have implemented.

Keywords: communication, security, symmetric, asymmetric, protection.

OBJECTIVE:

Our objective in this paper is to analyze the concept of onion routing and to implement it for LAN. Onion Routing is a general-purpose infrastructure to support private and anonymous communication over a public network. Preserving privacy not only means hiding messages sent, but also who is talking to whom (Traffic analysis)..

EXISTING SYSTEM:

The Existing Hiding information into the image is performed either in the spatial or in the transform domain. The idea lies behind modifying the image in an invisible manner. Transform domain embedding is performed by modifying coefficients in the frequency domain, the DCT domain, or in the wavelet domain. In the spatial domain, by modifying the statistics of the image, a spread spectrum based technique, or a simple additive method. On the other hand, quantized projection-based embedding techniques for data hiding have been exploited in the literature under different contexts. In particular, the simplest form is called the low-bit(s) modulation (LBM) where the least significant bit(s) in the quantization of the host signal are replaced by a binary representation of the embedded signal. A class of quantized projection approach has been called the quantization index modulation (QIM) which includes the generalized LBM, spread LBM, spread transform dither modulation STDM, and distortion compensated QIM (DC-QIM). Transparency of the embedded data is guaranteed by using the visual masking model either in the spatial or the transform domain.

Attacks range from simple signal processing operations such as filtering and image coding like JPEG to severe intentional ones such as rotation, scaling, cropping and noise addition. The watermark should be able to survive these attacks. Rotation and scaling attacks have the effect to desynchronize the hidden data. This is due to the fact that changing the image size and/or its orientation, even by slight amount, could dramatically reduce the receiver ability to retrieve the watermark.

Limitations of Existing System:

LSBCoding.Inserting the data in the Least Significant Bits of the carrier files.
Transform domain embedding is performed by modifying coefficients in the frequency domain, the DCT domain or in the wavelet domain.
Spread spectrum technique to hide data inside the carrier. In this method each code bit is represented using multiple code bits and hence the code is spread over large Bandwidth.

Passive Traffic Analysis:

The best protection against traffic analysis lies with obscuring traffic patterns. Making sure that all onions are of the same size, that
timing information on circuits is obfuscated,
and possibly adding noise traffic are all valid methods of protection against traffic analysis. Pipenet provides a very good model for counteracting traffic analysis attacks, however it is an idealistic measure, not attainable in real life.

Active Traffic analysis:

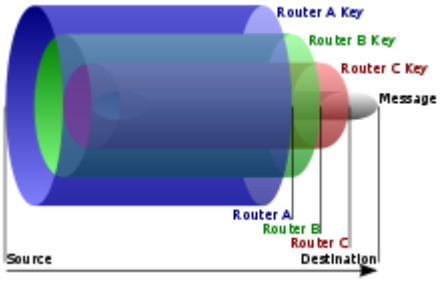
This is the hardest to deal with. Active attacks can include corrupting or delaying traffic between onions to reveal circuit information, as well as setting up a large number of attacker-controlled routers. There are few effective measures against these attacks, however they are quite hard to perform in reality. Again, Pipenet provides an idealistic solution, that is not attainable in real life.

ONION ROUTING:

Onion Routing is flexible, communication infrastructure that is resistant to both eavesdropping and traffic analysis. Onion Routing accomplishes this goal by separating identification from routing. It is a bi-directional, near real-time, and can be used for both connection based and connectionless traffic.

Onion routing has **two main phases**: **Connection setup** and **data movement**. In the phase of connection setup an onion is formed. An onion is a cryptographically layered data structure that defines the route through the onion routing network. This onion is used as the destination address by onion routers in setting up the connection. Onions themselves appear differently to each onion router as well as to network observers. After the connection is established, data is sent, which is repeatedly pre-encrypted with the keys that were carried in the onion. As data moves through the anonymous connection, each onion router removes one layer of encryption, so it finally arrives as plaintext. This layering occurs in the reverse order for data moving backward. In this manner, data appears differently to both onion routers and network observers as it traverses the connection.

Structure of Onion:



The primary innovation in onion routing is the concept of the routing onion. Routing onions are data structures used to create paths through which many messages can be transmitted. To create an onion, the router at the head of a transmission selects a number of onion routers at random and generates a message for each one, providing it with symmetric keys for decrypting messages, and instructing it which router will be next in the path. Each of these messages, and the messages intended for subsequent routers, is encrypted with the corresponding router's public key. This provides a layered structure, in which it is necessary to decrypt all outer layers of the onion in order to reach an inner layer.
The onion metaphor describes the concept of such a data structure. As each router receives the message, it "peels" a layer from the onion by decrypting with its private key, thus revealing the routing instructions meant for that router, along with the encrypted instructions for all of the routers located farther down the path. Due to this arrangement, the full content of an onion can only be revealed if it is transmitted to every router in the path in the order specified by the layering.

Once the path has been specified, it remains active to transmit data for some period of time. While the path is active, the sender can transmit equal-length messages encrypted with the symmetric keys specified in the onion, and they will be delivered along the path. As the message leaves each router, it peels off a layer using the router's symmetric key, and thus is not recognizable as the same message. The last router peels off the last layer and sends the message to the intended recipient.

ONION ROUTING SPECIFICS:

A. Onion Routing Proxies:

A proxy is a transparent service between two applications that would usually make a direct socket connection to each other but cannot. Because it is necessary to bridge between applications and onion routing network, proxies must understand both application protocols and onion routing protocols. Therefore, we modularize the design into components: the application proxy, the onion proxy, and the entry funnel.

B. Phases:

There are four phases in an onion routing system: Network setup, that establishes the long standing connections between the onion routers; connection setup, which establishes anonymous connections through the onion router network; Data movement over an anonymous connection; and the destruction and cleanup of the anonymous connections.

C. Application Proxy:

The interface between an application and the application proxy is application specific. The interface between the application proxy and the onion proxy id defined as follows: For each proxy request, the application proxy first determines if it will handle or deny the request. If accepted, it creates a socket to the onion router well-known port. The application proxy then sends a standard structure to the onion proxy

D. Onion Proxy:

Upon receiving the standard structure, the onion proxy can decide whether to accept or reject the request based on the protocol, destination host, destination port, or the identity of the application proxy. If rejected, it sends an appropriate error code back to the application proxy, closes the socket, and waits for next request. If accepted it proceeds to build the onion and connects to the entry funnel of the first of the last. It next sends the standard structure to the exit funnel over the anonymous connection, and then passes all future data to and from the application proxy and anonymous connection.

E. Onions:

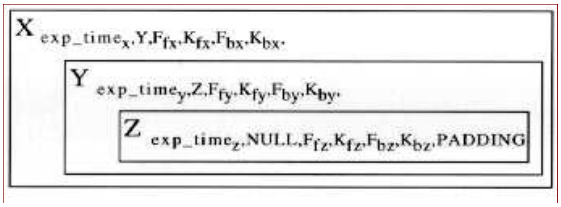


Fig 2.A Forward Onion

The onion data structure is composed of layer upon layer of encryption wrapped around a payload. Leaving aside the shape of the payload at the very center, the basic structure of the onion is based on the route to the responder that is chosen by the initiator's proxy. Based on this route, the initiator's proxy encrypts first for the responder's proxy, then for the preceding node on the route, and so on back to the first routing node to whom he will send the onion.

When the onion is received, each node knows who sent him the onion and to whom he should pass the onion. But, he knows nothing about the other nodes, nor about how many there are in the chain or his place in it(unless he is last). What a node P_x receives looks like this.

{exp_time, next hop, F_f, K_f, F_b, K_b, payload} PK_x

Here PK_x is a public encryption key for routing node P_x, who is assumed to have the corresponding decryption key. The decrypted message contains an expiration time for the onion, the next routing node to which the payload is to be sent, the payload, and two function/key pairs specifying the cryptographic operations and keys to be applied to data that will be sent along the virtual circuit. The forward pair (F_f, K_f) is applied to data moving in the forward direction. (Along the route the onion is travelling) The backward pair (F_b, $K_{b)}$ is applied to data moving in an opposite direction (along the onions reverse route). (If the receiving node is the responder's proxy, then the next hop field is null). For any intermediate routing node the payload will be another onion. The expiration time is used to detect replace, which pairs of compromise nodes could used to try to correlate messages. Each node holds a copy of the onion until exp_time. If he receives another copy of the same onion within that time he simply ignores it. And, if he receives an onion that has expired, he ignores that as well.

Notice that at each hop the onion shrinks as layer is peeled off. To avoid compromised nodes inferring route information from this monotonically diminishing size, a random bit string the size of the peeled off layer is appended to the end of the payload before forwarding. No proxy except the last will know how much of the payload he receives is such padding because he won't know where he is in the chain. He simply 'decrypts' the padding along with the rest of the onion. Even a constant size onion might be traced unless all onions are same size, so we fix the size of the onion. To maintain this constant size to hide length of the chain from the responder's proxy, the initiator' s proxy will pad the central payload according to the size of the onion, i.e., the number of hops. So, when any onion arrives at the

responder's proxy it will always have the same amount of padding, either added initially or en route.

CREATING THE CIRCUIT:

The goal in sending the onion is to produce virtual circuit within link encrypted connections already running between routing nodes. An onion occurs as the data field in one of the presently described message's. Such messages contain a circuit identifier, a command (create, destroy and data), and data. Any other commands considered an error, and the node who receives such a message ignores that message except to return a destroy command back through that virtual circuit. The create command accompanies an onion. When a node receives a create command along with an onion, he chooses a virtual circuit identifier and sends another create message containing this identifier to the next node and the onion (padded with his layer peeled off). He also stores the virtual circuit identifier he received and virtual circuit identifier he sent as a pair. Until the circuit is destroyed, whenever he receives data on the one connection he sends it off on the other. He applies the forward cryptographic function and key (obtained from the onion) to data moving in the forward direction (along the route the onion traveled) and the backward cryptographic function and key to data moving in the opposite direction (along the onion's reverse route). The virtual circuit established by the onion in Fig 2 is illustrated in fig 3

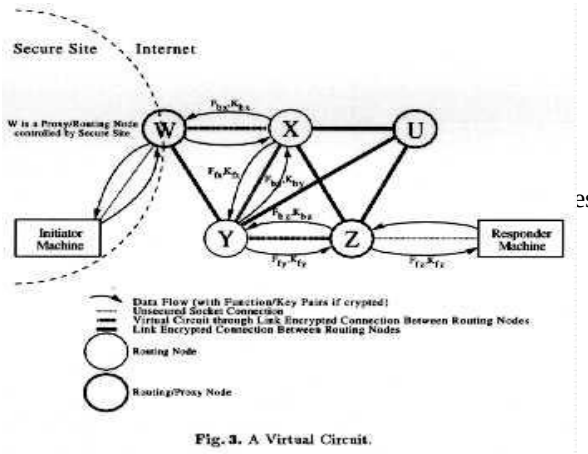


Fig. 3. A Virtual Circuit.

Module I:

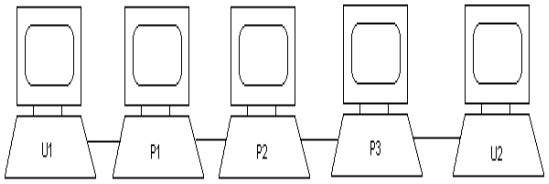
The first step to implement onion routing is to **design the encryption algorithms.** Different algorithms are used for connection establishment and data transfer.

For connection establishment **RSA algorithm** is used. RSA algorithm is the standard public key cryptographic algorithm and the cipher text cannot be decrypted easily because, In RSA algorithm the process of decryption is not an inverse process of encryption. The encryption and decryption keys are generated using random prime numbers. While sending the onion keys are generated every time using system time .So it is difficult to predict the keys.

For **transferring the data RC4 algorithm is used**. This is a variable key size algorithm.

Module II:

The second step is the **connection establishment** .To explain the connection establishment let as consider the following **local area network**.



Here **U1 & U2** are **source and destination** and **P1 , P2 & P3 are proxies**. The proxies are assumed to be consisting of number of users. U1 is the user of proxy p1 and U2 is the user of proxy p2.

The connection is established using **TCP socket connections** .Inorder to perform the private and anonymous communication the route to be found first.ie., the path that is to be followed from sender to receiver and the addresses of the proxies through which they pass. Then an onion is formed in the initiating proxy P1 using the addresses of the proxies. This onion is then passed to the next proxy along the route. One layer of onion is decrypted at each intermediate proxy and appropriate entries such as IP address; keys and functions for decrypting the data will be made in to the routing table. In order to maintain the size of the onion, a random string is appended at the end of the payload so that it appears to be of constant size throughout the transfer.

Module III:

After the connection is established the **data is passed over the connection**, which is also encrypted in the form of the onion. Similarly the data is sent from the receiver to sender by encrypting at each of the intermediate proxies and it is finally decrypted at the initiating proxy and finally given as a plain text to the sender

APPLICATIONS:

A. VPN's (Virtual Private Network)
B. Anonymous Chatting
C. Anonymous Cash

D. Web Browsing

E. Onion Routing provides a new way of approach to prevent the traffic analysis and eaves dropping.

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.

Reference:

Murdoch Steven J, DanezisG.low-cost traffic analysis TOR In: IEEE symposium on security and privacy, May 2005.

Owens Mark. A discussion of covert channels and steganography

The art of computer virus research and defense.Addissm Wesley; 2005