Development of an Instant Messaging Application Based on MLS using QRNG-Generated Keys

Iván Cillero, David Soler, Carlos Dafonte, Manuel Fernández-Veiga, Ana Fernández-Vilas, and Francisco J. Nóvoa

Centro de Investigación CITIC, Universidade da Coruña, 15071 A Coruña, Spain
I&C Lab, atlanTTic, Universidade de Vigo, 36310
Correspondence: david.soler@udc.es

DOI: *https://doi.org/10.17979/spu.23.c51*

Abstract: Nowadays, instant messaging applications have become the main channel of communication between people. In response to demands of privacy by users, some applications have begun to incorporate end-to-end encryption to protect conversations. The Messaging Layer Security (MLS) is a recent protocol that offers end-to-end encryption designed for groups, providing both efficiency and security. In this work we develop a messaging application that uses MLS as the basis for secure communications between users. The application will use a peer-to-peer network to prevent messages from passing through servers that could store them and try to decrypt them. Furthermore, the cryptographic keys used will be generated by a Quantum Random Number Generator and accessed through an Entropy-as-a-Service platform.

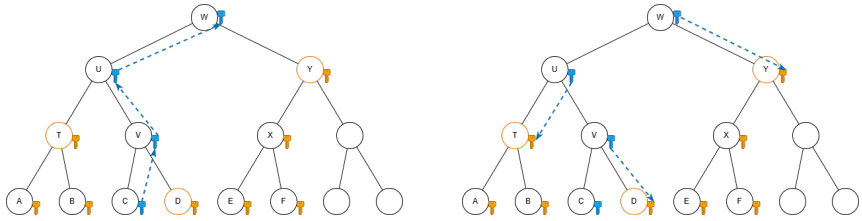
1 Introduction

Since the dawn of the Internet, numerous communication protocols have emerged that, together with their associated applications, have facilitated interaction between users. Currently, instant messaging applications have established themselves as the most common and widespread form of digital communication.

Initially, these messages were sent between users through the messaging applications' proprietary servers. The messages were encrypted between client and server. This protected the messages from eavesdroppers, but did not prevent the servers from accessing their contents. Over time, and with growing privacy concerns, users began to demand greater confidentiality guarantees. In contrast, modern applications employ end-to-end encryption (E2E) Cohn-Gordon et al. (2020): this approach ensures that only legitimate participants in a conversation can access the content of messages, preventing even intermediate servers from being able to read them.

However, these solutions initially focused on protecting individual chats, later adapting their schemes to work in group conversations. This adaptation has not always been efficient, leading to scalability and performance limitations in group environments. To solve this problem, the Messaging Layer Security (MLS) protocol, designed specifically for group messaging environments, was developed. This protocol enables more efficient key exchange and better encryption management, optimizing performance without compromising security.

To ensure truly secure communications, it is not enough to use a robust protocol such as MLS; it is also essential to use secure cryptographic keys. These keys must be as random as possible, which ideally requires the use of True Random Number Generators (TRNG). Since many devices do not have a built-in TRNG, an *Entropy-as-a-Service* (EaaS) platform can supply



(a) User C updates their keys and derives the keys for their upstream nodes from them, including the root node.

(b) Transmission of keys to sibling nodes of updated nodes so that they receive the new keys from their parent nodes.

Figure 1: Example of an update in an MLS group.

this need, which allows keys generated by a remote TRNG to be obtained over the Internet. In this way, devices can access a high-quality entropy source without the need for specialized hardware Vassilev and Staples (2016).

The combination of a secure protocol such as MLS and truly random keys makes it possible to establish highly secure communication channels. However, to further strengthen privacy, it is advisable to avoid messages passing through centralized servers, where they could be intercepted and stored by third parties with the intention of decrypting them later. An effective solution to this problem is the use of peer-to-peer (P2P) networks. In this type of network, devices communicate directly with each other, without the need for intermediaries.

In this work we design and develop a secure group communication application based on the MLS protocol that employs key material obtained through an Entropy-as-a-Service platform. The application provides confidentiality, authentication and integrity and handles changes to group membership efficiently. Furthermore, our application makes use of a P2P network to distribute messages to strengthen its privacy.

2 Background

We introduce the technologies employed in the design of our messaging application.

2.1 MLS

The MLS protocol organises group participants into a binary tree topology. This arrangement allows for efficient management of a single key within a group. Members can periodically update the shared secret to heal from compromises. Additionally, the group key is updated whenever a new user is inserted or an existing one is removed from the group. Thanks to this architecture, MLS provides guarantees of *post-compromise security* and *forward security*.

For shared key management, MLS assigns each node in the tree a pair of asymmetric keys. The private keys of any node are known only by their respective descendant nodes. In this structure, group users are located at the leaf nodes. The group's common encryption key is derived from the root node's private key, thus ensuring that all legitimate members can access it securely Barnes et al. (2023).

In order to update the encryption key for a group in MLS, it is necessary to modify the private key of the root node. Any user is at liberty to propose this update. The procedure is illustrated in Figure 1a: first, the user generates a *seed* for their leaf node, from which the parent node seed is recursively derived. The set of nodes from a specific leaf to the root is known as the *direct path*. The next step in the process is to generate a new key pair for each node from its corresponding seed.

Once the keys have been regenerated, the rest of the group must be notified of the changes made to the tree structure. To achieve this, the sibling nodes of the direct path are informed of

the modification of their parent nodes (see Figure 1b). Specifically, for each sibling node, the seed of the parent node is encrypted using its public key. In this way, each node on the *copath* can recover this secret and, from it, derive the new asymmetric keys of its ascending nodes, continuing the derivation until reaching the root node. The descendants of the copath nodes are also able to decrypt these seeds, since they possess the private keys of their ancestors. In this way, all members of the group are able to reconstruct the new state of the tree and, consequently, obtain the new encryption key for the group.

2.2 Point to point network

Traditional communications models are typically client-server architectures, where the client interacts directly with a central server. In the context of messaging applications, this means that messages generated by a user are first sent to the server, which acts as an intermediary and is responsible for delivering them to the recipients. This approach establishes a centralised infrastructure, whereby the server becomes a single point of failure, has total control over network traffic and limits the scalability of the system.

Libp2p

libp2p is a modular and extensible library designed to facilitate the implementation of P2P networks. Initially developed as part of the IPFS project, its purpose is to abstract and unify the different aspects of transport and communication in distributed networks. In this framework, each node is identified by a *PeerId*, generated as the hash of its public key. Establishing communication with a node involves the use of *multiaddresses*, which are structured textual representations that encode the necessary information to initiate and maintain a connection within the network.

In the context of P2P networks, the *GossipSub* protocol is a foundational element for the exchange of information between nodes. This protocol implements a publish/subscribe (*pub/-sub*) model that allows messages to be distributed efficiently among participants Vyzovitis and Psaras (2019). In GossipSub, nodes subscribe to specific topics and publish messages within those topics. In order to facilitate the sending and receiving of messages, nodes are required to establish a mesh for each designated topic. To that end, nodes running GossipSub exchange information regarding their subscriptions with neighbouring nodes in their routing table. Using the information provided, they establish connections with other nodes that share common interests, thus forming meshes based on different topics.

3 Entropy-as-a-Service

Entropy-as-a-Service (EaaS) is an architecture in which clients delegate entropy generation to specialised remote servers. These services provide truly random numbers over the Internet, allowing a large number of devices to access high-quality entropy without the need for local TRNG hardware Gong et al. (2019); Vassilev and Staples (2016). This approach enhances the security of systems that, due to technical or economic constraints, are unable to generate reliable entropy locally.

The EaaS architecture employed in this work is based on the *Privacy-Preserving Key Transmission Protocol* defined in Soler et al. (2024). In it, Users interact with two logical entities: the Authentication Server (AS) and the Proof Validation Server (PVS). The two entities possess some information in common: a Merkle Tree, which the AS populates and the PVS uses to verify proofs.

The communication between Users and the servers is divided into two phases: Authentication and Key Request. In the former, the User provides identifying information (e.g. a digital certificate) to the Authentication Server. If successful, the AS authorises the User to perform the

next step. In the Key Request phase, Users present a *Zero-Knowledge Succinct Non-Interactive Argument of Knowledge* (zk-SNARK) proof demonstrating that they have been previously authenticated. Crucially, it is impossible to link the zk-SNARK to the credential the User presented to authenticate.

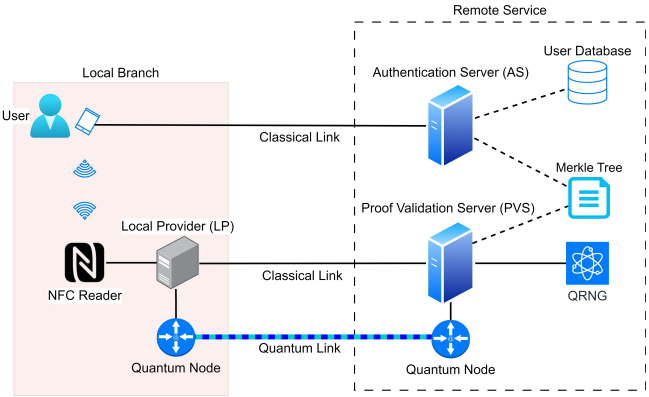


Figure 2: Architecture of the EaaS Platform.

EaaS Architecture The EaaS architecture we employ in this work is shown in Figure 2. It inherits the server structure and communication steps from the Privacy-Preserving Key Transmission Protocol of Soler et al. (2024). The EaaS platform is structured as a centralised service with multiple *Local Branches*, as shown in Figure 2: both the AS and the PVS are located in a *Remote Service* where common information is stored and the QRNG is accessible. In each of the Local Branches, a device known as *Local Provider* acts as intermediary with the Remote Service.

The Authentication phase unfolds as follows: The User first presents authentication information to the AS, which checks a User Database to verify its validity. If successful, the AS will authorise the user to perform the Key Request step. The AS will then validate that the User is authorised to request key material by checking the User Database, in which the access control policies for each User are specified.

The execution of the Key Request phase between Users and the PVS is mediated by a Local Provider, which forwards all messages exchanged between them. This device is located in the local branch and communicates with the User through Near Field Communication (NFC). In addition to a classical link the Local Provider and PVS are also connected through a quantum channel, used exclusively for the transmission of the key generated by the QRNG from the PVS to the Local Provider.

4 Messaging Application

In this Section we present the messaging application developed for this work.

4.1 Design

Figure 3 shows the network environment in which the messaging application is executed. Each instance of the application is executed in a mobile phone, and is connected with others through a P2P channel. The GossipSub protocol is used for this purpose, whereby each user subscribes to their own PeerId and the UUIDs of the groups they belong to. The application instances discover each other automatically inside a local network through the use of the mDNS protocol. Additionally, the message applications interact with the EaaS service to obtain entropy.

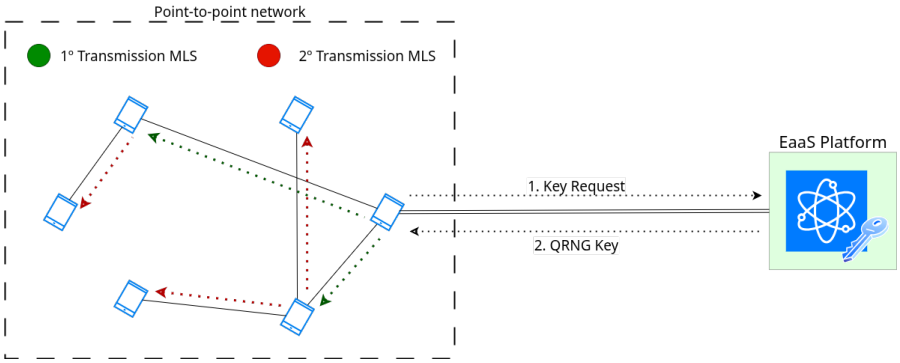


Figure 3: Network environment of the messaging application.

This network includes three main types of messages. The first is an *Invitation*, which is sent to users to invite them to an MLS group and is published in the thread associated with the target user's PeerId. *Upd* messages modify the group's shared secret, and *Msg* messages contain the encrypted application information exchanged between the users in the context of a group; both message types are published to the thread corresponding to the group's UUID.

All encryption keys used in MLS are obtained or derived from those provided by the EaaS service, as described in Section 3. The obtained entropy is stored in a file in the mobile device and it is periodically consumed whenever MLS requires a random input - which mostly involves the creation of *Upd* messages.

4.2 Implementation

We have implemented the messaging application for Android devices in Rust. Our choice of programming language is determined by the availability of cryptographic libraries to handle MLS and to interact with the EaaS service, which requires the creation of zk-SNARK proofs. We employ the *OpenMLS* openmls (2025) and *ZoKrates* ZoKrates (2025) libraries for said operations. To deploy a Rust application in Android we employ the Tauri Tauri (2025) framework. The application front-end is implemented with TypeScript and React, which are compatible with Tauri.

However, NFC communication is a low-level operation for Android devices and thus cannot be executed with Tauri. To that end, we have developed an auxiliary application in Java that exclusively handles NFC communication with the EaaS service. Both applications interact through an internal WebSocket.

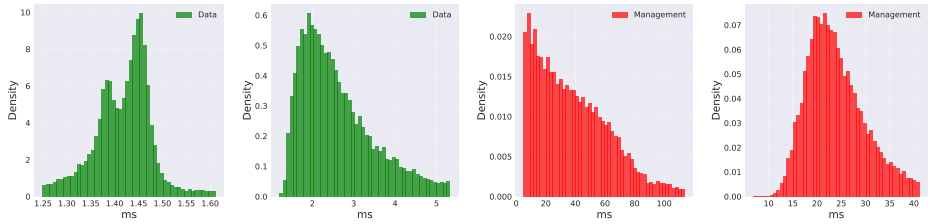
5 Evaluation

To verify the efficiency of our application, we conducted tests assessing the effectiveness of encryption and decryption using MLS, as well as transmission via the P2P network. These tests were performed on a virtual machine (VM) provided by CITIC. The VM ran Debian 11 as its operating system and had an `AMD EPYC 7763 64-Core Processor`. For the tests, the VM was assigned four CPU cores and 32 GB of RAM.

The tests involved running Docker containers that executed 100 instances of the application. These created groups and communicated with each other. The time taken to encrypt and decrypt the messages, as well as the time taken to receive them, was recorded. The values shown the analysis are the mean of the measurements obtained.

MLS analysis. Figure 4 shows the results obtained from the MLS operations. Here, we can observe the differences in efficiency between encrypting and decrypting various types of messages. Overall, operations on text messages are much more efficient than key management operations.

Operations on text messages (Figures 4(a) and 4(b)) are very efficient, taking less than 2 ms for encryption and 6 ms for decryption. By contrast, operations on key management messages are much slower, with the encryption process, shown in Figure 4(c), taking longer than the decryption process, shown in Figure 4(d). This is because asymmetric cryptography is used for this type of message and several encryption operations are required, whereas only one is required for decryption.



(a) Encryption data messages (b) Decryption data messages (c) Encryption management messages (d) Decryption management messages

Figure 4: Distributions of MLS operation times, categorized by message type.

P2P analysis. Figure 5 shows the results obtained from the delay of the different messages. As can be seen, the messages that take the longest to send are those related to key management, as these are much larger than text messages.

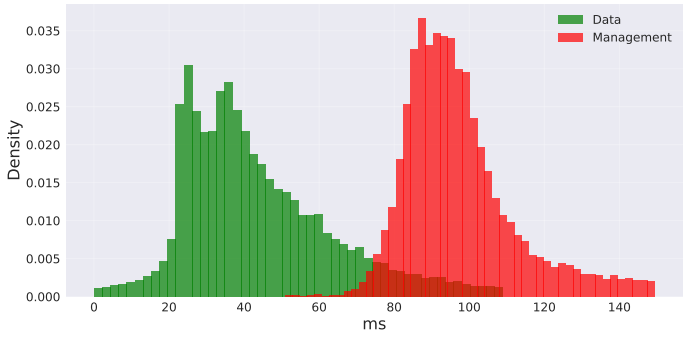


Figure 5: Distributions of sending times in a P2P network, categorized by message type.

Taking into account the total time required for encryption, transmission and decryption, the application's waiting times are very acceptable. The time elapsed from sending a message to receiving it is negligible for the user. Management messages take longer to send, but they are not sent often enough to affect the user experience.

6 Conclusions and Future Work

In this work we have developed an instant messaging application focused on the confidentiality and privacy of the exchanged messages. The MLS-based application logic efficiently handles

changes to the group composition. We have successfully integrated an EaaS platform into the application, which allows keys generated from a QRNG to be used to increase de entropy of shared secrets. The communication layer of the application has been made to work through a P2P network, thus eliminating the need for intermediate servers and guaranteeing the privacy of communications.

The implementation and evaluation of this application have laid the foundations for building secure messaging solutions using the new MLS protocol. In the future, this application could serve as a starting point for the development of new tools that take advantage of some of these technologies, promoting communication systems with more robust security.

For future work, we plan on improving the discovery method in the P2P network - such as employing the *rendezvous* protocol libp2p (2025) - thus allowing the application to find other users even in larger networks or with greater segmentation. It would also be interesting to investigate mechanisms to allow communication over the Internet such as WebRTC, overcoming the limitations imposed by the use of NAT.

7 Acknowledgements

The work is funded by the Plan Complementario de Comunicaciones Cuánticas, Spanish Ministry of Science and Innovation, Plan de Recuperación NextGenerationEU de la Unión Europea (PRTR-C17.I1, Ref. 305.2022), and Regional Government of Galicia (Agencia Gallega de Innovación, GAIN, Ref. 306.2022). We also acknowledge support from the Xunta de Galicia and the European Union (FEDER Galicia 2021-2027 Program) Ref. ED431B 2024/21, ED431B 2024/02, ED481A-2023-219, and CITIC ED431G 2023/01. CITIC, as a center accredited for excellence within the Galician University System and a member of the CIGUS Network, receives subsidies from the Department of Education, Science, Universities, and Vocational Training of the Xunta de Galicia. Additionally, it is co-financed by the EU through the FEDER Galicia 2021-27 operational program (Ref. ED431G 2023/01).

Bibliography

R. Barnes, B. Beurdouche, R. Robert, J. Millican, E. Omara, and K. Cohn-Gordon. *RFC 9420: The Messaging Layer Security (MLS) Protocol*. RFC Editor, USA, June 2023.

K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila. A formal security analysis of the signal messaging protocol. *Journal of Cryptology*, 33(4):1914–1983, Oct. 2020.

L. Gong, J. Zhang, H. Liu, L. Sang, and Y. Wang. True random number generators using electrical noise. *IEEE Access*, 7:125796–125805, 2019.

libp2p. *https://docs.libp2p.io/concepts/discovery-routing/rendezvous/*, Aug. 2025. URL *https://docs.libp2p.io/concepts/discovery-routing/rendezvous/*.

openmls. https://github.com/openmls/openmls, June 2025. URL *https://github.com/openmls/openmls*.

D. Soler, C. Dafonte, M. Fernández-Veiga, A. F. Vilas, and F. J. Nóvoa. A privacy-preserving key transmission protocol to distribute qrng keys using zk-snarks. *Computer Networks*, 242: 110259, Apr. 2024.

Tauri. https://v2.tauri.app/es/start/, Apr. 2025. URL *https://v2.tauri.app/es/start/*.

A. Vassilev and R. Staples. Entropy as a service: Unlocking cryptography's full potential. *Computer*, 49(9):98–102, Sept. 2016.

D. Vyzovitis and Y. Psaras. Gossipsub: A secure pubsub protocol for unstructured, decentralised p2p overlays, 2019.

ZoKrates. *https://github.com/Zokrates/ZoKrates*, May 2025. URL *https://github.com/Zokrates/ZoKrates*.