TOR-CLI-Routing Project

Authors: Tusshar Rana 100426359

Alex Babenko 100442814

Date: 21 November 2024

Abstract

The TOR-CLI-Routing Project aims to establish a secure communication channel using Tor for anonymity and RSA encryption for security. This paper details the setup, implementation, and benefits of using Tor and RSA in a client-server architecture. The project demonstrates how to route traffic through Tor and encrypt messages to ensure privacy and security.

Introduction

In today's digital age, secure communication is paramount. The TOR-CLI-Routing Project addresses this need by combining the anonymity provided by Tor with the robust security of RSA encryption. This project involves setting up a client-server architecture where the client routes its traffic through Tor and encrypts messages using the server's public key. The server decrypts these messages using its private key. This paper provides a comprehensive breakdown of the code, the rationale behind the chosen technologies, and the potential applications of this setup.

TOR

Tor (The Onion Router) is a free, open-source software that provides anonymous communication on the internet by routing traffic through a global network of volunteer-operated servers. Tor's unique ability to anonymize data flow makes it valuable for individuals seeking privacy, such as journalists, activists, or anyone navigating restrictive regimes. Its key purpose is to protect against network surveillance and traffic analysis, which can expose users' identities and online activities (Tor Project, 2024; IdentityIQ, 2023).

Components of Tor

1. **Tor Browser**
   The most common way users access the Tor network is through the Tor Browser, which is a modified version of Firefox. This browser is built to prioritize privacy, blocking tracking technologies and ensuring that users remain anonymous while browsing. It also prevents sites from seeing users' physical locations (Tor Project, 2024).
2. **Tor Network**
   The Tor network is a decentralized collection of thousands of volunteer-operated relays. These relays work together to maintain the anonymity of users by obfuscating the path that data travels from sender to receiver (ITP, 2023).
3. **Nyx**

Nyx is a command-line application for monitoring Tor, offering insights into network activity, bandwidth usage, and more. It provides detailed statistics for relay operators and advanced users, displaying circuits, streams, and resource consumption. Nyx helps manage and visualize Tor's operations, making troubleshooting and performance monitoring easier (Tor Project, n.d.).

Benefits of Using Tor

1. **Anonymity**
   Tor is designed to provide anonymity by routing traffic through multiple relays and concealing users' IP addresses. This is critical for users who wish to avoid surveillance, such as journalists working in authoritarian regimes or activists in need of secure communication channels (Tor Project, 2024).
2. **Bypassing Censorship**
   Another significant advantage is Tor's ability to bypass censorship. Many users in countries with restrictive internet policies rely on Tor to access blocked websites, allowing them to circumvent government restrictions and communicate freely .
3. **Enhanced Security**
   Tor's multi-layer encryption provides an additional layer of security against surveillance. However, Tor does not encrypt the last packet that will be send from the Exit node/router, while Tor enhances privacy, it does not guarantee total protection from all forms of cyber threats, such as malware or phishing attacks (IdentityIQ, 2023).

Challenges and Drawbacks

1. **Slower Connection Speeds**
   Due to the multiple layers of encryption and routing through several relays, Tor users

often experience slower browsing speeds compared to conventional internet use. This trade-off is the result of prioritizing privacy over performance (IdentityIQ, 2023).

2. **Potential for Malicious Exit Nodes**
   While most nodes are run by volunteers committed to protecting privacy, there is always the risk of malicious exit nodes, which can potentially intercept unencrypted traffic. This is why it's important to use HTTPS whenever possible when using Tor (IdentityIQ, 2023).
3. **Potential for All the Nodes to be Malicious**
   Although Tor is designed to ensure privacy and anonymity by routing traffic through a network of relays, there is a theoretical risk that all of the nodes in the network could be compromised or malicious. Since Tor relies on the trustworthiness of individual volunteer-run nodes (entry, relay, and exit nodes), it is possible that an adversary with control over multiple nodes could compromise user privacy by tracking or even altering traffic

RSA Encryption

RSA encryption is a widely implemented public-key cryptosystem that enables secure communication by using two keys: a **public key** for encryption and a **private key** for decryption. The system ensures that even if an attacker intercepts the encrypted message, they cannot decrypt it without the corresponding private key (Schneier, 2015).

How RSA Works Without Equations:

1. **Key Generation**: RSA generates two keys—a public key, which is shared and used by senders to encrypt messages, and a private key, kept secret by the receiver to decrypt incoming messages. This system ensures that unauthorized users cannot access the original message, even if they know the public key (Rivest, Shamir, & Adleman, 1978).
2. **Encryption**: In the encryption process, the message is encrypted using the recipient's public key, meaning only the holder of the private key can decrypt it. This method ensures that even if the communication is intercepted, the content remains unreadable (Schneier, 2015).
3. **Decryption**: The recipient uses the private key to decrypt the message, ensuring that the intended party can access the information. The security of RSA lies in the fact that it's computationally infeasible to derive the private key from the public key alone (Anderson, 2020).
4.

How Tor works

Ask Tor Directory :

When a client initiates a connection over the Tor network, it first contacts a trusted Tor directory. The directory provides a list of available relays along with important metadata, which includes public keys, location, speed, and bandwidth of each relay. This metadata allows the client to make informed decisions when building a secure communication circuit(Tor Stack Exchange, 2020a).

1. **Selecting Relays:** The client uses the metadata from the Tor directory to select three relays that will form the circuit: a Guard node (the entry relay), a Middle node (the intermediary relay), and an Exit node (the final relay). The client chooses these nodes based on factors such as geographical location, speed, and bandwidth. At this stage, the client only decides on the nodes and does not yet send any data(Tor Stack Exchange, 2020a).
2. **Building the Circuit:** The client starts by establishing an encrypted connection with the Guard node. To do this, the client uses the public key of the Guard node, obtained from the Tor directory, to encrypt the initial message that will initiate the session. This process involves the following steps:

a. The client encrypts a message with the **Guard node's public key** (PublicKey1) and sends it to the Guard node. b. The Guard node responds with an encrypted message containing a **session key** (SessionKey1), which will be used to secure the communication between the client and the Guard node.

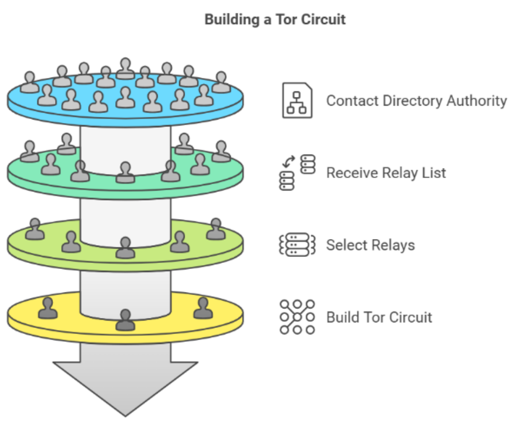
This exchange ensures that any data sent between the client and the Guard node is encrypted using SessionKey1.

3. **Extending the Circuit:** After establishing a secure connection with the Guard node, the client initiates a connection with the Middle node. The client sends the message to the Guard node, which forwards it to the Middle node. However, this message is doubly encrypted—first with SessionKey1 and then with the **Middle node's public key** (PublicKey2). This ensures that the Guard node can only decrypt its portion of the message (encrypted with SessionKey1), and the Middle node can decrypt the message intended for it (encrypted with PublicKey2) (Tor Stack Exchange, 2020a).

a. The client sends a message encrypted with SessionKey1 and PublicKey2. b. The Middle node responds by sending a new **session key** (SessionKey2), used for the communication between the client and the Middle node. The response is similarly encrypted and passes through the Guard node.

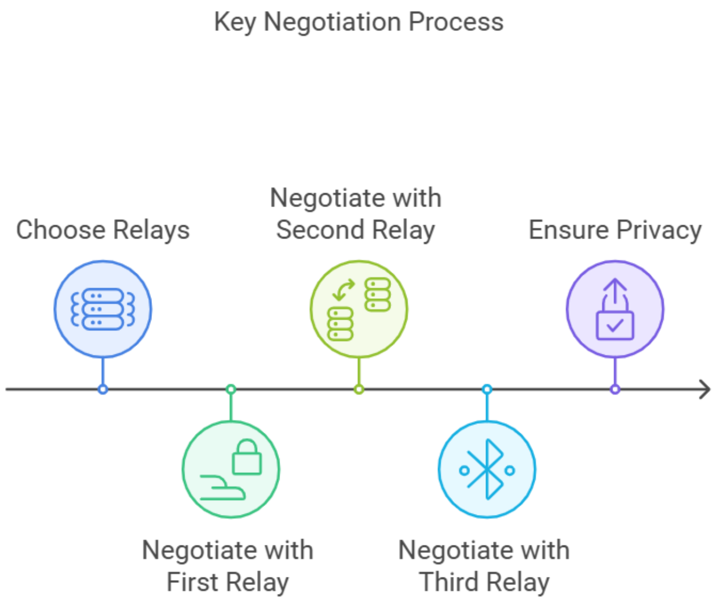
4. **Completing the Circuit:** This process repeats for the Exit node, where the client uses the **Exit node's public key** (PublicKey3) to establish a session key (SessionKey3). All messages between the client and the Exit node pass through the Guard and Middle nodes, ensuring each layer of encryption protects the content as it passes through the network(Tor Stack Exchange, 2020b).

Once the session keys for each of the three relays (Guard, Middle, and Exit) are established, the circuit is complete, and the client can securely transmit data. Each relay can only decrypt the portion of the message intended for it, while the overall communication remains fully encrypted through the layered encryption approach, often described as "onion encryption" (Tor Stack Exchange, 2020a; Tor Stack Exchange, 2020b).



Negotiate Keys :

1. **Selection of Relays**: The client selects three relays to form a circuit.
2. **Layered Key Negotiation**:
   - **First Relay**: The client negotiates a shared secret (symmetric key) with the first relay.
   - **Second Relay**: The client then negotiates a shared secret with the second relay through the first relay.
   - **Third Relay**: Finally, the client negotiates a shared secret with the third relay through the first and second relays.
3. **Onion Encryption:** This process creates multiple layers of encryption, similar to an onion. Each relay only decrypts its layer, knowing only its immediate neighbors, not the entire path. This enhances privacy and security (Tor Project, 2023).


Key Negotiation Process
Choose Relays
Negotiate with Second Relay
Ensure Privacy
Negotiate with First Relay
Negotiate with Third Relay

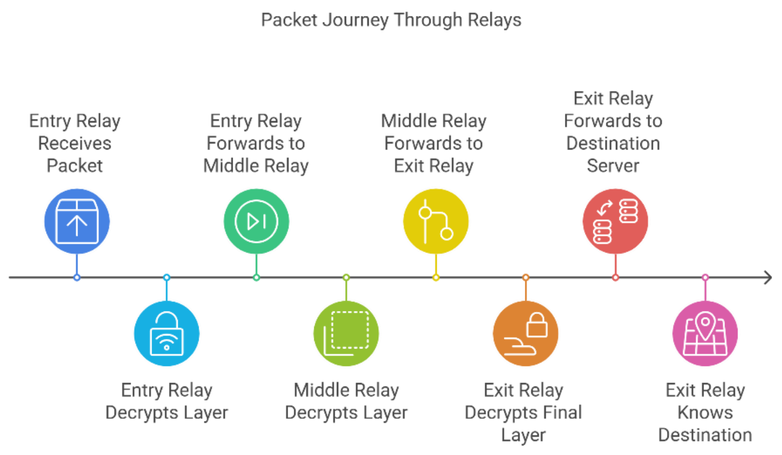
Encrypt a Packet :

1. **Initial Packet Preparation**:
   - The client prepares a packet that includes the source address (the client's address) and the destination address (the server's address).
2. **First Layer of Encryption (Exit Relay)**:
   - The client encrypts the entire packet (including source and destination) using the key for the third (exit) relay. This layer ensures that only the exit relay can decrypt this part and see the destination address.
3. **Second Layer of Encryption (Middle Relay)**:
   - The client takes the already encrypted packet and encrypts it again using the key for the second (middle) relay. This layer ensures that the middle relay can only see the next hop (exit relay) but not the final destination.
4. **Third Layer of Encryption (Entry Relay)**:
   - Finally, the client encrypts the packet once more using the key for the first (entry) relay. This layer ensures that the entry relay can only see the next hop (middle relay) but not the final destination or the source.
5. **Sending the Packet**:
   - The fully encrypted packet is sent to the first (entry) relay.
6. **Decryption by Relays**:
   - **Entry Relay**: The entry relay decrypts the outermost layer, revealing the next relay (middle relay) and forwards the packet.
   - **Middle Relay**: The middle relay decrypts the next layer, revealing the exit relay and forwards the packet.
   - **Exit Relay**: The exit relay decrypts the final layer, revealing the destination server and forwards the packet to the server.

How the Packet Would Go (Client -> Server) :

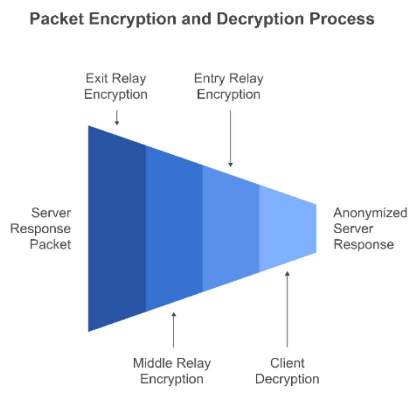
- The encrypted packet first goes to the entry relay, which decrypts one layer. This reveals the next relay (middle relay) and the partially decrypted packet, which still contains the source and destination addresses.
- The entry relay forwards the packet to the middle relay.
- The middle relay decrypts another layer, revealing the next relay (exit relay) and the further decrypted packet.
- The middle relay forwards the packet to the exit relay.
- The exit relay decrypts the last layer, revealing the destination server and the original packet with the source and destination addresses.

- The exit relay forwards the packet to the destination server.
- Only the exit relay knows the final destination (e.g., a website), but it doesn't know who the client is, as that information is hidden by the encryption layers (Tor Project, 2023).


Packet Journey Through Relays
Entry Relay Receives Packet
Entry Relay Decrypts Layer
Entry Relay Forwards to Middle Relay
Middle Relay Decrypts Layer
Middle Relay Forwards to Exit Relay
Exit Relay Decrypts Final Layer
Exit Relay Forwards to Destination Server
Exit Relay Knows Destination

How the Packet Would Go Back (Server -> Client) :

- When the server responds, the packet takes the same path back through the relays, but in reverse.
- The packet is encrypted by the exit relay, then sent to the middle relay.
- The middle relay adds a layer of encryption and forwards it to the entry relay.
- The entry relay adds the final layer of encryption and sends the fully encrypted packet to the client.
- The client then decrypts all the layers to read the server's response.
- This ensures that the server doesn't know who the client is, maintaining the anonymity (YouTube, 2023).



Step	**Source Address**	**Destination Address**	**How the Packet Knows Where to Travel**
Client to Entry Relay	Client	Entry Relay	Client encrypts packet with Entry Relay's key and sends it to Entry Relay.
Entry Relay to Middle Relay	Entry Relay	Middle Relay	Entry Relay decrypts one layer, revealing Middle Relay's address, and forwards the packet.
Middle Relay to Exit Relay	Middle Relay	Exit Relay	Middle Relay decrypts another layer, revealing Exit Relay's address, and forwards the packet.
Exit Relay to Destination Server	Exit Relay	Destination Server	Exit Relay decrypts the final layer, revealing Destination Server's address, and forwards the packet.
Server to Exit Relay	Server	Exit Relay	Server responds and sends the packet to Exit Relay.
Exit Relay to Middle Relay	Exit Relay	Middle Relay	Exit Relay encrypts the packet and uses its routing table to send it to Middle Relay.
Middle Relay to Entry Relay	Middle Relay	Entry Relay	Middle Relay adds a layer of encryption and uses its routing table to send it to Entry Relay.
Entry Relay to Client	Entry Relay	Client	Entry Relay adds the final layer of encryption and uses its routing table to send it to Client.

Conclusion

The TOR-CLI-Routing Project demonstrates the effective use of Tor and RSA encryption to establish a secure and anonymous communication channel. By routing traffic through Tor and encrypting messages with RSA, the project ensures both privacy and security. This setup can be applied to various scenarios where secure and anonymous communication is essential. Future work could involve adding more features, such as message integrity checks and support for additional encryption

References:

Schneier, B. (2015). *Applied Cryptography: Protocols, Algorithms, and Source Code in C* (2nd ed.). Wiley.

Rivest, R., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*.

Anderson, R. (2020). *Security Engineering: A Guide to Building Dependable Distributed Systems* (3rd ed.). Wiley.

IdentityIQ. (2023). What is Tor and how does onion routing work? Retrieved from www.identityiq.com

Tor Project. (2024). The Tor Project. Retrieved from www.torproject.org

ITP. (2023). Demystifying the dark web: An introduction to Tor and onion routing. https://itp.nyu.edu/networks/explanations/demystifying-the-dark-web-an-introduction-to-tor-and-onion-routing/

YouTube. (2023). *How Tor works.* [Video]. YouTube. https://www.youtube.com/watch?v=gIkzx7-s2RU

Tor Project. (n.d.). *Nyx - Terminal (command-line) status monitor for Tor*. Retrieved from https://nyx.torproject.org

Dingledine, R., Mathewson, N., & Syverson, P. (2004). *Tor: The second-generation onion router*. The Tor Project. https://svn-archive.torproject.org/svn/projects/design-paper/tor-design.pdf

Tor Stack Exchange. (2020a). Differences between onion routing public key and Tor circuit session key. Stack Exchange. https://tor.stackexchange.com/questions/19256/differences-between-onion-routing-public-key-and-tor-circuit-session-key

Tor Stack Exchange. (2020b). Relay cell size moving backward. Stack Exchange. https://tor.stackexchange.com/questions/19484/relay-cell-size-moving-backward/19493#19493