Session

Angelica Shelman

Maria Orms

University of Denver

COMP 4722

Session

Session is an end-to-end encrypted messenger designed to minimize metadata, which refers to the data that provides information about other data. In the context of Session, metadata includes details such as the time and date of a message, the sender and recipient's IP addresses, and the size of the message. Session aims to support completely anonymous, secure, and private communications by minimizing metadata. The application allows one-on-one direct messages, group chats, and voice calls. Users can share files, images, and other attachments on the Session network. The application is fully open source, and it provides anonymity, security, and privacy that conventional social media apps, such as Facebook Messenger, Instagram, TikTok, and WhatsApp, cannot match. These qualities can be linked to several innate characteristics, including anonymous accounts with no associated phone numbers and emails, the avoidance of data collection, the lack of digital footprints, and censorship resistance. The networking component comes into play in Session by allowing users to connect, communicate, and share information.

Session's strategies to minimize metadata provide the first insights into the app's network security measures. According to Jefferys, Shishmarev, and Harman (2020), the application employs a decentralized network of thousands of economically incentivized nodes, rather than central servers, to perform all essential networking functions. Decentralization could be more practical for some services, such as hosting large group chat channels and attachment storage. However, the application accommodates user-side self-host infrastructure and uses in-built metadata protection and encryption to address trust issues. Furthermore, Session uses an onion routing protocol called an onion request to ensure IP addresses cannot be traced to messages received or sent by users (Jefferys et al., 2020). The application does not require email addresses or phone numbers to open new accounts. Instead, pseudonymous public-private key pairs are employed as the basis for account identity. In other words, accounts are identified using public-private key pairs, which provides a level of anonymity because real names or personal information aren't directly involved in the account identification process. This approach optimizes security at endpoint nodes in the application's network. Therefore, network security in Session focuses on data at rest and in transit and individual endpoint nodes.

Session's threat model indicates strategies for network security. The primary protections include sender and recipient anonymity, data integrity, secure storage, and end-to-end encryption (Jefferys et al., 2020). Only the conversation members know the sender's long-term identity key to ensure sender anonymity. All messages in the application also remain intact and unchanged, and they possess Off the Record (OTR) messaging protocol attributes to achieve end-to-end encryption, including Deniable Authentication and Perfect Forward Secrecy. Additionally, service node operators store messages on the application. These measures provide robustness to passive attacks, such as eavesdropping on network traffic, and active attacks, such as attempts to modify or disrupt the network.

Session employs a combination of advanced cryptographic techniques to secure user communications. At the core of its security is the open-source *libsodium* library, which provides the foundational tools for encryption, decryption, and digital signatures. When a user creates an account, Session generates a long-term X25519 keypair, with the public part serving as the user's "Session ID" for secure identification and communication. For one-on-one chats, Session relies on the Signal protocol to ensure end-to-end encryption (E2EE), whereby each message is encrypted with a unique key, maintaining confidentiality even if some keys are compromised. In group chats, Session uses the Sender Keys system to streamline encryption processes by allowing each participant to encrypt messages with their unique sender key, thereby reducing computational overhead while preserving security. Together, the Signal protocol and the Sender Keys system provide Perfect Forward Secrecy (PFS), meaning that the frequent rotation of session keys ensures that even if long-term keys are compromised, past communications remain secure. This layered approach ensures robust privacy and security for all Session app users.

Session uses an onion routing protocol (onion requests) as a foundational element. Onion requests allow Session users to establish a 3-hop randomized path through the service node network, facilitating the obfuscation of their IP addresses (Jefferys et al., 2020). Additionally, onion requests apply a basic routing protocol that encrypts requests for each hop. This approach limits the first node's knowledge of the client's and the middle service node's IP addresses. Furthermore, the middle service node only knows the first and last service nodes' IP addresses.

In contrast, the previous service node only knows the IP addresses of the request's final destination and the middle service node. Session requires clients to create a path on startup by choosing three random nodes from a service node list, specifying each service node's X25519 key, storage server port, and IP address (Jefferys et al., 2020). The client leverages this information to establish an onion, and the service node's X25519 key encrypts each layer. The created onion is conveyed to the first service node storage server, where its layer is decrypted to reveal the next node's destination key. The service nodes use the ZeroMQ is a high-performance asynchronous messaging library, aimed at use in distributed or concurrent systems. It provides an abstraction of sockets and asynchronous messaging patterns, allowing you to build scalable and resilient communication between different parts of your application, or between different applications on a network. It supports various messaging patterns such as publish-subscribe, request-reply, and push-pull, among others. ZeroMQ is known for its simplicity, flexibility, and efficiency, making it popular among developers for building distributed and high-performance systems. The first service node decrypts its layer and launches a ZMQ connection with the required downstream node (Jefferys et al., 2020). ZMQ efficiently manages tasks like message queuing, load balancing, and fault tolerance (Jefferys et al., 2020). After receiving the onion, the final node conveys a path-build success message downstream. Once the client receives this message, they encrypt it using the

final destination's X25519 keys. An ephemeral key, which is a temporary encryption key that is used only for a short period of time and then discarded, is also included in a client request, allowing the decryption of encrypted responses from the destination client or server (Jefferys et al., 2020). Therefore, the onion request system delivers a straightforward anonymous networking layer, which ensures optimal security.

While the onion requests and service node network achieve adequate network security, they feature several limitations that warrant additional services in Session's design. For example, while Loki blockchain incentives manage service node behavior, they are ineffective against data center outages, software bugs, and other unexpected threats to nodes (Jefferys et al., 2020). Thus, Session provides a secondary logical layer for redundancy on small clusters of service nodes referred to as swarms, illustrated in Figure 1. A service node is assigned to a swarm during registration, and each node retains minimal influence over its swarm. This scenario prevents malicious nodes from joining swarms, a crucial prerequisite for sustaining network self-regulation.

Session applies a swarm algorithm that rebalances the swarms to ensure an optimum number of swarms and a balanced number of nodes in each swarm (Jefferys et al., 2020). For example, a new swarm is established when many new nodes enter the network and saturate existing swarms (each swarm capacity is ten nodes) (Jefferys et al., 2020). This algorithm prevents single entities from controlling entire swarms and fosters network resilience to small and large-scale events involving attacks on service nodes. Thus, data privacy and integrity are assured in the network.

Session also employs the Signal protocol to ensure deniable authentication and perfect forward secrecy (Jefferys et al., 2020). The described protocol employs the Double Ratchet protocol to generate message keys and the X3DH key agreement protocol to establish shared secrets securely. This combination enables Perfect Forward Secrecy within asynchronous messaging scenarios (Jefferys et al., 2020). These additional services contribute to Session's private routing qualities, maximizing network security and ensuring users' messages remain anonymous.

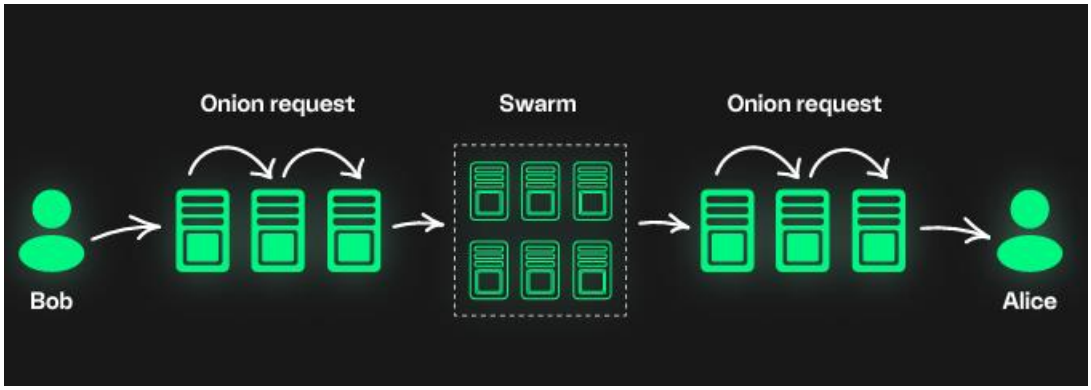

Figure 1 illustrates an onion request through a swarm in a Session message from Bob to Alice (Source: Session, n.d.)

In conclusion, Session is a unique messenger that differentiates itself from other similar applications through its absolute privacy and freedom from surveillance. Session's network security strategies are critical to achieving this differentiation. For example, the application uses the Loki Service Network Node network to store and route messages. All messages are also onion-routed through Session's network to prevent internal and external surveillance. Each encrypted message is routed through three nodes on the service node network, limiting the nodes' ability to compile meaningful information about senders and recipients of individual messages. These features translate to a secure Session protocol that offers users high confidence in the privacy of their shared messages and files.

References

Jefferys, K., Shishmarev, M., & Harman, S. (2020). Session: A model for end-to-end encrypted conversations with minimal metadata leakage. *arXiv preprint arXiv:2002.04609*.

Session (n.d.). *Lightpaper*. https://getsession.org/lightpaper

Session (n.d.). *Lightpaper*. https://getsession.org/lightpaper

Session (n.d.). https://getsession.org/blog/introducing-the-session-protocol

Session (n.d.). https://getsession.org/faq#onion

Session (n.d.). https://getsession.org/blog/session-protocol-technical-information