P2P Based Secured Communication Mechanism

***Abstract*—Peer-to-peer (P2P) communication provides a method for exchanging data directly without a central authority, however, it creates several challenges with regards to authentication, key management, and privacy. The paper presents a novel approach to secure and anonymous P2P communications leveraging the Tor network, and hybrid cryptographic techniques.**

P2P architecture separates control plane functions (registration, authentication, lookup) from data plane communications enabling peer-to-peer (P2P) direct encrypted messaging through Tor hidden services. The proposed architecture employs Ed25519 for identity and authentication, X25519 for ephemeral key exchange, and AES-256-GCM for secure message transport providing confidentiality, integrity, mutual authentication, forward secrecy and replay protection. The proposed architecture was validated through experimental evaluations by performing a VirtualBox-based simulation demonstrating successful secure communications, reduced reliance on server trust and good privacy preservation from various common network threats.

***Index Terms*—Peer-to-Peer Communication, Tor Network, Anonymous Communication, Cryptography, Key Exchange, Mutual Authentication, End-to-End Encryption, Network Security**

I. Introduction

P2P communication is an important technology. A P2P service can provide protection against a variety of threats. RE, P2P systems provide a large amount of redundancy and fault tolerant operations. These services use large numbers of redundant servers to provide adequate levels of service, low costs, and the ability to provide service even under adverse conditions. By providing redundancy and allowing devices to act as both clients and servers, P2P systems provide improved scalability. When combined with other technologies (such as Tor), P2P systems also provide improved privacy.

P2P also has challenges because there are no centralised authorities or control mechanisms. The following are examples of the challenges associated with P2P systems. Authentication of peers; establishing trust for peers; securely exchanging keys; discovering peer endpoints; and providing protection against interception of messages.

The challenges of P2P systems are solved through the use of cryptographic mechanisms and authentication and authorisation procedures, which may include using symmetric and asymmetric encryption, hashing and digital signatures. In the example of a system described herein, the sending party authenticates to a server, receives the receiving party's endpoint information and public key, establishes a connection using the Tor anonymisation network, performs an authenticated key exchange and then begins encrypted communications with the receiving party. It should be noted that the receiving party will be functioning as a hidden service on the Tor network so that the true IP address of the receiver remains hidden.

This research aims to provide an effective way for users to communicate through use of a P2P model while maintaining their anonymity. The benefits of this model will include minimizing dependence upon centralized authorities, providing protection for information transmitted over the network against certain types of common threats (for example, eavesdropping, man-in-the-middle attacks, tampering and replay attacks), and allowing users to share files between one another privately.

II. Techniques Used

A. Cryptographic Techniques

- **Asymmetric Key Cryptography**: Uses public and private key pairs for encryption, decryption, authentication, and key exchange.
- **Hashing**: Converts readable input into a fixed-length output and is used for identity representation and integrity support.
- **Digital Signature**: Uses a private key to authenticate a sender and to verify message integrity.

B. Authentication and Authorization Methods

- **Certificate-Based Authentication**: Uses digital certificates to establish trust.
- **Token-Based Authentication**: Grants access through short-lived tokens.
- **Reputation-Based Authentication**: Assigns trust scores based on peer behavior.
- **DPEM (Dedicated Private Key Exchange Mechanism)**: A private-key exchange approach for controlled trust establishment.

III. Literature Review

Traditionally, peer-to-peer (P2P) communicative systems have employed a combination of cryptography, authentication, and protocol-level security mechanisms in order to avoid being intercepted or impersonated. The initial research into secure computer networks revealed that you cannot trust a particular connection medium and, thus, every connection needs to be protected by means of encryption, key management, and security protocols. [10]

The two main types of crypto techniques are symmetric and asymmetric. One of the purposes of symmetric encryption techniques (AES for example) are for bulk data transfer and efficiency. On the contrary, asymmetric crypto techniques are designed more for secure key exchange or other mechanisms of authentication. [2], [14].

Research indicates that AES provides better performance than earlier ciphers such as DES when considered in terms of security as well as efficiency [13],[15]. Modern systems

therefore typically use a hybrid encryption cryptosystem in which asymmetric encryption is employed for authentication and key distribution/ exchange, while symmetric encryption will provide confidentiality for session data [5]. The need for secure authentication is very important in distributed systems. Traditional realms of secure authentication methods include passwords, hardware tokens, and biometric identifiers using the categories of "something you know, something you have, or something you are" [6]. However, these methods fall short of providing secure authentication in a decentralized manner, and are limited to a few forms of attack such as replay attacks and impersonating. There have been several published studies on the security of authentication in an IoT context; to confirm their research findings, these papers have determined that true secure authentication requires mutual authentication, lightweight protocols, and protection from replay and man-in-the-middle (MITM) attacks [4],[7].

It has been well documented that public-key cryptography is an essential component of safe communications when dealing with untrusted parties. Many researchers have reported that the level of assurance for actual security is dependent upon strong cryptographic assumptions and efficient implementation [9]. There are many examples of complex advanced cryptographic constructs which highlight the evolution and complexity of asymmetric cryptographic designs, such as hidden monomial cryptosystems [8]. Such examples illustrate the need for a careful selection of cryptography primitives that are both secure and have been thoroughly analysed and vetted.

P2P communications offer many advantages over centralized systems, particularly with respect to privacy and decentralisation. P2P architectures eliminate the reliance on a central message relay, thus improving scalability and fault tolerance. However, P2P systems also present some unique challenges for endpoints including discovery, establishing trust, and finding secure means of communicating between peers. Much of the research on secure communication has focused on separating the control/management plane from the data exchange plane; where authentication and coordination are centralised, and the actual data is exchanged directly between the two peers. [1].

The second major area of focus within the realm of communication systems has to do with how efficient they are. Research on network packet compression has demonstrated that by reducing transmission overhead, packet compression can enhance performance, particularly in situations where available bandwidth is limited [3]. Furthermore, in modern communications systems, header compression techniques are also necessary due to the amount of metadata associated with packets being greater than the amount of data contained in the packet [12]. These studies illustrate that there is a considerable need for developing lightweight and efficient protocol designs for secure peer-to-peer (P2P) systems, in addition to providing cryptographic mechanisms for secure endpoints. Lightweight security mechanisms will also be very beneficial to both IoT and distributed environments. The importance of efficiently balanced security and computational efficiency in authentication protocols has been emphasized in previous studies, particularly for devices with limited resources [4], [16]. Using efficient hashing and encryption-based techniques provides an advantage over other methods by decreasing processing overhead while providing security guarantees.

Usability and human factors further impact on how users authenticate to a system. Research has shown that users may find they are unable to comply with an overly complex authentication system and this may result in the use of alternative means, reducing the security of the overall system [11]. Therefore, when developing secure systems, usability must be included in the development along with the strength of cryptography. The literature reviewed also indicates that a combination of cryptographic techniques, authentication mechanisms, and well-designed protocols are essential for achieving a secure communication. It is essential to include hybrid cryptography, mutual authentication, replay protection, and lightweight protocols when designing modern secure systems. The above principles are the basis of the proposed P2P communication framework; a framework that integrates the security of cryptography with the key attributes of decentralized and privacy preserving communication.

IV. Methodology

A. Introduction

This chapter describes the methodology adopted for the proposed anonymous peer-to-peer (P2P) communication system. The system is implemented through three functional entities: the Sender client, the Receiver client, and the Server. The Sender authenticates to the server, performs receiver lookup, retrieves the receiver's public key, establishes a Tor-based connection, completes an authenticated key exchange, and transmits encrypted messages. The Receiver creates a Tor hidden service, registers its endpoint with the server, accepts incoming connections, and decrypts messages locally. The Server functions as a directory and authentication authority only; it stores hashed user identifiers, public keys, endpoints, tokens, and nonces, but it does not store or relay message content. This behaviour is consistent with the client and server code structure and with the runtime logs produced during execution.

The methodology follows a control-plane and data-plane separation model. The control plane is handled by the server and is responsible for registration, authentication, and lookup. The data plane is established after lookup and consists of direct encrypted communication between the Sender and Receiver over the Tor network. This separation limits the server's role to coordination functions and keeps the message channel outside the server's reach.

B. System Design Approach

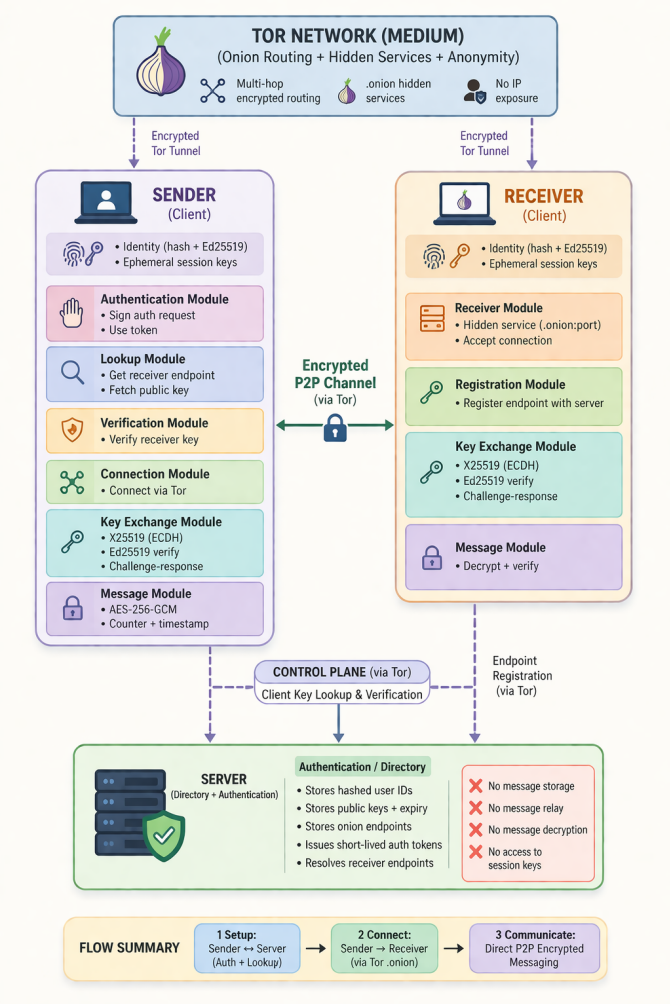

Fig. 1. System Architecture of the Tor-Based Anonymous P2P Communication System

The system is designed as a modular architecture in which each component has a clearly defined responsibility. The Sender client handles authentication, lookup, connection establishment, key exchange, and encrypted messaging. The Receiver client handles hidden-service creation, endpoint registration, incoming connection handling, and message decryption. The Server handles key registration, authentication, endpoint resolution, and public-key retrieval. The Tor network serves as the transport medium for both the control plane and the P2P communication channel.

The sender-side logic is implemented in the outgoing pipeline, which authenticates the user, obtains a token, resolves the receiver endpoint, fetches the receiver public key, verifies the peer identity, and then proceeds to encrypted messaging. The receiver-side logic creates the hidden service, registers the endpoint, accepts incoming Tor connections, performs handshake validation, and processes encrypted traffic. The server-side logic exposes the routes used for registration, endpoint registration, authentication, lookup, and public-key retrieval, as confirmed by the server logs.

C. Operational Mechanism

The operational mechanism begins with the generation of a random identity hash and ends with secure encrypted message exchange. The identity hash is not derived from a local username or label. Instead, it is generated as a cryptographically secure random 256-bit value and represented in hexadecimal form:

$$ID = Hex(CSPRNG(256 \text{ bits})) \tag{1}$$

For authentication and signing purposes, the client will also create an Ed25519 Key Pair. The Private Key will be stored on disk locally and the Public Key will be registered with the server to allow the server a future means to verify the identity of this client. Additionally, both the identity and Key Material will be persisted to disk this way, enabling the same client to be used in multiple sessions.

After the client has created an Identity, the client will then register its Public Key with the server by calling the Register endpoint for Public Keys (`/keys/register`). The server will then check its database to see if there is already the same `user_id_hash` on record. If so, it will reject the registration request and require that the client create a new random hash identifier. This process of checking for duplicates in the database will assure the uniqueness of the records within the system and will help to prevent accidental reuse of the same client.

Once registration is complete, the Receiver creates a Tor hidden service and registers its onion endpoint with the server. The Sender then authenticates, receives a short-lived token, looks up the receiver endpoint, retrieves the receiver public key, connects through Tor, performs authenticated key exchange, and finally exchanges encrypted messages.

The system architecture is illustrated in Fig. 1, while the step-by-step communication flow is shown in Fig. 2.

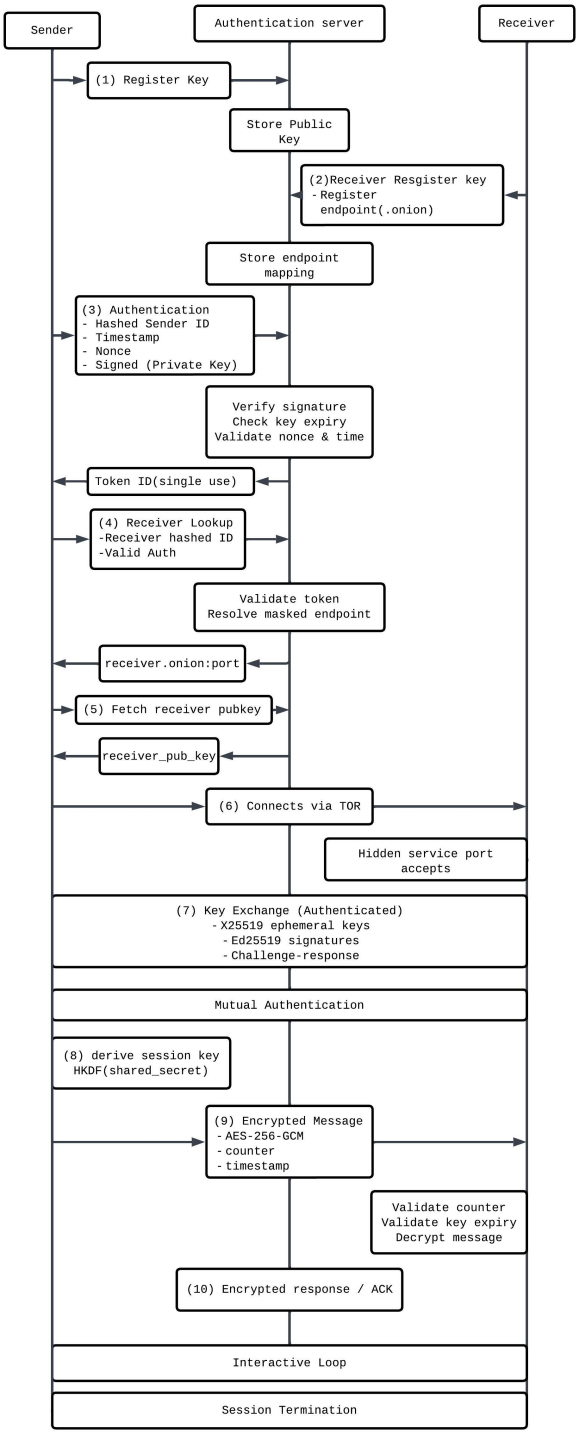

Fig. 2. Operational Mechanism and Secure Communication Flow

D. Identity Generation and Persistence

At system initialization, the client checks whether an identity already exists on disk. If not, the client generates a new random 256-bit identifier and stores it together with the cryptographic key material. The identifier is encoded as a 64-character hexadecimal string and becomes the public user identity within the system. The private key remains local and is never transmitted. The Ed25519 public key is used for registration and later authentication.

The identity generation process is therefore independent of human-readable names and depends entirely on locally generated random data. The identity and key pair are stored persistently so that the same client instance can be reused across sessions without regenerating a new key pair each time.

E. Registration with the Server

Once a new identity is created the client will register the public key with the `/keys/register` endpoint. The following attributes must be included in the request: a randomly generated `user_id_hash`, Public Key, and Timestamp. When the server receives the `user_id_hash` and the Public Key it will check to see if the `user_id_hash` is already present in the identity database. If the identity record associated with that `user_id_hash` already exists the server will reject the registration request and the client must create a new `user_id_hash`. This prevents duplicate identities from being created in the identity database.

Once the Public Key has been registered by the client the receiver will create a Tor Hidden Service and acquire a `.onion` address for the receiver then using the endpoint registration endpoint the receiver will register the endpoint with the server. After the endpoint has been successfully registered with the server the receiver will be reachable via Tor without disclosing its true IP address.

F. Authentication and Token Issuance

To initiate contact with a receiver, a sender must first authenticate with the authentication server via the `/auth/` endpoint. The authentication request consists of the sender hash, timestamp, nonce, and Ed25519 signature. The payload being signed is:

$$m = sender\_id\_hash \parallel t \parallel n \tag{2}$$

and the signature is:

$$\sigma = Sign_{sk}(m) \tag{3}$$

The authentication server will verify the signature and approve the timestamp for validity and will ensure the nonce was not previously used. If successful, the authentication server issues a single-use token that must be used by the sender to look up the receiver. The authentication service and clean-up logs provide evidence that tokens and nonces are short-lived secure artifacts.

G. Receiver Lookup and Public-Key Retrieval

After authenticating, senders can perform a lookup request using the token provided by the authentication server. The lookup request will verify the token and return the receiver's masked endpoint corresponding to the receiver's `.onion` address. The sender will then retrieve the receiver's public key from `/keys/{user_id_hash}`. This action binds the receiver's masked endpoint to its intended public key thereby reducing the chance of impersonation or endpoint hijacking.

The sender-side implementation also verifies that the public key received from the peer during the handshake is the same as the public key retrieved from the authentication server. If they do not match, the connection will be rejected. This comparison process is part of the sender's identity.

H. Tor-Based Connection Establishment

A SOCKS5 proxy is utilized by the sender to establish a network connection to Tor. The receiver is a Tor hidden service; therefore, the connection's target is the `.onion` address provided by the Tor lookup service. The provided Tor Manager Module has multiple functions, including: connecting via Tor; parsing an onion address; creating a hidden service; and creating the local listening socket for the receiver.

This stage ensures that:

- the Sender does not directly contact the Receiver's IP address,
- the Receiver remains hidden behind Tor,
- the Server is not part of the data channel.

The execution logs show that the hidden service is created and published before interactive communication begins.

I. Authenticated Key Exchange

After the Tor connection is established, both clients perform an authenticated key exchange using ephemeral X25519 keys and Ed25519 signatures. The shared secret is computed using Diffie–Hellman:

$$K = X25519(a_{priv}, b_{pub}) = X25519(b_{priv}, a_{pub}) \quad (4)$$

The session key is derived from the shared secret using HKDF:

$$K_{session} = HKDF(K) \quad (5)$$

A challenge-response exchange is also performed so that both peers prove possession of their long-term private keys. This provides mutual authentication and prevents man-in-the-middle attacks. The result is a fresh session key for each communication session.

J. Secure Message Transport and Replay Protection

After the session key is established, all application messages are encrypted using AES-256-GCM. The encryption frame includes a length prefix, counter, timestamp, nonce, and ciphertext. The authenticated data binds the counter, timestamp, and nonce to the ciphertext so that modification is detected during decryption.

Two conditions are enforced to ensure replay protection:

$$Counter_{new} > counter_{last} \quad (6)$$

$$|t_{now} - t_{msg}| \leq 30 \text{ sec} \quad (7)$$

A message will be deemed not accepted if there was an absence or failure of any of those conditions; then the message is not considered to be a fresh message and the possibility of its exploitation through replay attacks.

K. Receiver Operation and Endpoint Maintenance

The receiver operates as a background running the service after it has been started up. It will create its Tor hidden service, register its endpoint, listen for incoming connections, perform handshake processing, decrypt incoming messages, and either send back encrypted replies or acknowledgments to the sender. The receiver also periodically re-registers itself in order to maintain an up-to-date server directory.

L. Server-Side Security Controls

Several security measures have been implemented by the server, such as token expiration, nonce verification, rate limiting, timestamp checking, and public key expiration for the purpose of preventing the reuse of stale authorization data and curtailing abuse of the system. The background cleanup process removes expired tokens, expired endpoints, and expired nonces from the directory during periodic operations in order to maintain the minimal and consistent state of the directory.

V. Results and Discussion

A. Experimental Environment

A simulation-based experiment was conducted on VirtualBox using three separate virtual machines (VMs): a Server VM, a Sending Client VM, and a Receiving Client VM. Each of the VMs could communicate with each other and were tested without using public network resources. The Server VM was started prior to starting the client VMs, and both clients were configured for Tor usage, Python virtual environments and the dependencies necessary to perform registration, identity lookups, hidden service creation and encrypted communication. The first time the Server VM was executed, it indicated successful start-up, successful connection to the Tor network, successful database reset and successful start-up of the FastAPI web service to be served on Port 8000.

B. Initializing Clients and Generating Identities

When executing the clients for the first time, a local identity and its corresponding cryptographic key were created for each client. The log files for both clients indicated that no prior identity existed, thus both clients generated a new identity. After creating a new identity, each client registered their public keys with the server and completed the remainder of the communication process. The identities persisted across executions, indicating that local state persistence is functioning correctly.

C. Registration and Endpoint Publication Results

The server accepted the public-key registration requests from both clients. The server log shows successful `POST /keys/register` events for both identities, followed by successful endpoint registration for the receiver. This confirms that the directory service correctly maintained the mapping between identity hashes, public keys, and onion endpoints during the simulation. The receiver endpoint was therefore discoverable through the server, while still remaining hidden behind Tor.

D. Tor Hidden Service and Connectivity Results

The receiver successfully created a Tor hidden service and obtained a valid `.onion` address. The logs show the hidden service being published and the local listener being activated on the receiver machine. After this step, the receiver was ready to accept connections through Tor. The sender later used the resolved onion endpoint and connected through the Tor network instead of using a direct network address. This confirms that the anonymity layer was operational in the virtual environment.

E. Authentication, Lookup, and Public-Key Retrieval Results

The sender authenticated successfully with the server using a signed authentication request and received a valid token. The server log confirms the authentication event, followed by a successful receiver endpoint lookup and a successful public-key fetch for the receiver identity. This shows that the control-plane workflow operated correctly: the sender could authenticate, obtain authorization, resolve the receiver endpoint, and retrieve the receiver public key before establishing the secure session.

F. Secure Chat Session Results

Following the authentication process and lookup of both sender and receiver, an interactive encrypted chat session was successfully established. The logs from the tests revealed that the receiver accepted the request and, upon acceptance by the receiver, an active session was created whereby messages were exchanged between participants. The session ended cleanly when one of the users typed the command `quit`, and the client returned to the menu without experiencing any type of crash. Thus, it can be concluded that all components of the key exchange, encryption, and message transport layers were properly functioning during the simulation.

G. Security Evaluation Table

TABLE I
SECURITY PROPERTIES ACHIEVED BY THE PROPOSED SYSTEM

Security Aspect	**Status**
Confidentiality	Strong
Integrity	Strong
Mutual Authentication	Strong
Forward Secrecy	Strong
Replay Protection	Strong
Token Reuse Prevention	Enforced
Endpoint Integrity	Enforced
Key Revocation	Supported
NAT Compatibility	Supported
Server Trust Minimization	Preserved

The simulation provided security properties as described by each entry in Table I.

The entries include examples of the security behaviour of each of the security controls that were provided within each of the sender, receiver, and server components. Each of the replay protection mechanisms, authenticating session setup methods, and token-based lookups were all tested during the simulation runs.

H. Problems Addressed by the Proposed System

According to the results from testing, the proposed system solves many real-world issues involving the ability to communicate.

Firstly, it reduces IP exposure by using Tor hidden services to route communications. Now, neither client has to disclose their actual IP address to each other during the establishment of the connection; you can see from the logs that both parties are connecting to one another by using the `.onion` endpoint.

Secondly, by eliminating centralized trust, the server only has certain functions (i.e. registration, authentication, lookup of users, retrieval of public-key information) — the server does not store plaintext messages nor does it have any part to play in the connection that is encrypted. Basically, the logs for the server will only contain control plane activity and routine cleansing of tokens, nonces, and endpoints.

Thirdly, it provides for authenticated communication. The sender will verify that they have the correct public key of the intended recipient before continuing with the session, which means that impersonation could not happen at the connection layer. Moreover, following the lookup and key-fetch phases of the connection, both parties have confirmed through these operations that the receiving party has bound their public key to the endpoint they are trying to connect to.

Fourthly, the proposed system includes protection against replay attacks and modification of data through the use of AES-256-GCM and the use of replay guards. The encryption process uses both counter and timestamp, and the receiving party will perform a validation on all frames received before they are accepted.

I. Privacy Level Achieved

According to the results of the simulation, privacy was very high in each of four areas.

Network privacy: The Tor hidden service provides complete obfuscation of the receiving device's network address by having the sending device connect via Tor, instead of directly connecting to the receiving device.

Identity privacy: This system stores a local identity hash and a public key for each of the two communicating parties. All information that is sent to and from either party is stored at the server as an identity hash, while the server never has access to the plaintext message.

Communication privacy: All message exchanges are encrypted from one client to the corresponding client; therefore, the server does not have the ability to view the contents of either chat or email messages.

Metadata reduction: The server stores as little information about each user as necessary to complete registration, perform lookups, and validate tokens, and it removes expired records during the background cleaning.

Therefore, the degree of privacy achieved in this simulated design of a P2P system based on the use of Tor is very significant, particularly with respect to IP address obfuscation, confidentiality of messages, and reduced trust in the server.

J. Limitations Observed During Simulation

Although the system input an operational mode, it became apparent from the output of the simulation that there are a number of limiting factors to its performance in real world conditions. First, both users must be online at the same time for communications to occur; furthermore, the hidden service that a user is connected to must also be published and accessible before the user can connect to send or receive messages. In addition, the system must have successfully initialized the Tor bootstrap and have access to a stable SOCKS proxy. Limiting factors such as these are prevalent in the majority of anonymous communications systems using Tor; therefore, they were clear in the startup logs.

VI. Conclusion and Future Work

A. Conclusion

This project presented the design, implementation, and simulation of a secure anonymous peer-to-peer communication system built around the Tor network. The system separates coordination functions from communication functions by using the server only for identity registration, authentication, endpoint lookup, and public-key retrieval, while the actual message exchange occurs directly between the sender and receiver over Tor. The receiver operates as a Tor hidden service, which allows communication without exposing its real network address. The logs from the VirtualBox-based simulation confirm that the full workflow operated successfully, including key registration, endpoint registration, authentication, lookup, hidden-service creation, and encrypted interactive chat.

The implementation achieved the intended security objectives. Identity handling was based on randomly generated hash identifiers, while the server enforced duplicate-hash rejection during registration. The communication layer used authenticated key exchange with ephemeral keys, followed by AES-256-GCM encrypted messaging with replay protection. As a result, the system preserved confidentiality, integrity, mutual authentication, forward secrecy, and resistance to replay attacks. The evaluation results also showed that token reuse prevention and endpoint integrity were enforced, while server trust was minimized because the server never handled plaintext messages or session keys.

The simulation further demonstrated that the system addresses several real-world communication problems, including IP exposure, centralized trust dependency, and message interception risks. By routing traffic through Tor hidden services and restricting server involvement to control-plane tasks, the system achieved a high level of privacy suitable for anonymous secure communication in a controlled environment.

B. Limitations

Although the system performed correctly in simulation, some limitations remain. To communicate through the Tor network, both parties must be online at the same time, and the bootstrap time of Tor will affect the initial delay experienced when starting to use Tor. Presently, the Tor implementation does not support the storing of messages or delivery of messages while nodes are offline; these two limitations exist as a result of the Tor design and typify the trade-off between anonymity and usability.

C. Future Work

Further improvements could consist of users storing their chat histories on their device using encryption so they do not see these messages in clear text. To do so, we could leverage the AES-GCM encryption method already in place to store chat messages on the user's device in an encrypted format, using keys that are controlled by the user to protect them.

There are also some potential improvements that would include better methods of how users find contacts (contact discovery), improvements related to how we manage user sessions, and enhancements to protect users from attack methods that attempt to correlate or trace traffic.

References

[1] S. Kumari, "A research paper on cryptography encryption and compression techniques," International Journal of Engineering and Computer Science, vol. 6, no. 4, pp. 20915–20919, 2017.

[2] M. A. Al-Shabi, "A survey on symmetric and asymmetric cryptography algorithms in information security," International Journal of Scientific and Research Publications, vol. 9, no. 3, pp. 576–589, 2019.

[3] S. Dorward and S. Quinlan, "Robust data compression of network packets," Bell Labs, Lucent Technologies, 2000.

[4] A. Esfahani et al., "A lightweight authentication mechanism for M2M communications in industrial IoT environment," IEEE Internet of Things Journal, vol. 6, no. 1, pp. 288–296, 2017.

[5] M. S. Henriques and N. K. Vernekar, "Using symmetric and asymmetric cryptography to secure communication between devices in IoT," in Proc. ICIOT, 2017, pp. 1–4.

[6] N. A. Lal, S. Prasad, and M. Farik, "A review of authentication methods," International Journal of Scientific and Technology Research, vol. 5, no. 11, pp. 246–249, 2016.

[7] T. Nandy et al., "Review on security of internet of things authentication mechanism," IEEE Access, vol. 7, pp. 151054–151089, 2019.

[8] N. Koblitz, "Hidden monomial cryptosystems," in Algebraic Aspects of Cryptography. Springer, 1998, pp. 80–102.

[9] D. Pointcheval, "Asymmetric cryptography and practical security," Journal of Telecommunications and Information Technology, no. 4, pp. 41–56, 2002.

[10] G. J. Popek and C. S. Kline, "Encryption and secure computer networks," ACM Computing Surveys, vol. 11, no. 4, pp. 331–356, 1979.

[11] K. Renaud, "Quantifying the quality of web authentication mechanisms: A usability perspective," Journal of Web Engineering, pp. 95–123, 2004.

[12] M. Tömösközi et al., "Packet header compression: A principle-based survey of standards and recent research studies," IEEE Communications Surveys & Tutorials, vol. 24, no. 1, pp. 698–740, 2022.

[13] R. Davis, "The data encryption standard in perspective," IEEE Communications Society Magazine, vol. 16, no. 6, pp. 5–9, 1978.

[14] M. F. Mushtaq et al., "A survey on the cryptographic encryption algorithms," International Journal of Advanced Computer Science and Applications, vol. 8, no. 11, 2017.

[15] M. Mathur and A. Kesarwani, "Comparison between DES, 3DES, RC2, RC6, Blowfish and AES," in Proc. NCNHIT, 2013, pp. 143–148.

[16] B. V. Sundaram et al., "Encryption and hash based security in Internet of Things," in Proc. ICSCN, 2015, pp. 1–6.