Development of Blockchain-Enabled Radio Communication Protocol for Secure Information Transmission

by

Nusrat Jahan Nishu
22101170
Faria Islam Dipti
22101729
Md. Arafat Sarkar
22101914

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
January 2026.

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Nusrat Jahan Nishu
22101170

Faria Islam Dipti
22101729

Md. Arafat Sarkar
22101914

Approval

The thesis/project titled " Development of Blockchain-Enabled Radio Communication Protocol for Secure Information Transmission " submitted by

1. Nusrat Jahan Nishu (22101170)
2. Faria Islam Dipti (22101729)
3. Md. Arafat Sarkar (22101914)

Of Fall, 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science.

Examining Committee:

Supervisor:
(Member)

Dr.Jannatun Noor Mukta

Associate Professor, CSE
Director, BSDS
United International University

Thesis Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam

Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi

Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

In modern communication systems, maintaining confidentiality, integrity, and availability plays a vital role in protecting information against people who aim to exploit vulnerabilities. Traditional radio communication protocols are reliable, but they lack robust security mechanisms, which causes data theft by interception and data tampering. Radio frequencies are crucial for emergencies, but are not reliable for transmitting confidential information without encryption, and Blockchain technologies are implemented in Radio Access Networks (B-RAN), but are still widely unexplored in radio systems. In this aspect, blockchain technologies in radio protocols can help to enhance messaging integrity and security. Our research aims to develop a blockchain-based decentralized communication protocol for amateur radio systems, which is designed to ensure secure and tamper-proof information exchange in real-time.

Keywords: Blockchain-Enabled Amateur Radio Protocol, Lightweight Blockchain, Software Defined Radio, Packet Integrity, Bandwidth-Constrained Networks, Proof Hash Binding, Plane Separation

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Amateur radio operators, also known as ham radio operators, play an important role in maintaining communication. During emergencies when commercial networks don't work, amateur radio systems provide necessary connectivity. They have built a global network that assists during emergencies, such as natural disasters, helps with communication at public events, and continually explores new ideas to improve the functionality of radios. Amateur radio uses different radio frequencies and communication methods that can work even around the world by bouncing the signals off the atmosphere. During natural disasters, when other conventional communication systems fail, ham radio can quickly demonstrate communication. Despite this, the amateur radio system has significant security challenges. Many nations have restricted or limited the use of encryption on amateur frequencies. So when messages are not encrypted, anyone with simple radio equipment can intercept them. Private and important information can be misused if it is not secured with encryption.

Blockchain technology has changed the way data is stored and shared by making it secure, transparent, and decentralized. This technology ensures that a single person or a company no longer controls data. Even though a few of the commercial networks have already used blockchain in their system to make them more reliable. But the radio context, it has not implemented blockchain technology to make protocols safer, stronger, and more reliable. As both blockchain technology and amateur radio systems are decentralized, combining them could enhance security and global connectivity with decentralized control.

1.2 Rationale of the Study or Motivation

The main motivation for this research is to increase safety measures in amateur radio systems while keeping the fundamental features of the radio intact. The amateur radio system has several regulatory requirements, which cause challenges in terms of the implementation of new security protocols. Due to this, the use of encryption is limited in the radio system, as it goes against the commitment of open communication and technical transparency. In recent times, some countries have started to allow limited use of encryption for emergencies. The progress in digital signal processing and software-defined radio technologies has created new opportunities for increasing amateur radio safety. Not only this, but it has also maintained compliance with the regulatory developments, licensing

requirements, and amateur radio protocols.

The role of amateur radio systems in emergency communication has become increasingly important as the commercial communication infrastructure has proven to be weak during times of natural disasters and other crises. Amateur radio structure allows significant backup communication for hospitals, emergency management agencies, the military, and disaster relief organizations. However, the decreasing percentage of safe communication in the amateur radio systems leads to the ineffectiveness of emergency communications. Coupled with this, the amateur radio system has been subjected to interruption by the commercial facilities in deliberate jamming attacks. This endangers the reliability of communication since the existing protocol lacks the mechanism to identify or validate the authenticity of data.

Amateur radio systems lack the necessary security measures despite their significance in emergency communications. The integration of blockchain technology in amateur radio protocol can ensure data integrity through its cryptographic function. The consensus mechanism of the blockchain can help validate and authenticate communication without any centralized authorities.

1.3 Problem Statement

Amateur radio systems lack security mechanisms despite their significance in emergency communications. Most of the amateur radio systems did not allow the use of encryption or had limitations on usage because they did not want to commercialize things and kept it open [63]. Eventually, the traditional system made it hard to protect the important information. However, many countries worldwide are now allowing encryption in ham radio for certain emergency communication systems. But the administrative framework is different in every place. Hence, there is an urgency for creating a new security technology that will enhance the security of amateur radio communication. The new security solution should follow the rules, alongside allowing amateur radio to maintain its experimental integrity and learning nature.

Digital amateur radio systems like Automatic Packet Reporting System (APRS) [31] do not use strong security encryption to enhance security. Rather, they use simple tools for correction and user identification. This makes the verification standard weak as they fail to recognize whether the signal/ message is coming from a reliable source or not. Initially, getting fake or repeated messages may seem harmless, but during any emergency, it will create a negative impact, as the data or information can also be misused. Many radio technologies are not immune to sniffing (transmitted packet collection), jamming (creating interference by emitting signals), relay (relaying transmission over different channels), replay (recorded packet retransmission), spoofing, and injection(rogue signals/packet transmission) attacks for the shared characteristics of their spectrums [33]. Digital protocols need to be developed to increase the security and maintain the standard of protection.

The decentralized nature of the amateur radio system often creates challenges in the network. Amateur radio is a self-organizing and distributive network, therefore making it suitable for use during disasters or any other emergency. That's why amateur radio needs to be secured to remain useful and safe during emergencies. Also, interference is very easy in amateur radio transmission as they are open, and the data is not encrypted. Amateur radio systems frequently face jamming and signal manipulation attacks. In this

case, attackers can easily send fake, threatening, and harmful messages. The receivers can be misled by these because there is no proper way to check the reliability of the incoming information.

1.4 Objective

This research aims to modify and evaluate a communication protocol based on blockchain technology, which is specifically created for amateur radio systems. This protocol will provide secure, decentralized information transmission while preserving the experimental nature of amateur radio and the associated protocols. The key objectives are:

- Utilizing the blockchain technology to integrate an amateur radio transmission system and a cryptographic data exchange.
- Reconfiguring the system to work with amateur radio frequencies, power constraints, and regulatory provisions, and also maintaining interoperability with the existing amateur radio equipment and operating procedures.
- Developing a security system, based on blockchain, capable of functioning with the current regulations on amateur radio and ensuring the transparency and authenticity of the data.
- Designing a reliable architecture capable of authenticating the integrity of data and stopping the improper interference of the data, but preserving the amateur radio identification protocols.
- Improving communication and transmission efficiency through the implementation of blockchain solutions to the radio network to minimize the overhead. Even when subjecting it to the communication barriers and security provisions of the current amateur radio activities.
- Assessing the performance of the amateur radio operations by conducting simulations to identify the protocol potential in secure communication.

These objectives will contribute to determining whether or not the performance value is good enough to support the emergency communications and experimental operations.

1.5 Methodology in Brief

The main objective of this research is to modify and develop a blockchain-enabled communication protocol, specifically for amateur radio systems, that will accommodate both emergency and classified communications. The proposed methodology (Figure 1.1) consists of seven different phases that address different challenges within the integration of blockchain technology and amateur radio constraints. The phases are:

A. **Protocol Selection:** The existing amateur radio protocols, such as APRS (AX.25), HSMM, and Winlink, need to be explored first. Later on, an analysis needs to be done on data transmission, text formatting, and station identification mechanisms to find a suitable point where blockchain should be implemented. Blockchain components need to be lightweight to match the limited bandwidth of amateur radio systems.

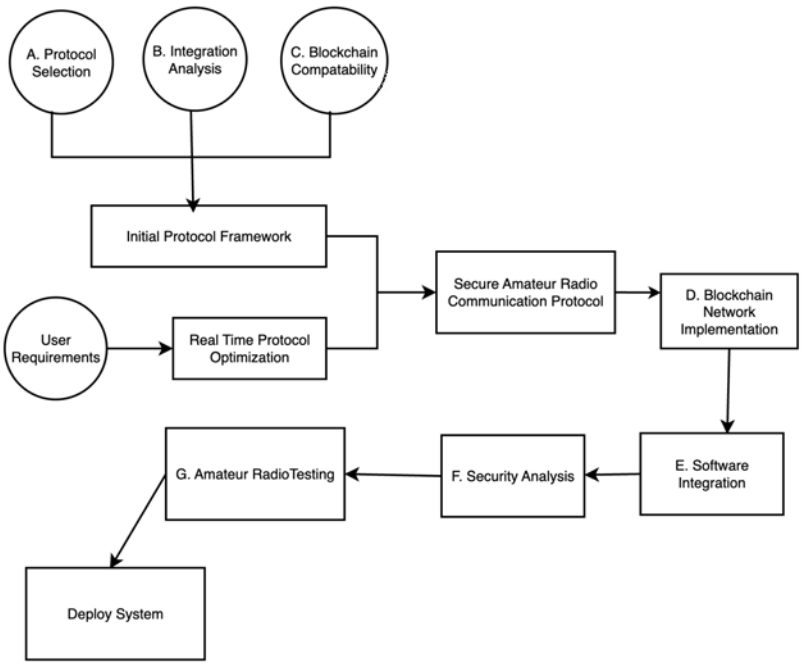

Figure 1.1: Architecture of the proposed Workflow

B. **Amateur radio integration analysis:** This will ensure that the proposed blockchain protocol aligns with the existing mechanism and limitations of amateur radio. In this part, frequency band behavior, signal propagation, and limitations affecting data transmission - all these will be studied. While designing the new system, a few key constraints like limitation of power, size restrictions, and usage efficiency - all these should be considered. The goal is to ensure the balanced integration of cryptography with conventional amateur radio protocol.

C. **Blockchain Compatability:** Blockchain creates identical copies of a ledger and distributes them throughout multiple participants in a network, whereas a traditional centralized database is managed by a single entity. The use of identical copies ensures that no single point of failure exists, and all of the data remains sustainable and accessible to all authorized parties. Transaction transparency can be verified easily through this process, and it offers greater security than conventional centralized ledger systems. Therefore, to track the data, the data must have an identifier, which will require a custom packet design. This customization will be done in this phase [62], [43].

D. **Blockchain Network Implementation:** In this phase, a newly developed blockchain solution will be implemented on amateur radio communication systems. Here, the main activity of blockchain is using a lightweight consensus algorithm. A lightweight solution is suitable for this problem because of its power and energy consumption, making it more effective in real life. It would also include digital signatures and other security features while following the amateur radio rules.

E. **Software Incorporation:** The blockchain system will be tested through software applications. In this phase, a special protocol will be created that can send packets in a blockchain-based network to the amateur radio while implementing the encod-

ing techniques. And it will be designed to work for both new and existing radios, running on software called Software Defined Radios (SDRs).

F. **Security Analysis:** A Security check should be performed to ensure that the new protocol performs well in any conditions. This incorporates cryptographic examination of employed security systems in its regulations. The threats, repetition, and jamming will be checked using it, and we will test it to confirm that the blockchain is effective in the varying conditions of signals.

G. **Amateur Radio Testing:** Simulated amateur radio environment implemented to test using transceivers, SDRs, and protocols operating in different frequency bands. Emergency communication situations will replicate the test to determine the performance of the protocol. This entails the research of the packet functionality, blockchain network protocol tests involving message integrity, and latency. The system's theoretical effectiveness will also be evaluated in the supporting cases of amateur radio emergency communication usage that will provide the basis for future real-life deployments.

1.6 Scopes and Challenges

This study will revolve around the development of a comprehensive blockchain-based communication protocol to be used in amateur radio systems, and this will support emergency as well as classified communications. The proposed work is expected to come up with practical solutions that can be applied using the existing radio equipment, and they are expected to provide significant security enhancements and make sure that the rules of amateur radio are followed. Moreover, it will cover the technical challenges associated with the implementation of blockchain and amateur radio, and the realities of implementation to the amateur radio community. The technical part of this paper is the creation of cryptographic protocols which are specifically aimed at use in amateur radios. This study will be used to test and compare the performance of the system over various amateur radio frequency bands, including HF (3-30 MHz), VHF (30-300 MHz), and UHF (300 -3000 MHz)[31]. Another thing to observe is the formulation of interoperability standards in individual radio systems and software frameworks, and it will also investigate how this band can be compatible with widely utilized amateur radio modes like SSB voice, FM, PSK31, FT8, and APRS [31]. The simulation-based tests will be useful in knowing the flexibility and reliability of the protocol throughout the various amateur radio programs.

In this research, there are several important issues to be overcome. Another significant challenge is the adherence to the rules governing amateur radio, because the process of implementing safety measures in amateur radio systems has to deal with complex and varied regulatory frameworks in different countries. The regulations of many amateur radio authorities regarding encryption emphasize that messages must be readable and identifiable, putting specific constraints on the use of blockchain, not comparable to those existing in the case of commercial communications systems. Amateur radio communication equipment typically operates within constraints of limited bandwidth assignments, limited power, and variable propagation conditions,s which may vary with frequency and solar activity. Such mechanical limitations of amateur radio could also be a problem. Consequently, gradual strategies of optimization are needed to implement light-weight

blockchain-based infrastructure capable of running well within these limitations and still ensuring security measures. Since there is radio equipment already available in the system, the amateur radio systems have become very difficult to function effectively. To incorporate blockchain into these protocols, much testing must be conducted on a variety of amateur radio protocols. It will be making sure that all is working well, and the rules are being kept. At the same time, there is a need to ensure that the system remains flexible to allow the users to experiment with their skill sets. Nevertheless, there is a layer of complexity in the reliability of emergency communications. That way, it must be made sure that the system does not go soft when there is a power failure, broken antennas, signal interference, or any other situation of critical concern.

Chapter 2

Literature Review

2.1 Preliminaries

The merging of blockchain technology and amateur radio communication brings up the concept of security and decentralization of power in the conventional Radio Frequency (RF) based communication systems. Conventional Amateur Radio (AR) protocols have a reputation for being flexible and not depending on commercial Radio Access Networks, including Next-Generation RANs. However, amateur radio systems suffer from insecure transmission protocols, lacking encryption and authentication mechanisms for sensitive data [13]. On the other hand, blockchain offers immutability, consensus-driven integrity, and a decentralized governance strategy [54]. These features of blockchain are parallel to amateur radio protocols.

However, interest in secure communication systems is growing rapidly, especially when commercial networks collapse after any disaster; the position of amateur radio stands as a critical fallback channel. Malaysian amateur radio emergency services(MARES) works as a volunteer-based community model for emergencies [22]. The implementation of blockchain in such protocols, already explored in advanced cellular networks like B-RAN, has not yet been fully realized in amateur systems [30]. Hence, it is crucial to understand the existing studies that have worked in amateur radio security, protocol design, and blockchain adaptability in communication networks.

2.2 Review of Existing Research

2.2.1 Radio Access Networks(RANs) and Amateur Radio Services(ARSs)

Radio Access Networks (RANs), such as 5G, 6G Next Generation-RANs, are commercial wireless infrastructures designed for high output, heavy connectivity, and quality of service (QoS) [25]. NG-RAN like 5G uses a flexible and split architecture between Distributed Units (DUs) and Central Units (CUs), using F1 [50], [25] and Xn [44] protocols (Crucial Interface for RAN) and applies security via encryption [55]. These are all managed by heavily controlled cellular networks.

On the other hand, Amateur Radio Service (ARS) or ham radio operates open-access principle using volunteers. The U.S. ham community functions under polycentric governance. According to his study, decentralized self-regulation enables spectrum sharing,

innovation, and emergency coordination [49]. Their model is inspired by Elinor Ostrom's Institutional Analysis and Development (IAD) framework [1], which evaluates a governance pattern that aligns with the principle of blockchain, proving that non-commercial, multi-layered control structures can succeed in shared communication systems. In addition to this, amateur radio's use during major crises like Hurricane Katrina and the 2004 tsunami, for emergency responses [22]. However, their study also flagged the lack of adoption and training as critical barriers.

2.2.2 Frequencies and Amateur Radio protocols

Radio frequency sends data using waves in the air. Eight categories of frequencies are essential for radio communication. They are Very Low Frequency(VLF), Low Frequency(LF), Medium Frequency(MF), High Frequency(HF), Very High Frequency(VHF), Ultra High Frequency(UHF), Super High Frequency(SHF), Extremely High Frequency(EHF) [6]. But the traditional amateur radios work on three frequency bands these are HF(3-30 MHz), VHF(30-300 MHz), and UHF(0.3-0 GHz). These frequencies have their limitations and trade-offs in terms of data speed and range. As an example, HF radio communication transmission is done by ionospheric reflection. HF transmission is reliable on 3 main concerns - Lowest Usable Frequency (LUF), Maximum Usable Frequency(MUF), and Frequency of Optimum Transmission (FOT). They can send signals for thousands of miles [26]. For optimal transmission, the frequency should be passed by FOT, and the frequency of MUF should be around 85 percent. Though the values can fluctuate by solar strength, daytime, terrain, latitude, seasons, and availability of HF links.

According to the American Radio Relay League (ARRL), which was a space-based experiment held in 2015, among all the participants, only 14 operators were successful in sending Morse-Coded continuous waves to an e-POP satellite using amateur radio [27]. The transmission signal strength was 100Hz and roughly 20 bits per second. The satellite received the transmissions while passing over, which is a total of 380 bits/sec, achieving 0.2 bits/Hz. It shows the remarkable distance capability of amateur radio.

AX.25, a packet radio protocol, came from an older protocol named X.25 and is widely used for amateur Radio for reliable digital communication using satellite [12]. Research shows that using reconfigurable Frequency-shift keying (FSK) Ax2.5 can operate over 2-FSK and 4-FSK schemes, and the data rates stay in the range of 1200bps to 2400bps, which is optimized for a low power narrow band of 435 MHz. This is suitable for communicating in line-of-sight (LOS) bands like VHF/UHF. Despite being a low data rate, it is still good for low data rate acceptable environments as it has built-in error checking, CRC-16, and a structured frame. Another research shows that the AX.25 protocol can give more throughput depending on the microcontroller it uses. By using software and Z80-based Terminal Node Controller(TNCs), the throughput spiked from 1.2kbps to 38.4 kbps [15]. In brief, TNC3, TNC7, and DLC7 controllers achieved throughput up to 11 kbps. After using optimized software, faster microcontrollers, and higher radio links, the speed can theoretically be achieved at 38.4kbps. But there is no effective range confirmed for this throughput.

In this particular issue, more powerful bandwidth and throughput for amateur radio are needed. High Speed Multimedia or HSMM and Amateur Radio Emergency Data Network (AREDN) networks deliver the fastest throughput using the IEEE 802.11b/g/n/ac (defines the different versions of wireless local area network (WLAN)) operable in 2.4GHz and 5.8GHz amateur bands [20]. This is done with data rates of 11 Mbps to 600 Mbps,

which are faster than narrowband digital modes and helpful for using real-time applications over amateur radio. These frequencies work in UHF and SHF, which brings the drawback of line of sight (LOS), limiting the range from 5-10 miles. But experimentally, the achievable range is 200+ miles, which requires powerful antennas that cost more energy.

2.2.3 Decentralized Ledger Technology, SDRF, and Radio Repeaters

As Radio Frequency passes through air, it is crucial to maintain the range of passing the data, but also make sure the privacy is intact. Here, the Decentralized Ledger Technology concept comes forth, which allows multiple participants (nodes) to share and update a synchronized digital record in a decentralized way. It ensures no central control and eliminates a single point of failure. Blockchain is the most well-known and widely used type of DLT [34]. In the ISO standard, DLT follows a 6-layered architecture, and those layers are: non-DLT systems, User, Application Programming Interface, Platform, Infrastructure, Cross-layer functions [48]. Blockchain-specific software also uses a 6-layer architecture, and the layers are: Data Layer, Network Layer, Consensus Layer, Incentive Layer, Contract Layer, Application Layer [41]. But recently, there has been a 3-layer architecture which includes the Protocol and Network layer, the Scalability layer, and the Interoperability layer [39]. These three layers are a combination of 6 layers from the blockchain model that help the developers to work faster, which can be useful to implement in nodes and applications.

Recently, software-defined radio frequencies have become cheap and can be used easily to communicate. Recently, SDR on Raspberry Pi can process radio signals without heavy energy consumption and big infrastructure [19]. In this case, Raspberry Pi can also be used as a node for DLT or a repeater with computational power. The usual repeater for radio signals has no computational power, but a Raspberry Pi has computational power. Raspberry Pi has already been used in various radio technologies and also in many projects. It has also been used as a repeater, which captures radio signals using dongles and strengthens the signal, and sends it via General Purpose I/O (GPIO) [38]. By using those concepts, it is possible to create a repeater as a Node for the Distributed Ledger Technology.

2.2.4 Blockchain's Challenges in Radio Contexts

Blockchain is a distributed ledger system that ensures tamper-proof and plain data recording through agreed-upon algorithms like Proof of Work (PoW), Proof of Stake (PoS), Practical Byzantine Fault Tolerance (PBFT), and Delegated PoS. These blockchain algorithms mainly differ in power consumption and computational effort. Recent research gives us insights into their architecture, showing elements like Merkle trees, hash-chaining, and smart contracts [53], [23].

Despite its robustness, blockchain introduces heavy energy and computational demands. The PoW systems (e.g., Bitcoin) consume above the limited amount of energy due to competitive mining, which requires computation. Even lightweight models like PoS face centralization risks and need stable, always-on connectivity, which is challenging for ad-hoc, constrained networks like Ad-hoc Relay Networking station ARS. Moreover, throughput in many chains remains lower (e.g., 7 TPS in Bitcoin) [23], which determines

that latency-sensitive applications or protocols like emergency radio messaging may not tolerate typical block confirmation delays.

2.2.5 Blockchain Radio Access Network or B-RAN

Researchers explored blockchain radio access network systems, two of which are notable. Firstly, a blockchain-based Radio Access Network (B-RAN) that lets user equipment (UEs) and access points (APs) talk to each other in a secure, decentralized, and self-managing way [30]. The framework uses smart contracts, enforcing service agreements, and introduces consensus models like PoW and PoD to accommodate low-power devices. Simulations with 5,000 UEs and 125 APs showed that B-RAN outperformed traditional decentralized access schemes in data delivery, latency, and achieved resilience to attacks (e.g., double-spending) while maintaining decentralized integrity. They also mentioned that using the traditional blockchain slows the transmission speed compared to the usual network.

The Blockchain-based Fog Radio Access Network (BF-RAN) framework embeds blockchain, partial learning, fog computing, and smart contracts into a multi-layered architecture [45]. This works by using hierarchical blockchain layers for access, terminal, and cloud management. To ensure latency, scalability, and security, they also recommend Practical Byzantine Fault Tolerance (PBFT) and Reliable, Replicated, Redundant, and Fault-Tolerant (RAFT) consensus to be used for real-time, decentralized radio access in future 6G networks.

2.2.6 Blockchain for IoT and Constrained Systems:

Blockchain is being used in IoT and edge computing, but it also has similar challenges to amateur radio, like limited power, low bandwidth, and the need for secure, decentralized communication. As seen in such projects as FHIRChain, Prescrypt, and SmartMedChain, permissioned blockchains and smart contracts can offer data integrity and safe information exchange in resource-constrained settings [29]. The privacy-preserving mechanisms of blockchain technology are also confirmed using these systems. For example, FHIRChain enables medical data exchange securely and audibly using cryptographic authentication and modular access policies. Prescrypt ensures patient privacy by embedding digital identity and consent records, and SmartMedChain allows anonymous access to records while maintaining data provenance and tamper evidence [29]. These models exemplify how blockchain can ensure authentication, access control, and transaction confidentiality.

It is also important to know that blockchain immutability is good for security, but not for privacy. Certainly, a team suffered high energy cost, a need for protocol standardization, and privacy risk while working on scalable ITS Systems in India [57]. All their transactions in the public blockchain are stored and broadcast, which may expose metadata to attackers. The same issue is reported in a conceptual framework that integrates Blockchain Technology(BT) with Differential Privacy(DP), which also addresses privacy issues in the Electronic Health Record or EMR System [51]. Even though there is no working prototype, it ensures security by adding Differential Privacy in 6 core layers of the blockchain, which actually adds noise or randomness to the data so that it can be tamperproof [7].

2.2.7 Lightweight Blockchain for Amateur Radio

Recent research highlighted the Lightweight Blockchain Protocols designed for devices with constrained power and bandwidth, like amateur radio. These models present the most promising light for implementing blockchain technology in amateur radio protocols. Here are some findings of a few Lightweight blockchains,

- LSB (Lightweight Scalable Blockchain) uses randomized waiting instead of mining and rotates keys to prevent tracking. It consumes just 0.07 mJ of extra energy per event and achieves <50 ms added latency [21].
- LiteChain uses CBFT consensus and hierarchical committee structures to maintain speed and fault tolerance in massive distributed systems [59].
- Tree-Chain uses randomized validator selection with sub-second block generation and high scalability (tested up to 500 validators) [36].

These models offer cryptographic verification, fast consensus, and low energy usage, perfectly aligned with the limitations and needs of amateur radio operations. For example, nodes using FT8 or AX.25 could exchange signed payloads under 300 bytes and still participate in consensus.

2.2.8 Dual-plane Architecture

Instead of relying on a single communication structure, the dual-plane architecture separates the traffic across two paths, each with its own role. This separation makes the networks easier to scale and better performing. Due to these advantages, such a design as dual-plane is particularly applicable to networks of safety significance, where the issues of reliability and timing matter. In this manner, the OrthoNoC design employs a dual-plane methodology of Ethernet combining wired and wireless communications [24]. Instead of the traditional coupled systems, the OrthoNoC considers these media as a distinct plane, and routing decisions are made at the network interface via certain traffic rules. This isolation enables designers to enhance wired and wireless connections separately, taking advantage of the dependability of wired connections and the high-speed and proper order broadcast of wireless connections. It has been demonstrated through experimental findings that this method is much superior to conventional wired-wireless designs, which provide more flexibility and are capable of addressing synthetic workloads [24].

On the same note, Time-Triggered Ethernet (TTE) dual-plane architecture is employed to isolate the time-triggered traffic and event-triggered communication. It is significant to aerospace and industrial automation, where the safety-critical data needs an accurate timing and no interruption by other non-critical traffic. The system eliminates jitter through the use of separate planes, and the critical messages are guaranteed to meet the strict real-time deadlines even when the network is overloaded [35]. Thus, be it to support physical communication pathways or to isolate two types of traffic within Ethernet switches [35], dual-plane architecture provides isolation, which is required to ensure the entire system reliability and performance in complex communication environments.

2.2.9 Hash-based Message Authentication Code(HMAC)

HMAC is popular and extensively popular in fast data transfer and storage systems to authenticate data. Recent research is aimed at ensuring that HMAC is faster in

order to ensure that packets can be checked within a very short time without necessarily increasing the network overhead. The most important problem with the network coding is parameter pollution attacks that not only consume bandwidth but also processing resources. As much as traditional HMACs and digital signatures assist in securing the packets, they add an extra layer to the packets and lower the speed in general networks. To solve this, a scheme that is bandwidth-efficient proposes a layered scheme based on two homomorphic tags and a signature to secure a payload as well as the authentication data itself [37]. It minimizes the number of symbols used to represent each message, and minimizes communication overhead tremendously, but with no security loss by using the Admissible Payload Rate (APR) measure. This has rendered it suitable in settings where the bandwidth is limited and high throughput is a concern. Strong hash functions such as SHA-256, SHA-3, or SHA-512, and HMAC provide additional security measures to the system. It operates by combining a secret key with a hash to produce a new code that indicates that the data has not been altered, and it is of a source [61]. Research indicates that such enhanced HMAC protocols are quite effective in dealing with most cyberattacks, and thus, they are a reliable and versatile means of safeguarding sensitive information.

2.2.10 Zero-knowledge Proof (ZKP) Protocols

Zero-knowledge proof (ZKP) interactive protocols are transforming IoT connectivity. They enable the devices to interact with one another without exchanging sensitive information and relying on a central authority. A good one would be the zk-IoT framework. It employs a blockchain-based protocol to establish secure interactions between devices that do not entirely trust each other and is based on functional commitment-based ZKPs. Within this context, a device can demonstrate integrity by simply having verified firmware in 694 milliseconds, and the nodes of a blockchain can confirm these proofs within 19 milliseconds. Due to this reason, a trusted session can be achieved after the proofs fulfill the requirements of the service. The whole procedure could use approximately 3 seconds, offering high security and privacy of data [56]. On that note, Z-PMA (Zero-Knowledge Proof-based Privacy-Preserving Mutual Authentication) system introduces a low-weight authentication functionality to IoT devices requiring low-latency connections [58]. In comparison with the older protocols, Z-PMA combines a permissioned blockchain and interactive ZKPs to make it easier on devices that have limited resources. It also ensures the privacy of users because the user information is directly linked to the key exchange, and this helps in avoiding man-in-the-middle and replay attacks when establishing a session. This establishes hardwired trust anchors in which it is possible to generate session keys in a secure manner without necessarily storing them in a memory that may be vulnerable to attacks.

2.2.11 Linear Chain Topology

The most common topology to configure nodes in wireless sensor networks (WSNs) is a linear chain topology. Linear chain networks also have communication and energy issues as opposed to normal two-dimensional networks. Due to this reason, they need special design considerations to work well. Thin linear networks have nodes that are almost parallel to a single line. In dense linear networks, a small separation is observed between two parallel boundaries. Such a structure simplifies routing and reduces redundant transmis-

sions, which conserves energy. But the nodes that are nearer to the data sink are forced to support larger traffic. This may eventually cause their energy to run out more swiftly and consequently reduce the overall life of the network in case the network is not well managed [10]. Chain-type topology tends to be utilized to track extensive infrastructure. In this case, nodes are hidden in a sequence along the route, which creates a simple and easy-to-follow chain of communication. The major benefit of such an arrangement is that the data is transferred in only one direction, rather than anycast or mesh routing. This simplifies the network to implement and estimate the network communication delays. Meanwhile, a single hop in the chain is a single point of failure, and that is only one of the mechanisms of such a design. Nodes can be clustered together in order to save energy and support larger networks. Cluster heads gather data from the neighboring nodes and transmit it to the sink. This minimizes long-distance transmissions, as well as minimizes the total communication overhead [9]. These methods are particularly valuable in enormous networks, where it is vital to maintain communication over long distances [16].

2.3 Summary of Key Findings

The reviewed literature focuses on the foundational knowledge to build a decentralized, secure radio network. In this modern era of technology, the need for secure communication systems is growing day by day. Mainly, when the data is important and confidential. Despite its flexibility, amateur radio lacks in security and privacy. Therefore, the conventional amateur radio requires crucial security modifications that do not compromise decentralization and simplicity of the system. Selecting the appropriate bandwidth should be the first technical aspect, as different band ranges have trade-offs in throughput, noise, and range. Blockchain needs to have a higher throughput because of encryption and verification procedures. Selecting the right band will ensure that the blockchain's data demands are within the limited spectrum available in amateur radio.

Besides this, Distributed Ledger Tech (DLT) relies on network nodes to secure the data using cryptography and data verification. This decentralized method enables trustless and tamperless communication, which is ideal for confidential data. DLT doesn't rely on centralized infrastructure, which increases the integrity of data transmission during disasters and also in remote areas. In this blockchain implementation, SDRs, also known as software-defined radios, play a crucial role by using low-cost computing power, which can be used as a relay and to verify blockchain data. Amateur radio infrastructure can be modified to support blockchain functionality as SDRs enable the adaptation of radio functionality through software only. As software-defined radios increase security and flexibility.

Finally, blockchain is the core element that provides immutability, security, and privacy. However, traditional blockchain comes with trade-off issues and doesn't work on all devices. As a result, amateur radio requires optimization or simplification of the existing blockchain protocols. It will ensure that the system will function during emergencies, even with low power. Over time, the integration expanded, and recently, IoTs and Commercial Radio Access Networks have been integrated with blockchain too. These are feasible for adapting blockchain to RF-based networks like amateur radio for securing the communication framework. Protocols like LitChain, LSB, and ITree-Chain can work securely in a decentralized system while operating on low computational power. These lightweight

protocols are suitable for amateur radio, where speed and power matter. The dual-plane architectures divide the network traffic across two separate paths, making networks predictable and efficient. This approach is particularly useful for manycore processors and safety-critical systems. By handling wired and wireless links or separating time-triggered from event-triggered traffic, these designs can keep communication reliable even under heavy loads. For security purposes, HMAC and zero-knowledge proof (ZKP) protocols can be used to protect data integrity and privacy. The new methods of HMAC reduce the overhead and withstand attacks, and ZKP methods allow fast, secure authentication of low-resource devices. The line chain topology of WSNs provides a simplistic communication route, which eases routing and minimizes redundant transmissions. Energy-efficient clustering and adoption of topology-aware protocols are useful in ensuring an even distribution of energy consumption. This makes the network stable, particularly during large or long-distance deployments. Thus, it is possible to state that the above-mentioned research efforts and ongoing studies provide clues as to the opportunities of integrating amateur radio and blockchain technology to develop a decentralized, reliable, and secure system of communication. This hybrid system can preserve the experimental nature of amateur radio while supporting emergency communication.

Table 2.1: Summary of Key Findings

Author	**Year Published**	**Results/Findings**
V. Tomar and V. Bhatia	2015	Raspberry Pi can process radio signals and work as a Software Defined Radio with low energy consumption. It is useful in emergencies.
G. P. Kw5gp	2016	HSMM has the highest data throughput using Amateur Radio in optimal range among other protocols.
X. Ling, J. Wang, T. Bouchoucha, B. C. Levy, and Z. Ding	2018	Actively used blockchain in radio access networks where user and access points pass data in a secure and decentralized way using consensus models like PoW and PoD.
A. Dorri and R. Jurdak	2020	The Tree chain was designed to work with limited-resource devices like Raspberry Pi. The speed of this is lightning fast at generating blocks in 370 ms.
A. Kharche, S. Badholia, and R. K. Upadhyay	2024	The architecture used 3-layer models that are TinyML, AI/ML-based analytics, and permissioned Blockchain. The implication of decentralized communication, real-time data routing, and bandwidth optimization ensures data validation and tamper-resistant records across IoT networks.
H. Chen, R. Zhou, Y.-H. Chan, J. Zhihan, X. Chen, and E. Ngai	2025	The performance of the LiteChain is higher than other blockchain benchmarks, maintaining the consensus success rate. It uses SHA-256, mitigating relay attacks and data poisoning. The communication complexity is reduced compared to other blockchain models and has low latency, low storage, and high security.

Chapter 3

Requirements, Impacts and Constraints

3.1 Final Specifications and Requirements

This section of the research evaluates the achievements our system acquires and also the boundaries that need to be maintained to operate the system. The primary objectives are to create a secure communication network for amateur radio to prevent unauthorized message injection and modification. Moreover, it will maintain real-time communication performance, eliminate single-point failure, and achieve decentralization. The primary objective is to create a decentralized, authenticated, and real-time performative amateur radio-compatible blockchain network that guarantees security and integrity.

The Blockchain Enabled Radio Network (BERN) protocol is well-performed, highly secure, and scalable in urgent situations. Firstly, the radios create sessions within less than 150 seconds, transfer packets in real-time within 1ms, have a high throughput of 1000 packets/second, and also have fault tolerance up to f=[(n-1)/2] for neglected or malicious nodes. Secondly,y it is immune to wireless vulnerabilities (replay attack, tampering), ensuring authentication via sessions and message integrity using HMAC. Finally, for the decentralized nature of the system, it is scalable,e supporting 3-10+ consensus nodes and concurrent sessions.

BERN has many functional and non-functional requirements with few constraints. Functional requirements are in many categories. For authentication, it verifies message authenticity using HMAC, binds messages to ZK-SNARK using proof-of-work, detects and rejects duplicated packets, validate message timestamps to prevent replay attacks. For consensus, it collects majority node signatures from the network nodes, creates immutable blockchain record sessions, distributes final blocks to all network nodes, and synchronizes session caches from the blockchain. Moreover, sessions established using zk-SNARK proof generation derive keys per session using HKDF and expire sessions after 1 hour(default). Lastly, the system should process packets in 112 bytes header, verify packets before forwarding, deliver authenticated messages, and log verification failures for later review.

Most notable non-functional requirements are micro second lookup for sessions using cache, maintain 1000+ packets/second throughput, use constant time operation O(1) for HMAC comparison, protect session key in memory, log all security events, and human-readable error messages. A few limitations and restrictions that need to be addressed are that the system itself is a Docker simulation, session establishment takes 120+ seconds, the maximum packet size is 10MB, cache shortage for session capacity, and port separation

for different planes, with 7000 for the control plane broadcast and 5000+ ports for the data plane traffic. It also has limitations on practical hardware evaluations. BERN uses established cryptographic standards for development. BLAKE2b, SHA-256, ed25519, HKDF, and HMAC. It also follows TCP and UDP networking protocol standards and JSON for encoding.

3.2 Societal Impact

The overall outcome of societal impact is positive across all dimensions if the system is deployed responsibly. Appropriate safeguards and community engagement, like amateur radio communities, will be beneficial for society for emergency communication in disaster areas and rural areas. BERN needs lawful monitoring via systems authentication without encryption design, and stops the risks of misuse. In terms of health and safety, BERN has clear advantages to create a backup channel for telemedicine and infrastructure protection due to its nature to prevent a single point of failure. BERN can integrate privacy-preserving architecture through legal compliance for critical mission security and classified communication, but a 120+ seconds session establishment time is the biggest trade-off for mission-critical scenarios if the session is not predefined. The most impactful and efficient part of BERN is its low-energy consumption mechanism. The system is designed to run repeaters on Raspberry Pis in the future and is focused on computational efficiency. The majority of the nodes can run on renewable energy like solar panels. A 5 repeater node deployment will consume less than 1000 kWh, where 15W is the consumption rate of a Raspberry Pi 4 [47]. These findings support moving forward with system development and deployment under conditions like authentication vs encryption, mandatory backup for emergency or security incidents, and preference for using renewable energy in long-term installation processes.

3.3 Environmental Impact

Environmental sustainability of BERN is characterized as a moderate, continuous power-consuming design but exceptionally energy efficient compared to other cryptographic systems. Generating computationally expensive zk-SNARK proof takes high energy consumption across thousands of messages, where continuous HMAC verification takes negligible power. BERN's mesh topology reduces environmental impact using short-range transmission require lower power per range or hop than centralized long-distance architectures. BERN also eliminates the dependencies of energy-intensive data centers, characterizing cloud-based authentication systems and Key Management Facilities. Also, software sustainability complements hardware longevity using efficient algorithms, long-term maintainability, and minimal dependencies. This not only minimizes the energy consumption during development but also overhead during operation since the system uses O(1) session cache look up to avoid the unnecessary computation, such as O(n) constant time HMAC comparison to avoid timing attacks. In addition, conserving CPU cycles and disk at the same time, Hybrid storage is able to reduce the memory with no impact on performance. Dependence on the standard library of Python is minimal, resulting in reduced software bloat, reduced maintenance load, and external dependencies. Such design options have the advantage of saving immediate energy, fewer rewrites

required to minimize development time, and fewer bugs mean less effort to debug, since the development resources are not spent on detecting bugs.

Chapter 4

Proposed Methodology

4.1 Design Process or Methodology Overview

4.1.1 Introduction to the Design Process

Our objectives are to create a Radio Communication Protocol with a Blockchain to provide secure transmission of information. In this regard, we are adhering to some framework according to which the incorporation of this radio protocol applies a systematic and cyclic design methodology along the Design Science Research (DSR) framework [5], [8]. It is made to address the issues of integrating software-defined radio (SDR) simulations with decentralized technologies such as blockchain. The DSR approach is oriented toward problem identification and protocol, simulation, and digital ledger development of its solutions. With this strategy, we have attempted to make sure that all the stages of our system can be used in connection with real-life issues and technical obstacles. For example, amateur radio systems must follow FCC Part 97 regulations. It limits encryption and transmission power, while blockchain systems must handle issues of scalability, security, and transparency. We also aim to design packets for future RF hardware (USRP SDR), while following ham rules to ensure compatibility and create fixed-size proofs for blockchain verification, supported by multi-hop repeater networks for scalability. The Blockchain part of our network system will be integrated separately with the amateur radio. The radio will not work as a node in the blockchain; Rather, the blockchain network will be the gateway for the radios. This will establish the security while transferring the radio packets from one point to another. Producer, repeaters, and receiver will be the core nodes of the blockchain network for Amateur Radio. This will handle all the incoming packets coming from amateur radios and format the packet structure compatible with the blockchain network, produce blocks, create sessions, run consensus on the network, and deliver to the target amateur radios.

Through these steps, the research will also focus on the issue of reliability, scalability, and compatibility. All in all, the whole procedure will enable us to design a working and safe communication system which will meet the criteria of the technical and legal aspects of amateur radio transmission.

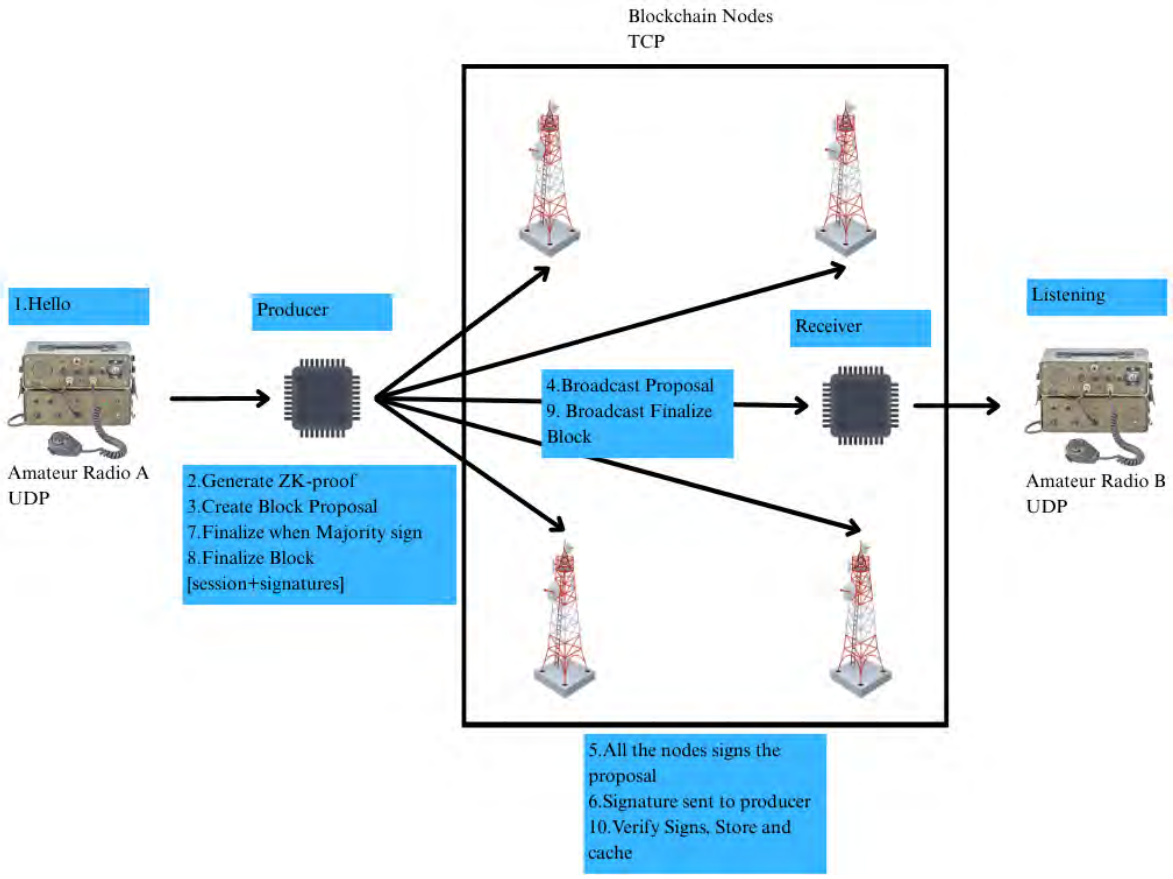

Figure 4.1: The Control Plane (Port 7000 Broadcast)

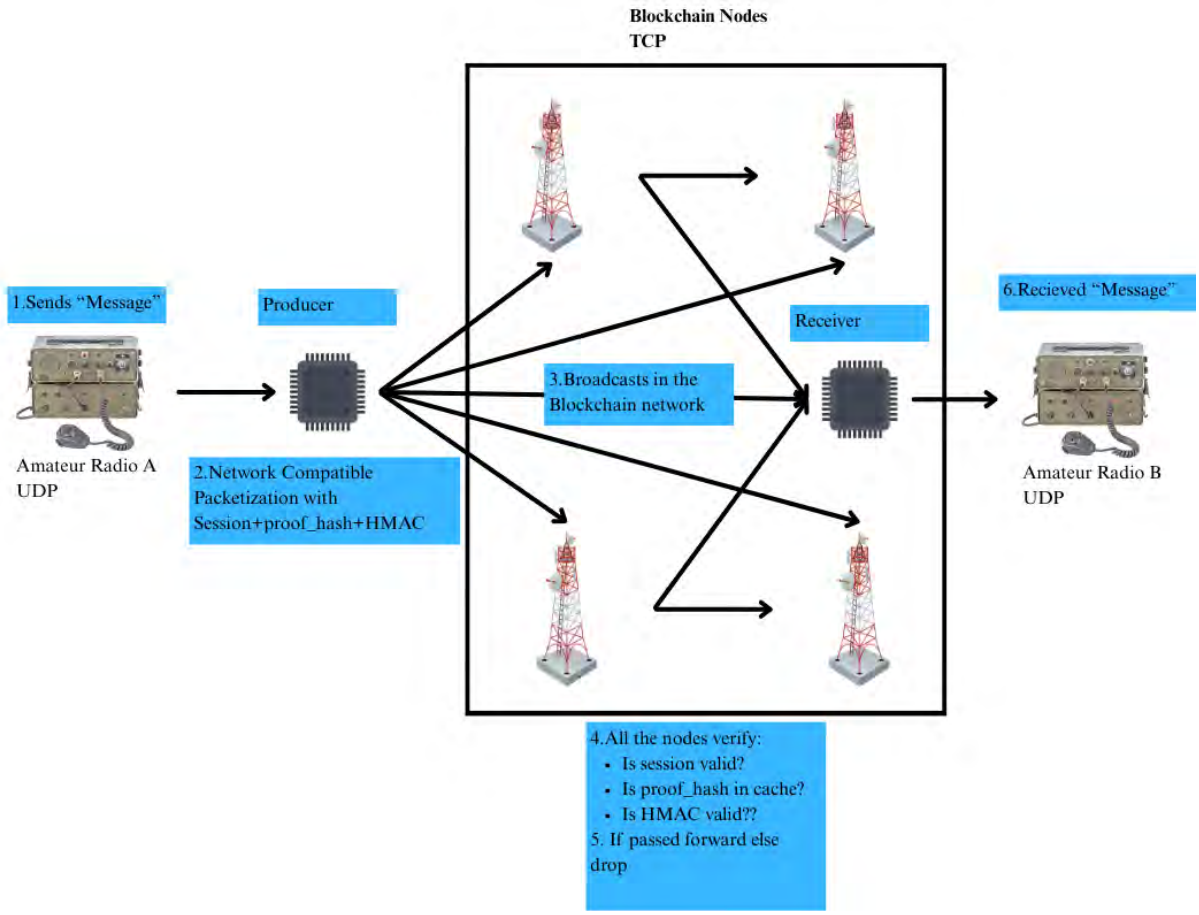

Figure 4.2: The Data Plane (Port 5000 series linear chain for performance testing)

Following the design process, we started by analyzing the protocol. After the design process, we began with protocol selection analysis, integration analysis, and blockchain compatibility. Next, we will proceed to Prototyping, which is the first step to obtain the security of the protocol. We intend to base our network implementation in the system on the principal concept of the Mina blockchain. GNU Radio will be integrated in terms of simulation of SDR, and the optimization will be performed based on software integration and security analysis. Finally, deployment of the system will be done based on the plausible result.

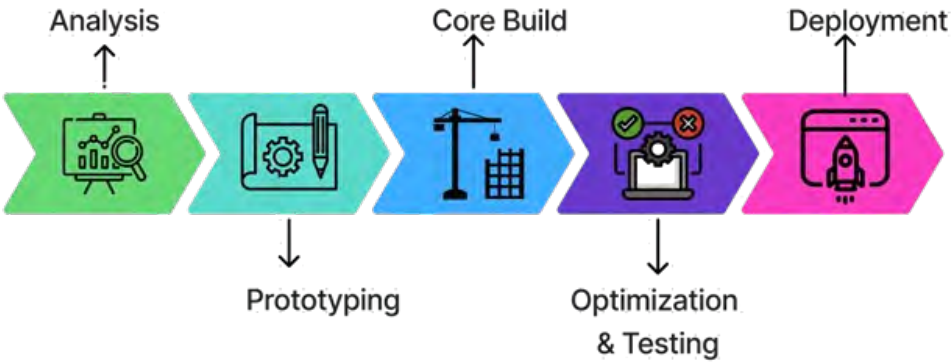

Figure 4.3: Research Workflow (DSR)

Our system will incorporate reliability and security into the conventional ham radio communication. To this end, we will develop a fixed packet-based radio connection that will be able to transmit and receive messages using amateur radio channels and implement blockchain technology in the system to trace and authenticate every message. We shall maintain the entire system light such that it can operate on equipment that does not need large computing capabilities, such as radios and repeaters. The primary purpose of this work is to design a safe and decentralized communication protocol that would guarantee the delivery of the messages with the correct meaning even in noisy conditions. The blockchain technology is also capable of recording the messages without a central server and assists the users in checking them in case of an emergency. We also aim to design packets for future RF hardware (USRP SDR), following ham rules to ensure compatibility and create fixed-size proofs for blockchain verification. This verification will be supported by multi-hop repeater networks for scalability. Through these steps, it will also focus on the issues of reliability, scalability, and compatibility.

4.1.2 Phase A: Protocol Selection

The initial phase of our system is the protocol selection. It demonstrates the foundational data framing rules by evaluating different existing protocols. This evaluation is done following our objectives of blockchain compatibility, constraints of simplicity, and low overhead. We conducted a comparative analysis focusing on metrics such as packet overhead, error resilience, and extensibility for zero-knowledge proofs (ZK-proofs) [46]. We also tried to assess Legacy amateur radio protocols, including AX.25 (a datalink layer

adapted from X.25 for packet radio) and APRS (An alternative method of expressing a station's location is to provide a Maidenhead locator in AX.25 bacon, with the APRS Data Type Identifier) [2], [3], but found them suboptimal for our needs. Based on TNC tests in noisy ham links, different models show PER ∼15-30% at BER >1% without FEC or compression, which is not plausible for our objectives [14]. The AX.25 protocol has a fixed 256-byte frame that imposes bulky headers with 16+ bytes for callsigns and substation IDs, which is inefficient for blockchain compatibility [2]. Similarly, APRS operates at 1200-baud audio frequency-shift keying (AFSK) on 144.390 MHz(typically done in the US). This works well in one-way signaling (simplex mode) but fails to provide proper function for reliable directed messaging, as it only offers limited sequencing and low throughput [64]. This restricts embedding blockchain elements like 32-byte Merkle roots [60]. Even though its digipeater routing is effective for multi-hop but does not allow native ZK verification. This results in exposing risks in decentralized relays.

In response to all these issues (Table 4.1), our custom protocol can be used as an alternative for blockchain integration. Considering that the system uses fixed 124-byte packets (23-byte header + 101-byte payload) with big-endian structure packing (for unique ID and segment number) to ensure portability across all platforms. The header part of the packet incorporates padded ASCII callsigns (7 bytes) for regulatory identification. It is a global unique ID that can be adapted into hashes by using a preferred algorithm. This design supports blockchain by reserving payload space for segmented zk-proofs and includes CRC-32 (polynomial 0x04C11DB7) for efficient channel error detection without FEC. Our GNU Radio simulation validates this choice by achieving 99.9 % reassembly success at 0.1% bit error rate (BER) in the simulation, surpassing AX.25's simulated 85% under identical NumPy-based noise conditions. The output is a clear protocol specification diagram: ID (4B) + Sender (7B) + Recipient (7B) + Segment (4B) + Total (1B) + Chunk (101B), positioning our work as a novel bridge between ham traditions and modern decentralization. Based on Terminal Node Controller (TNC) tests in noisy ham links, different models show Packet Error Rate (PER) 15-30% at Bit Error Rate (BER) >1% without Forward Error Correction (FEC or compression), which is not plausible for our objectives.

Table 4.1: Comparison of Zero-Knowledge Proofs(Based on Packet Overhead, Error Resilience and Extensibility)

zk-SNARKs	Bulletproofs	zk-STARKs
Packet Overhead is Extremely low. zk-SNARKs are a form of zero-knowledge proof that are known for their efficient structure. They generate small-sized proofs that remain constant even though the computation verification process is complex. This key feature gives them a major advantage by making zk-SNARKs highly suitable for blockchain applications where storage space and transaction bandwidth are limited.	**Compact Packet Overhead.** Bulletproofs are very efficient proofs that stay small even as the computation grows. Their size increases only a little because it scales logarithmically with the number of witnesses. This makes them significantly more compact than zk-STARKs, helping to reduce the extra data that needs to be sent or stored, and resulting in smaller packet overhead, especially for range proofs.	**Packet Overhead is High.** zk-STARKs create larger proofs compared to zk-SNARKs and Bulletproofs. The outcome requires more data to be sent, leading to higher packet overhead. However, there are drawbacks for Internet of Things (IoT) networks because efficient data transmission is important for low bandwidth
Error resilience depends on the trusted setup. The security of zk-SNARKs depends on a trusted setup. The setup creates public parameters for the proof system. If it is compromised, an attacker could use it to generate fake proofs. Newer protocols like PLONK have improved the process by creating a universal setup that can be reused in multiple applications.	**Inherently resilient to errors (Trustless).** Bulletproofs rely on cryptographic primitives that have strong security assumptions. They don't require a trusted setup, resulting in the reliance method preventing the risk of a compromised setup. This removes a single point of failure and makes Bulletproofs tougher to manipulate.	**Inherently resilient to errors (Trustless and Post-quantum).** The transparent zk-STARKs do not need a trusted setup. These are also considered quantum secure as they rely on collision-resistant hash functions. This particular aspect provides a strong, long-term error resilience
Low Extensibility. Several zk-SNARK frameworks require a trusted setup for every new application. Redesigning the circuit or creating a new setup is costly and limits extensibility. Universal SNARKs have refined this issue by allowing one-time setups.	**High Extensibility.** Bulletproofs can be used for any arithmetic circuit as they are highly flexible. These are specifically effective for authenticating secret values within a particular range (range proofs). This phase is done without any trusted setup, which makes the bulletproof highly scalable for different applications.	**High Extensibility.** zk-STARKs can be implemented in any function that can be used as a computational integrity argument. These are highly versatile and are considered a general-purpose primitive for creating proofs that can be used in arbitrary computations. Moreover, zk-STARKs are an essential part of zero-knowledge Virtual Machines.

4.1.3 Phase B: Integration Analysis

This part of our research evaluates how the custom protocol components, like SDR simulation and packet structure, interact with blockchain elements. We employed a Strengths, Weaknesses, Opportunities, and Threats (SWOT) analysis and UML sequence diagram to map protocol data unit flows, which allows us to integrate this phase with the DSR iterative "build–evaluate" method [32]. In this phase, cross-layer compatibility across OSI Layers 1-3 for radio transport and Layer 7 for blockchain applications is verified. The Socket PDU (UDP port 54321), used in the model, acts as a P2P bridge that connects to the blockchain network nodes, where packets identified using unique IDs are entered as payloads.

Buffering in Unique ID TX/RX blocks will also be used to simulate how blockchain nodes handle such packet inputs. The CRC Append/Check provides compatibility of hashing data integrity by detecting physical-layer errors and other logical errors. Since amateur radio is low-bandwidth, we formulated our protocol in such a way that it can be customized. It is easy to alter the way data can be divided and how quickly it can be transmitted to suit future requirements. This way, other functionalities, such as blockchain, can be added to the system easily. We did tests, whereby the system manages errors successfully and is ready to be simulated in the future, such as Universal Software Radio Peripheral (USRP). The computation load is eliminated through the regulatory focus on the open, unencrypted transmissions. We are considering adapters of GNU Radio in order to connect with the current standards. The simulation results indicate the opportunities in the development of lightweight proofs, such as the Mina zk-SNARK proof, to allow radio verification. Through converting the evidence into a hashed transaction tag, the repeaters would have the capacity to ensure the authenticity with minimum overhead by not necessitating the arduous chain sync and employing the effective multi-hop trust.

4.1.4 Phase C: Blockchain Network For Radios

This part of our research focuses on evaluating how the Blockchain network is working for amateur radio operators. We plan to run simulations on Docker containers to evaluate our blockchain network's computational and network performance. In this phase, the blockchain network will be implemented and simulated using three different core nodes: producers, repeaters, and receivers. which will be responsible for running the whole blockchain, like creating blocks, proposing blocks for signature, collecting signatures, establishing sessions for amateur radios, and enabling real-time communication for sender and receiver radios. The main idea is to design a network where integrity and availability mean the data cannot be tampered and the connection has to be capable of real-time communication. Security is ensured, but privacy will be compromised as our radios cannot implement encryption according to FCC Part 97 regulations. This system is built to create a trust model that can guarantee that the senders and the receivers in this network are authentic and authorized.

This blockchain model is a dual-plane model with control and data planes. For the control plane node compatible packet structure will be followed for block proposal, signature, block finalization, and block acknowledgement, where the producer will create the proof using a zero-knowledge proof and convert it into proof_hash to limit the block size and make it compatible for radio bandwidth and propose the repeaters via broadcasting.

The authorized repeaters and receivers in the network will independently verify the hash, accept the proposal, and sign digitally. The producer will collect the signatures and create the final block. The finalized block will be broadcast to the repeaters and receivers for creating a session; all nodes will update the final block in their caches with a brand new session. This will create an honest majority and create a cryptographic hardness using SHA256, BLAKE2b, and Ed25519. For the data plane, a live session packet transfer will be ensured using a different packet structure from the control plane. This 112B overhead packet structure will make sure that all nodes can check and safely push the packet as is to make sure that the payload is not lost. Such architecture will enable the repeaters to verify the session validity of the packet, proof-hash binding, and HMAC tag of the packet based on cache look-up. This makes the packet tamper-proof. In addition, timing attacks and race conditions will also be mitigated in the blockchain network.

4.1.5 Future implementation

Blockchain technology distributes the same copies of the ledger to the network members. The given feature eliminates the threat of single-point failures and increases the sustainability and accessibility of the data to the authorized users. Although all data transactions can be verified easily since databases are managed by a single entity, this is not possible in the decentralized model. To use these advantages to trace radio communications, our protocol needs a special ID in the stream of data. In order to address this challenge, we intend to design our own version of the packet, including an identifier to enable it to use blockchain primitives such as proof verification and hash of transactions [18]. It is possible to establish a structure in which radio packets will produce verifiable events, and the blockchain will offer an accessible path by connecting the packet fields with the ledger requirements. This compatibility will not only help us to satisfy our need for decentralization but also help to extend the standards of amateur radio to the networks around the entire world. In the next phase, we will develop a zk-proof-based session to verify packets, where we plan to use repeaters as verifiers to transmit RF signals. Mina's small 22KB blockchain is powered by zk-SNARKs, which is ideal for wireless constraints that support millisecond verifications on resource-limited devices for IoT nodes like Raspberry Pi repeaters. Through a zkApp smart contract, each reassembled message will become a verified transaction using its unique ID as a tx packet. Repeaters with USRP SDRs and Raspberry Pis will demodulate packets and rebroadcast them. Through using GNU Radio and Mina's zk-SNARK, the network will achieve low-bandwidth verification rather than full block translation, which will be the bridge between radio systems and blockchain technology. GNU Radio will also be used for SDR simulation.

After implementing the blockchain network, the next phase will focus on integrating all the components of the system. We plan to integrate GNU Radio for SDR simulation and Mina for Blockchain. The Security Analysis part will deliver a complete vulnerability analysis of the integrated system. This phase will measure whether the blockchain ledger authenticates only legitimate packets or not. In this section, we intend to assign precedence of availability and integrity within the various constraints and apply formal methods to guarantee fewer negative impacts. As well, to make sure that the assessment is proper, we intend to implement the STRIDE threat model in our system, which will address the following threat categories: spoofing, tampering, repudiation and information disclosure, denial of service, and elevation of privilege.

In the following stage of the design, performance of the protocol shall be tested with

simulated testing, using the amateur radio requirements. Once these are satisfied, we will proceed with the simulations of GNU Radio to acquire the simulation of RF channels using a port and containerize the systems using virtualization. The full reassembling of the packet and zk-SNARK generation will be checked by different functional tests. The other ham-specific tests will be transmitter identification and multi-hop relaying, which will be compared to the current radio systems in order to confirm the reliability of the blockchain. For the final phase, we will expand the prototype into a sustainable mesh network with embedded test methods to ensure reliability and scalability.

Lastly, for the final phase, we will expand the prototype into a sustainable mesh network with embedded test methods to ensure reliability. O1js will be used to connect edge devices and repeaters, and will have a decentralized architecture, and monitoring tools will keep track of packet verification of the radio on the blockchain. The methodology in general manages the conceptual choice of operational implementation and demonstration of the blockchain-enriched radio protocol. In addition, it also provides the roadmap to resilient communications in the future. [5], [8].

4.2 Preliminary Design or Design (Model) Specification

This section of the paper gives the outline of the preliminary design requirements of the packetized radio communication system. Three of the alternative models were created with the help of GNU Radio Companion (GRC) in a visual programming tool, software-defined radio (SDR). Such models point to the cyclic evolution of simple transmission of packets to sophisticated segmentation and voice modulation. We developed this by bearing in mind our goals of sound data transfer in noisy channels and the ability to have future prospects of integration of blockchain. To make sure that this process is flexible and robust, we did not hesitate to test various alternatives, and these enabled us to check the success rate of the phases by using simulations. The evaluation of these simulations was achieved with key performance measures such as packet error rate (PER), bit error rate (BER), and the successful percentage of end-to-end delivery. The simulations were done using a loopback environment with the specific ports, with some controlled noise to simulate the interference and barriers in the real world. This particular measure confirmed the opportunity of integrating a verifiable and decentralized communication protocol among the models.

4.2.1 Packet Radio Model V.4: Basic Packet Transmission

Packet Radio V.4 (Figure 4.4) is aimed to be a proof of concept that is centered on text-based messaging and includes error detection. The transmit (Tx) chain receives the input of the user and converts it into packets. Once the packets have been made, it calculates the error, simulates channel noise, and sends the data using UDP. The receiver (Rx) then verifies the errors of the packets entering it and extracts the message to proceed with it.

Transmit Chain Description: At the start of the Transmit chain, the text is read into the system through the input (stdin), then formatted into a Protocol Data Unit (PDU) using the Polymorphic Type Management (PMT) system of GNU Radio, using the format: [recipient] de [sender] [text]. In this block, the use of threading of non-blocking I/O is provided with real-time interaction. As the next step, it appends a 32-bit

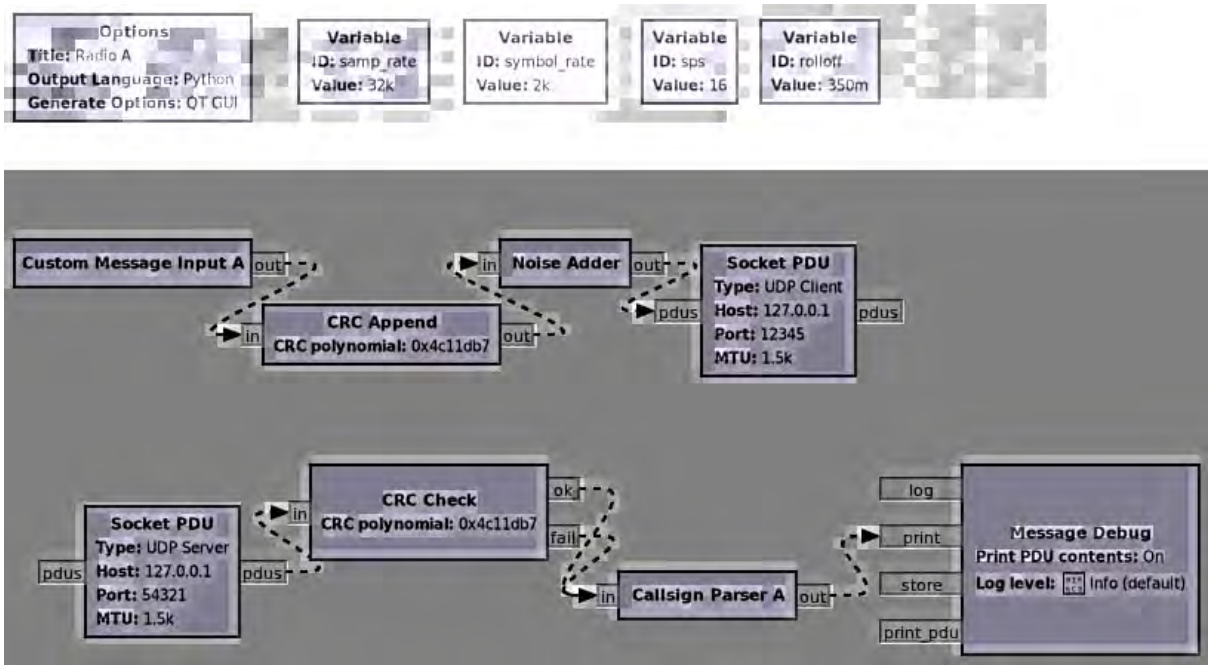

Figure 4.4: Packet Radio v.4

Cyclic Redundancy Check (CRC) checksum to the data to identify faults and errors by using the standard polynomial 0x04C11DB7. Every GNU Radio documentation has this block computing the CRC of the entire PDU, which enables error detection with a high probability. Custom blocks are employed to introduce impairments in channels by flipping bits randomly, and this can be implemented using a noise adder. NumPy is used to simulate a Binary Symmetric Channel (BSC), in which the bit error rate can be changed (the default is 1%) to learn how this system can behave in a noisy environment. Lastly, the Socket PDU transmits the packet using UDP to a local port, which works as a client in the testing loopback.

Receive Chain Description: The role of the receive chain starts with the block of the Socket PDU. This particular block is utilized as a UDP server at a given port, which transforms the received UDP packets to PMT PDUs. They are sent to the CRC check to check the integrity of the packets. In the verification, only packets that have valid checksums are sent, and damaged packets are lost. This is followed by the Callsign Parser, which parses sender/ receiver callsigns using regular expressions, error handling, and displays parsed callsigns. The PDUs are then logged in the message debug block to be analyzed and authenticated. Then, to check on the simulation, a loopback test on the GNU Radio Companion (GRC) sends 100 messages with BER between 0-5%. Our result at 1% BER was 98% success, and this attests to the reliability of the model.

4.2.2 Packet Radio Model V.5: Enhanced Segmentation and Identification

Improving on the basics of V.4, Packet Radio Model V.5 (Figure 4.5) introduces unique identifiers and message segmentation. These are used for handling larger payloads, improving traceability, and reliability in multi-segment transmissions.

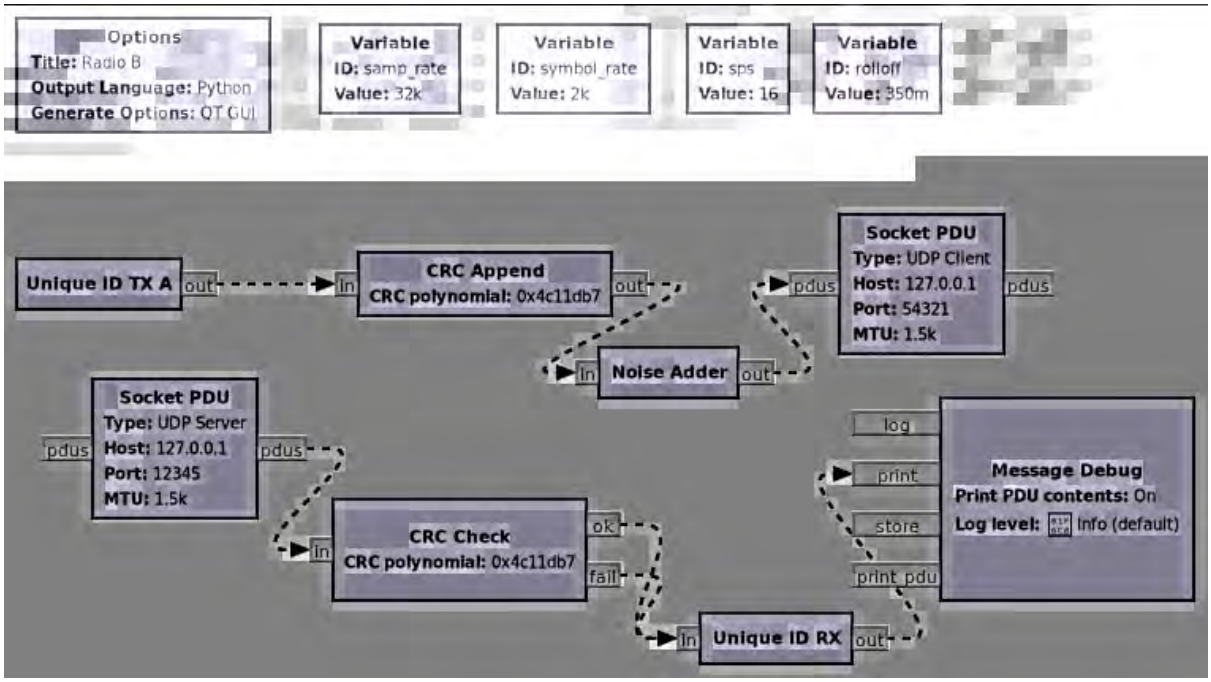

Figure 4.5: Packet Radio v.5

Transmit Chain Description: In the Transmit chain of this model, the Unique ID TX creates a custom block that chunks input into 101-byte segments, adds a 23-byte header (unique ID, callsigns, segment info), and outputs PDUs with throttling (0.1s sleep). This process enables ordered reassembly in the system. Then CRC Append, Noise Adder, and Socket PDU work the same way they did in the v.4 model.

Receive Chain Description: In the Receive chain, Socket PDU and CRC Check functions exactly as they did in V.4. After completing those steps, Unique ID RX creates a custom block parsing headers, buffering segments by unique ID. After completion, it reassembles them with validation checks, and the Message Debug part follows the same process used in V.4. To verify the simulation, we tested the model with 50 multi-segment messages (up to 1KB), achieving 97% full reassembly at 1% BER. This shows that it handles message fragmentation better than version 4.

4.2.3 Packet Voice Radio Model V.6: Voice Modulation with Channel Simulation

The third alternative (Figure 4.6) shifts to voice communication, incorporating modulation for realistic RF simulation, which is suitable for extending to blockchain-logged audio calls.

Transmit Chain Description: In this model, an audio source captures microphone input at an 8 kHz sample rate, which produces floating-point samples. A rational resampler then modifies the sample rate to meet the requirements of the modulator. The floating-point data is converted to 8-bit integers by a Float to Char block, then unpacked into individual bits for modulation. These bits are then encoded into the constellation points with the help of a Constellation Modulator, such as QPSK, which uses a differential encoding and RRC filtering. The modulated signal is sent through a Channel Model, where noise is added (50 mV), and finally it is sent via a UDP Sink.

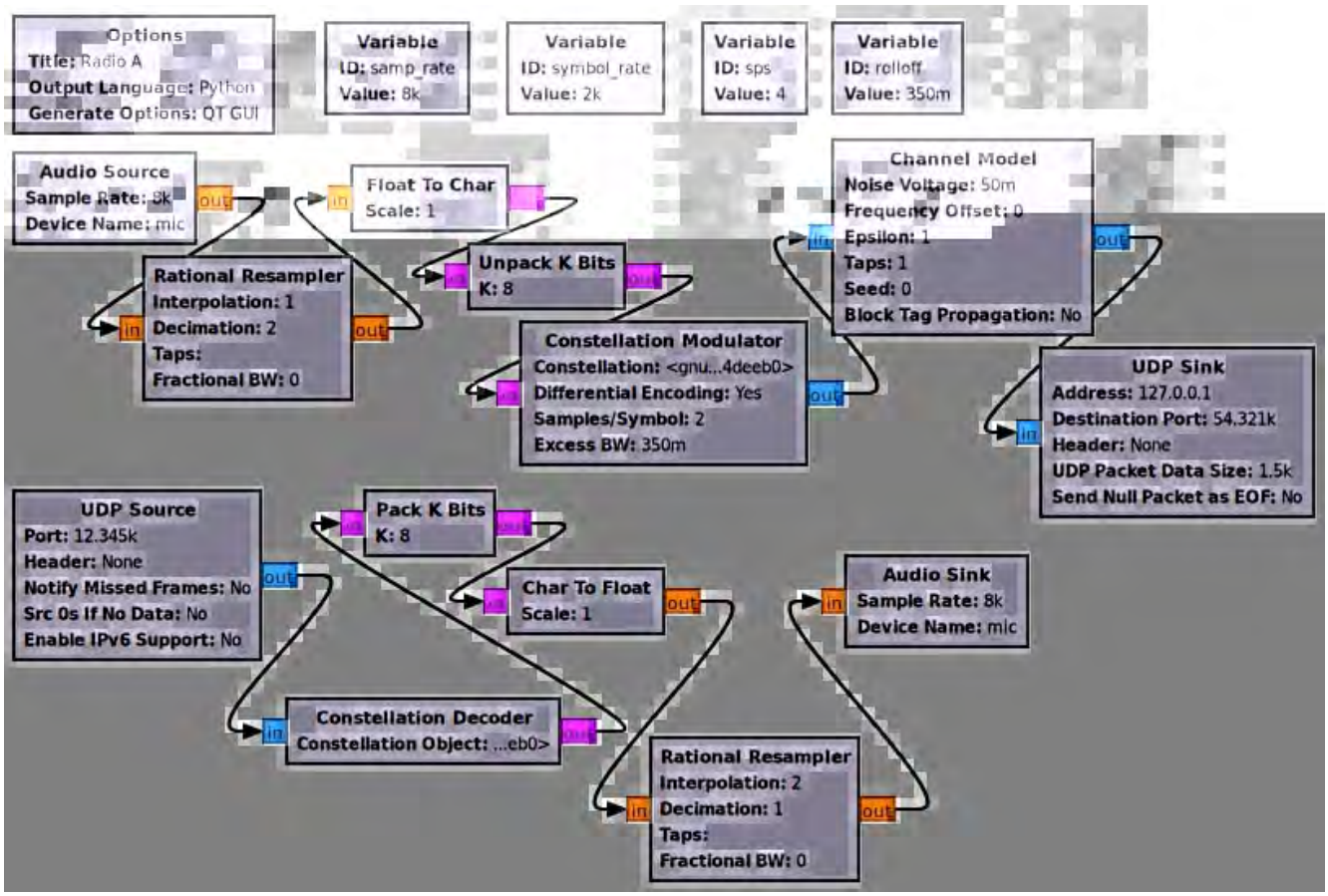

Figure 4.6: Packet Voice Radio v.6

Receive Chain Description: In this case, UDP packets are received at the UDP Source. The constellation decoder is used to decode bits into symbols with the help of the constellation map. Then that Pack K Bits packs bits into bytes. The Char to Float converts chars back to floats, and the Rational Resampler (Decimation) downsamples to meet the needs of audio output. And lastly, the Audio Sink is used to generate outputs to speakers at 8 kHz. The flowchart demonstrates these connections with samp_rate (8k) and symbol_rate (2k) variables. Audio clips (10s duration) were transmitted with varying noise to verify the simulation, where the SNR was 10 dB. Intelligible recovery was achieved with <5% distortion, verifying modulation efficacy.

These designs follow our initial approach while showing the progressive improvements in reliability and flexibility. In the future, we intend to integrate blockchain for verifiable message logging by building on this radio communication base.

4.2.4 System Design: Blockchain-Enabled Radio Network

In this section, the iterative design and implementation of a blockchain-enabled radio mesh network with cryptographic authentication are discussed. We have created the system by analyzing and experimenting with four different versions of the blockchain-enabled network. And we have included all of the versions addressing specific limitations while progressively making improvements upon achievements found beforehand.

In this part of the research, we have compiled all of the versions that cover certain limitations, and we have continued to apply improvement on the achievements that we

have made. Version 1 (4.2.5) sets the basic architecture and confirms the fundamentals of networking. In version 2 (4.2.6), the cryptographic security incorporates authentic zero-knowledge proofs into the system. Version 3 (4.2.7) uses session-based optimization to enable real-time performance after zk-proof integration, and finally, Version 4 (4.3.1) uses decentralized blockchain consensus without sacrificing real-time functionality. Design objectives, system architecture, and algorithmic specifications are presented in each of the versions, and limitations are identified to encourage further improvements.

4.2.5 Blockchain-Enabled Radio Network Version 01: Baseline Implementation

Design Objectives:

The purpose of this version was to create a baseline architecture for a blockchain-enabled radio-mesh network system. This implementation acts as a proof-of-concept system by showcasing the basic principles of cryptographic packet verification in multihop radio communication. It validates the core network architecture and establishes baseline performance metrics for subsequent enhanced versions. The current system applies a producer-repeater-receiver architecture where the packets traverse in a linear chain of verification nodes before reaching the destination. Each of the nodes performs integrity verification by using cryptographic hash functions. In this form of the system, there is a zero-knowledge proof mechanism of metadata-based authentication. However, the JSON metadata that is used as its proof is not actual cryptographic proof but simplified versions. This tradeoff has been chosen purposely to focus on system validation and quick development.

The implementation of the baseline provides a guarantee that network architecture validation is obtained by a sound multi-hop packet forwarding mechanism, where integrity verification is done by each node. A performance baseline is established through the measurement of throughput, latency, and resource utilization, which are used to maintain an incremental development base. A modular design is also designed to accommodate incremental improvement that may take place in subsequent versions. Finally, containerization is employed to provide consistency in deployment across various hardware platforms in order to provide a reproducible testing environment.

System Architecture:

The system consists of three primary node types, each serving a distinct role in the packet lifecycle.

Producer Node: The Producer serves as the network entry point, receiving messages from Radio A via UDP and creating packets with integrity metadata. After receiving a message, the Producer generates a unique packet identifier and computes a SHA-256 hash of the message data. Then, attaches simplified "zero-knowledge proof" metadata, which is actually JSON-formatted information including the packet UID, data hash, and timestamp. The packet is then sent to the first repeater node via TCP.

Repeater Nodes: The verification chain consists of repeater nodes. Each of them performs integrity checks and forwards packets toward the destination. Every repeater maintains a set of seen packet UIDs to prevent duplicate propagation. Due to this,

the system generates O(1) duplicate detection with O(m) space complexity, where m represents the number of unique packets processed. Whenever a packet arrives, the repeater verifies its data hash by matching it with the computed SHA-256 hash and checks the presence of proof metadata. If the verification process turns out to be successful, the repeater updates the packet's repeater ID and hop count before forwarding it to the next node in the chain.

Receiver Node: The Receiver does the final verification with a terminal integrity check by recomputing the data hash and comparing it against the packet's claimed hash value. After successful verification, the Receiver formats the message with packet metadata and transmits the authenticated messages to Radio B via UDP.

Network Communication:

The existing system applies Docker networking (172.25.0.0/16 subnet) to establish closed communication routes among nodes to establish uniform testing environments. TCP sockets, in this instance, are used in the reliable data transfer between the nodes. This enables adequate message sequence and packet delivery. UDP sockets are linked with radio equipment so that they can accommodate low-latency transmission that matches the radio interface.

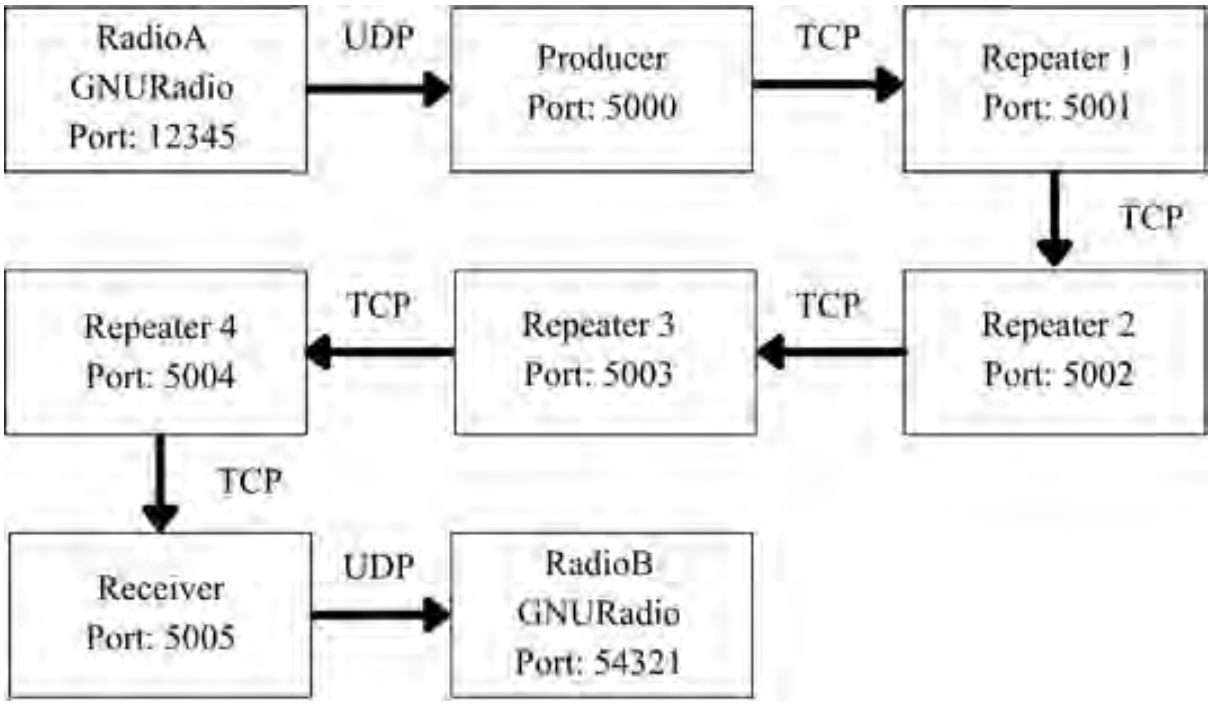

Figure 4.7: System Overview V1

System Components:

Packet Structure: Packet Structure: Each packet contains nine fields that support current functionality while accommodating future security enhancements. These are -

- uid: Unique packet identifier for duplicate detection

- sid: Source node identifier
- rid: Current repeater node identifier
- data: Message payload
- data_hash: SHA-256 hash of message data
- timestamp: Packet creation timestamp
- zk_proof: Placeholder for zero-knowledge proof (JSON metadata in V1)
- signature: Placeholder for digital signature (unused in V1)
- hop_count: Number of network hops traversed

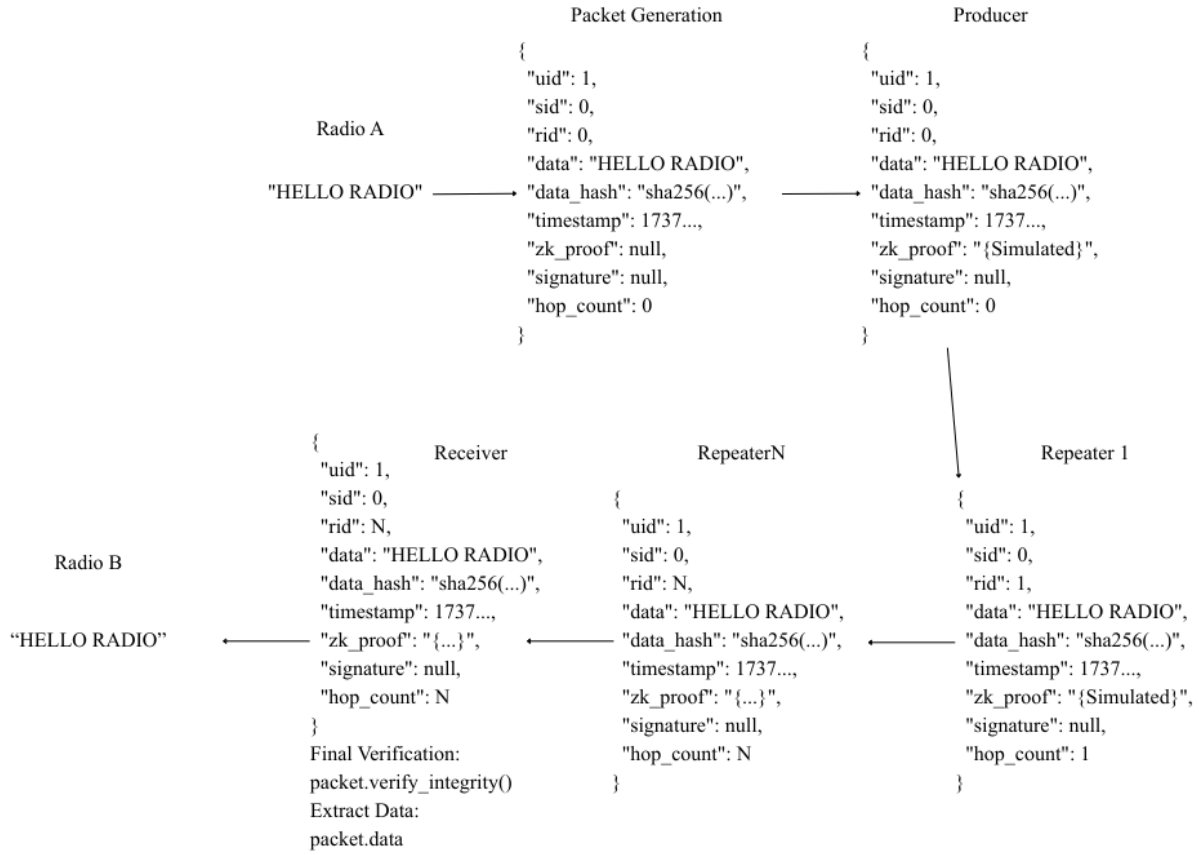

Figure 4.8: System Flow V1

End-to-End System Operation:

The end-to-end system (Algorithm 1) operation follows a structured protocol spanning four phases.

Phase 1: Network Initialization: All nodes initiate their communication interfaces and listening sockets. After the initialization, the Producer node binds a UDP socket to the radio interface and sets the first repeater as its next hop. Each repeater initialized a TCP listener in its designated port and identified the next hop. This hop can either be the subsequent repeater or the final Receiver. Lastly, the Receiver node also maintains a TCP listener for incoming traffic, along with a UDP socket for outbound transmission to Radio B.

Phase 2: Packet Creation and Authentication: When a message M arrives from Radio A, the Producer generates a unique identifier and timestamp and computes data_hash = SHA256(M). After that, the node creates the packet structure with all necessary metadata fields and generates a simplified proof metadata containing {uid, data_hash, timestamp} in JSON format. After completing the authentication, the Receiver transmits the packet to the first repeater via TCP and waits for acknowledgment before processing the next message.

Phase 3: Multi-Hop Verification and Forwarding: At each repeater node R_i, duplicate detection is done by checking if the packet.uid exists in the seen set, and then it computes hash = SHA256(packet.data) to verify Data Integrity. Then the node checks for the presence of the zk_proof field and updates the Packet Metadata by setting packet.rid = R_i.node_id and incrementing packet.hop_count. Lastly, if the next hop is configured, then the repeater transmits the packet via TCP and awaits acknowledgment.

Phase 4: Final Verification and Delivery: At the Receiver node, hash=SHA256 (packet.data) is recomputed and hash=packet.data_hash is verified to ensure terminal integrity. Along with this, the node also records packet UID, hop count, and timestamp for performance analysis and creates an output message as "[Packet {uid}] data" for delivery. And as the final step of delivery, the formatted message is transmitted via UDP to the destination radio interface.

Error Handling: The system implements comprehensive error handling for various failure scenarios. First of all, it logs security events, rejects packets, and notifies the previous hop with 'FAIL' for integrity violations and failed proof verification. To pinpoint Duplicate Packets in the transmission, the system logs an event and notifies the previous hop with 'DUP'. And considering the worst-case scenario, if the network fails, the system logs a transmission error and attempts to limit retransmission.

Algorithm 1 Verified Multi-Hop Radio Protocol

Initialize: $\forall n \in \{P, R, D\}$: bind sockets; $P.hop \leftarrow R_1, R_i.hop \leftarrow R_{i+1}, R_n.hop \leftarrow D$
procedure CreateAndTransmit(Message M) ▷ Producer P
    $pkt \leftarrow \{uid : \text{gen}(), data : M, hash : \text{sha256}(M), ts : \text{now}(), hop : 0\}$
    $pkt.\pi \leftarrow \text{GenZK}(pkt.uid, pkt.hash, pkt.ts)$
    return TCP_Send(pkt, $P.next\_hop$) = ACK ? SUCCESS : FAILURE
end procedure
procedure VerifyAndForward(pkt, R_i) ▷ Repeater R_i
    if $pkt.uid \in seen$ **then return** HandleError(DUPLICATE, pkt)
    end if
    if $\text{sha256}(pkt.data) \neq pkt.hash \vee \neg\text{VerifyZK}(pkt.\pi, pkt)$ **then**
        return HandleError(INTEGRITY/ZK_FAIL, pkt)
    end if
    $seen \leftarrow seen \cup \{pkt.uid\}$; $pkt.rid \leftarrow R_i.id$; $pkt.hop \leftarrow pkt.hop + 1$
    return TCP_Send(pkt, $R_i.next\_hop$)
end procedure
procedure FinalizeDelivery(pkt) ▷ Receiver D
    if $\text{sha256}(pkt.data) \neq pkt.hash$ **then return** HandleError(INTEGRITY, pkt)
    end if
    LogStats(pkt); $out \leftarrow \text{format}(pkt.uid, pkt.data)$
    return UDP_Send(out, $D.destination$)
end procedure
procedure HandleError($type, pkt$)
    match $type$: **Network** → Retry(pkt); **Default** → Reject & NotifyPrev()
end procedure

Limitations:

This initial system focuses on rapid development and functional validation over cryptographic security, which results in several limitations.

Cryptographic Deficiencies: The most notable weakness of this version is that of authentic zero-knowledge proofs. The generate zk proof() command generates just metadata in the form of a JSON, but not encrypted proofs. These evidences are easily falsifiable and does not have a cryptographic commitment to packet data. It is on this account that these "Proofs" are not able to guard against replay attacks and any other types of security violations. Also, the field of digital signatures is unused. Since the packets do not have any signatures, they do not generate or verify any signatures, and the identity of the sender can not be cryptographically verified.

Hash-only integrity checking also creates vulnerability in the system. While SHA-256 detects accidental corruption, it offers no sender authentication. Due to the lack of authentication, an attacker can intercept a packet. He can then modify its contents, recalculate the hash, and successfully transmit the tampered packet through the network.

Architectural Constraints: The linear chain topology causes several operational limitations due to its architecture. It lacks broadcast capability, which makes efficient multi-recipient messaging impossible. A single node failure breaks the entire chain of communication without any redundant paths available. As a result, during high load, potential

bottlenecks occur as traffic cannot be distributed. Also, the fixed chain structure does not reflect actual radio mesh network topologies because in a realistic system, nodes are connected in irregular, overlapping patterns based on physical proximity and signal propagation.

Integration Gaps: Despite the initial plan of integrating zero-knowledge proof in the system, Version 1 keeps several components disconnected. The zk-proof components are built using the o1js framework, but they are isolated from the Python networking system. The design also includes support for Mina Protocol features, yet the current implementation processes data without using these attributes. Session management is absent as there is no secure connection establishment, no key exchange mechanism, and no tracking of ongoing conversations between nodes.

Design Justification:

The limitations of this version are intentional design choices rather than errors. The primary goal at this stage was to validate that the basic networking functions work correctly through different phases. It would be difficult to debug such a full cryptographic system all at once, and it is also impossible to detect performance bottlenecks is such structures. This is why we have maintained the networking and cryptography functionalities independently. And architecture establishes integration points to be used in future security improvements. The progressive development method minimizes risk by developing a version that is based on already proven components. Version 1 ensures that the network is capable of transferring packets across a number of hops with sufficient amounts of reliability and is able to detect corruption. Version 2 will include some real cryptographic protection, and Version 3 will develop to realistic mesh networking, including broadcast support.

Conclusion:

Version 1 already shows that packet verification based on blockchain can be implemented in a radio mesh network. The implementation enables sound end-to-end deliveries of packets, avoidance of propagation of duplicate packets, checking of data integrity at each network hop, and a stable operation in a containerized Docker environment. There are also baseline performance metrics in the system that are measurable and are not bound by cryptographic overhead. These metrics present a point of reference to gauge the effect of security improvement on performance in future versions. While security and architectural limitations exist, these gaps define the strategy for future development. The next version of the system resolves cryptographic deficiencies by implementing authentic zero-knowledge proofs and digital signatures on the networking foundation established in Version 1, and it also preserves the established packet structure and forwarding mechanisms.

4.2.6 Blockchain-Enabled Radio Network Version 02: Enhanced System with Cryptographic Zero-Knowledge Proofs

Design Objectives:

The second version of the network system aims to resolve the critical security limitations that were detected in the baseline implementation. This version applies authentic zero-knowledge proof generation and verification using the Mina Protocol and the o1js framework. It replaces the simplified JSON-based proof metadata with cryptographically valid zk-snark proofs and also provides mathematical proof of data authenticity and sender legitimacy without revealing sensitive information. The central architectural improvement for this version was the integration of a TypeScript-based zkApp subsystem, which generates and verifies the zk-proofs. This particular step allows proper cryptographic security while maintaining the established network system from Version 1 (4.2.5).

Improvements in Version 2:

The earlier version had its fair share of major limitations, including explicit cryptographic limitations, architectural limitations, and integration gaps. Gradually, this version solves the cryptographic and integration problems. Version 2 will also seek to reform the network into a usable prototype with a cryptographically secure system by introducing authentic zero-knowledge proofs and enforcing detailed security policies.

Authentic Zero-Knowledge Proof Generation: The existing network system produces the actual zk - SNARK proofs with the help of o1js. This gives a mathematical demonstration of the authenticity of packets. Poseidon Hash Function is applied as an alternative to SHA-256. This is an optimized cryptographic hash function, a zk-SNARK that was developed as a special-purpose hash function to implement zero-knowledge circuits with high efficiency. Poseidon offers the required security features such as collision resistance and preimage resistance. It is also able to simplify the circuits and optimize the zk-SNARK arithmetic circuits to be eight times more efficient compared to SHA-256 implementations. O1JS signature generation is also added to the system to address the problem of digital signatures, with the help of elliptic curve cryptography. In such a manner, every packet is cryptographically signed with an authentication of the identity of a sender, without any deployment of any public key infrastructure. In this version, RadioMeshIntegrity smart contact is used to enforce holistic security policies as well as timestamp validation, sender authorization, and replay attack prevention. This version meets several security properties because of these changes. It enables tampering to be detected by verifying the integrity of the data by using cryptographic binding, authenticating the sender by use of a digital signature, which prevents impersonation. The system is also guaranteed to have authenticity with unforgeable signatures and has protection against replay attacks with unique identifier tracking.

zkApp Integration Architecture: The enhanced network platform is made up of three tiers architecture. The first one is the Python Networking Layer. The second layer is the TypeScript zkApp Layer. This component of the system carries out cryptographic functions with the o1js platform. In this system, proof.js is used to create zk-SNARK proofs in outbound packets, verify.js is used to verify incoming zk-SNARK proofs, and finally RadioMeshIntegrity.js is used to specify the verification circuit logic and security

rules. The third and last level of the system is Mina Local Blockchain. This mimics a blockchain environment where proof generation is performed, and realistic cryptographic functions can be performed without the overhead of deploying a mainnet. This is achieved by means of communication between these layers in the form of the JSON-formatted subprocess calls that make it possible to integrate cryptography.

Enhance Packet Statistics: In Version 2, we used the PacketStatistics class to measure the cryptographic overhead. This measures the real-world performance cost of our security features. We're tracking key data like proof generation times, total latency, and how much extra weight the crypto metadata adds to each packet. We also monitor the success rate of verified traffic and log exactly why certain packets fail, whether it's due to tampering, duplicates, or invalid proofs.

Radio Interface Simplifications: In this second version, we replaced the GNU Radio flowgraphs with a lightweight Python-based setup. This was done to discard external dependencies and make automated testing possible. Since our baseline system is designed to validate the network architecture without physical radio signals, the full signal processing stack, like modulation and filtering, added unnecessary weight. By reducing them, we managed to shrink our container size from a bloated 800 MB down to a sleek 150 MB. This makes our deployment cycles much faster. The new radio modules still do exactly what they did before, but now they are easier to automate and far more efficient to work with.

Radio A enhancements include automated message generation with configurable transmission intervals, non-blocking reply reception using background threading, statistics tracking (messages sent, replies received), and a command-line interface for batch testing. And the Radio B enhancements are a continuous receive loop with formatted message display, optional auto-reply mode with randomized acknowledgments, visual formatting using Unicode box-drawing characters, and, lastly, packet ID extraction from formatted messages.

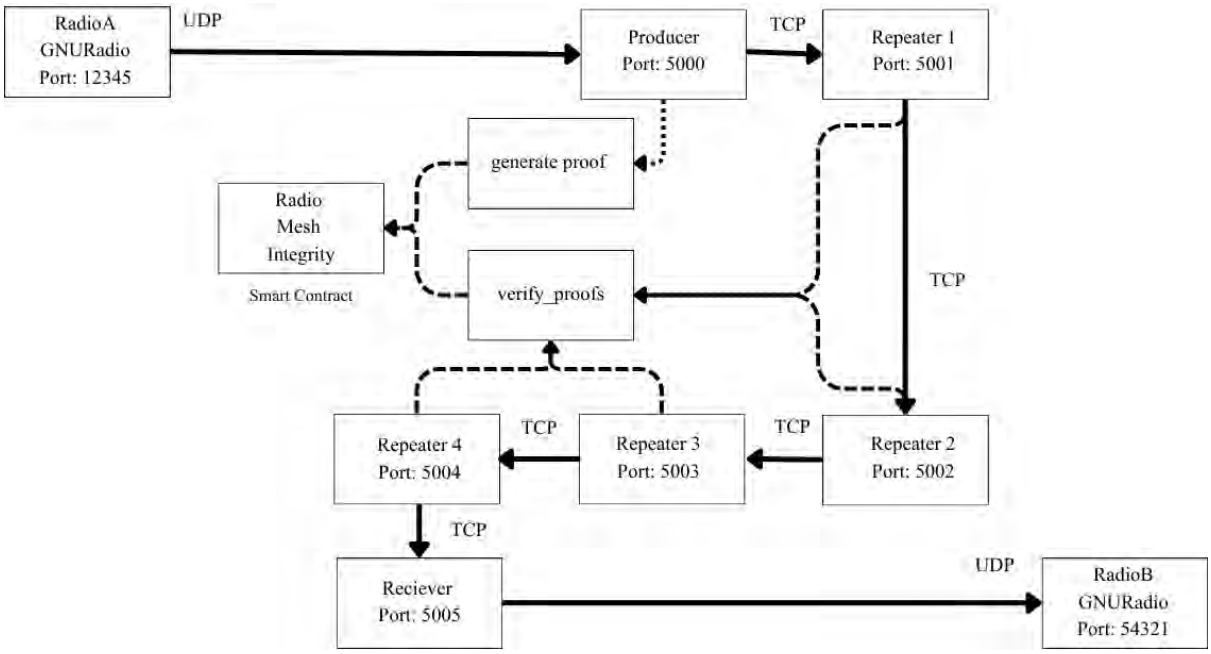

Figure 4.9: System Architecture

Modified System Components

Enhanced packet Structure: The packet structure remains unchanged from the previous version to maintain compatibility, but the zk_proof and signature field contains authentic cryptographic data in the current version.

Packet Structure $\mathcal{P}$:

`uid, sid, rid`	▷ *Identifiers (Unique, Source, Relay)*
`ts, hop`	▷ *Timestamp and Hop Counter*
`data, hash`	▷ *Message M and $SHA256(M)$*
`zk_proof`	▷ *JSON-encoded zk-SNARK (o1js)*
`signature`	▷ *Sender's Digital Signature*

End-to-End System Operation

Version 2 modifies the end-to-end system (Algorithm 2) protocol to incorporate authentic cryptographic verification while maintaining the networking structure from Version 1 (4.2.5).

For the first phase of the system, network initialization remains the same as Version 1, as all nodes establish communication interfaces, bind sockets, and configure next-hop routing in the same manner as before. For phase 2, an enhanced packet is created with cryptographic proofs. In this phase, when message M arrives from Radio A, the Producer performs authentic proof generation after creating a base packet structure by generating UID and timestamp, and computing hash_M = SHA256(M). The network transmission also remains unchanged from version 1 because this version forwards the packet to the first repeater via TCP as well. And the timeout mechanism in this phase ensures that the system remains operational even when proof generation encounters difficulties. And if somehow zkApp processing exceeds 120 seconds or fails, the system automatically falls back to simulated proofs. This maintains the network availability while logging the de-

graded security state. In the next phase, the Multi-Hop verification system is made more enhanced than the previous version. But the Repeater nodes perform authentic cryptographic verification through Duplicate Detection (unchanged from V1) and Hash Integrity Verification (unchanged from V1). Packet Update and Forwarding also remain the same as V1. The dual verification paths (simulated vs. authentic) enable progressive deployment. As a result, development and testing can proceed with fast simulated proofs while production systems use full cryptographic verification. And lastly, for the next phase, the Receiver performs comprehensive validation before delivery by doing a Terminal Integrity Check (unchanged from V1). This ensures enhanced final verification and delivery.

zkApp Integration: Cryptographic Subsystem

The TypeScript-based cryptographic subsystem handles all zero-knowledge proof operations, and it is isolated from the networking layer for security and maintainability.

Zero-Knowledge Proof Generation Workflow: The proof generation process follows a structured sequence optimized for the Mina Protocol. As the first stage of the workflow, it creates a Mina LocalBlockchain instance with proofs enabled, configures test accounts for deployment and signing operations. Then the subsystem compiles the RadioMeshIntegrity verification circuit ($\sim$ 30-60 seconds), generates zkApp address from random private key, and deploys contract to local blockchain with funded account. After deploying the contract, it converts packet fields (UID, source ID, data, timestamp) into zk-SNARK Field elements and computes Poseidon hash of message data for circuit-optimized integrity verification. In terms of completing the integrity verification, the process constructs a RadioPacket structure. This contains all packet metadata as Field elements and computes the packet hash as a cryptographic commitment of the complete packet state. Then, in the Digital signature generation part, it creates a signature over the packet hash using the sender's private key. After completing sender authorization and adding the signature to specific packet data, the process calls the verifyPacketIntegrity() method with packet, signature, and sender public key to complete the Smart Contract Transaction. In this state, the process also generates zk-SNARK proof of correct execution ($\sim$ 126 seconds) and serializes the proof to JSON format. And only after completing all these steps, the proof generation process returns a complete proof package. This package contains zk-SNARK proof, base58-encoded signature, sender public key, proof size in bytes, and generation time for performance monitoring.

RadioMeshIntegrity Smart Contract Logic: The verification circuit (Algorithm 3 enforces comprehensive security policies through circuit constraints.

RadioPacket Structure $\mathcal{R}$:

`uid`	▷ *Field: Unique packet id*
`sourceId`	▷ *Field: Originating node*
`dataHash`	▷ *Field: Poseidon hash of*
`timestamp`	▷ *UInt64: Creation time*

Algorithm 2 Enhanced zk-SNARK Multi-Hop Packet Protocol

procedure GenProof(packet p) ▷ Cryptographic Proof Generation
  try
    $p.\pi \leftarrow$ Subprocess("`gen_proof.js`", $p.input$, timeout $= 120s$)
    Log("Proof size: " + $p.\pi.size$)
  catchTimeoutError
    $p.\pi \leftarrow$ GenSimulatedProof(p) ▷ Fallback: Degraded security state
    LogWarning("Fallback simulated proof active")
  end try
end procedure
procedure VerifyProof(packet p) ▷ Cryptographic Proof Verification (enhanced)
  $pd \leftarrow$ JSON_Decode($p.\pi$)
  if $pd.type =$ "simulated" **then**
    return ($pd.uid = p.uid \wedge pd.hash = p.hash$) ▷ Basic testing fallback
  else ▷ Authentic zk-SNARK verification
    try
      $res \leftarrow$ Subprocess("`verify_proof.js`", pd, timeout $= 30s$)
      if $\neg res.valid$ **then** Log(res.reason); **return** FALSE
      end if
      RecordStats($p, res.time$); **return** TRUE
    catchException **return** FALSE
    end try
  end if
end procedure
procedure ValidateFields(packet p) ▷ Comprehensive Proof Validation (enhanced)
  $pd \leftarrow$ JSON_Decode($p.\pi$)
  if $pd.hash \neq p.hash \vee pd.sid \neq p.sid$ **then**
    LogError("Final verification: ID/Hash mismatch"); **return** FAIL
  end if
  RecordLatency(p, now()); **return** SUCCESS
end procedure
procedure HandleError(err, p) ▷ Error Handling Extensions
  match err:
    GEN_TIMEOUT $\rightarrow$ GenSimulatedProof(p) ▷ Degraded generation
    VERIFY_TIMEOUT $\rightarrow$ **Reject**(p); LogPerf("Timeout")
end procedure

Algorithm 3 RadioMeshIntegrity Smart Contract Logic

Function `hash`(p): **return** Poseidon.hash($[p.uid, p.sid, p.hash, p.ts]$)
procedure VerifyPacketIntegrity(p, σ, pk_sender)
    $lastUid \leftarrow$ this.lastVerifiedUid.get() ▷ 1. Replay Attack Prevention
    $p.uid$.assertGreaterThan($lastUid$) ▷ Enforce monotonic increase
    ▷ 2. Data Integrity Verification
    $computed\_hash \leftarrow$ this.hash(p)
    $p.dataHash$.assertEquals($computed\_hash$) ▷ Circuit-level equality
    ▷ 3. Digital Signature Verification
    $sig\_valid \leftarrow \sigma$.verify($pk_{sender}, [p.hash()]$)
    sig_valid.assertTrue() ▷ Confirm authenticity
    ▷ 4. Timestamp Freshness Check
    $now \leftarrow$ this.network.timestamp.get()
    $max\_age \leftarrow$ UInt64.from(300000) ▷ 5-minute threshold
    $(now - p.timestamp)$.assertLessThan(max_age)
    ▷ 5. Update Blockchain State
    this.lastVerifiedUid.set($p.uid$)
    this.totalVerified.set(this.totalVerified.get() + 1)
end procedure

Circuit Constraints: • $p.uid > lastUid$ ▷ Replay Protection • $p.dataHash \equiv$ Poseidon(p) ▷ Integrity Guarantee • Verify(σ, pk) = True ▷ Authenticity • $\Delta t < 300s$ ▷ Freshness Constraint

Zero-Knowledge Proof Verification: The verification of zk-proof operates faster than generation ($\sim$ 126 seconds). To ensure proper verification, the system parses JSON proof data and reconstructs the zk-SNARK proof object from the serialized format. After deconstruction, it decodes the signature and sender public key from base58. To verify the signature cryptographically before proof validation, the process reconstructs the packet hash from proof data as well. Finally, to ensure zk-SNARK validation, it applies the smart contract's quick verification method (optimized for fast checking) and returns the validation result with a detailed reason for any failures. This occurrence of slow generation, faster verification is inherent to zk-SNARK systems and is acceptable for the producer-repeater-receiver model, where one node generates proofs, and multiple nodes verify them.

Implementation Considerations

Cross-Language Integration: The main architectural challenge of Version 2 was combining Python networking code with TypeScript cryptographic operations. This integration process uses subprocess communication. In this particular communication system, Python serializes the packet data to JSON, and subprocess applies a Node.js script with JSON input. Meanwhile, the TypeScript zkApp will run cryptographic calculations, the Node.js script will write the result in the form of a JSON to the standard output, and Python will read the result, check it, and continue running. The maximum possible time to conduct a cryptographic calculation is 126 seconds, and the maximum possible time to verify a cryptographic calculation is 0.1 seconds; nothing will be stuck in an infinite

loop. Comprehensive logs will also be used to track everything; therefore, it will be easy to locate errors and track what is going on when something is misplaced.

Performance Challenges: In the second version of the system, generating a proof takes about two minutes on average, which is far too slow for real-time communication. To help with this, proof generation runs in the background. The Producer handles radio reception and proof creation separately, so incoming messages don't get stuck waiting for a proof to finish. If a proof doesn't work or takes too long, the system automatically uses a temporary simulated proof so that the network can keep on working without any interruptions.

There are a few ways to make this faster in the future. Proof batching could allow a single proof cover multiple packets, cutting down on repeated setup work. Proof caching also approves proof reuse for identical data patterns while avoiding unnecessary computation. Another option is proof delegation, where heavy cryptographic tasks are sent to specialized nodes, keeping the main network running without delays.

Cryptographic Design Choices: Cryptographic Design Selection: Version 2 uses both SHA-256 and Poseidon for different reasons. SHA-256 is compatible with Version 1, as it can do external integrity checks,s and Poseidon is designed to be used within zk-SNARK circuits. Local blockchain deployment will also exclude mainnet network latency and transaction fees. And since it can produce cryptographically identical proofs to mainnet proofs, this provides a testable environment for the network system.

Remaining Limitations:

Despite cryptographic improvements, some limitations persist. The long proof generation time makes real-time applications like voice communication and interactive messaging impractical with current performance features. This delay comes from circuit compilation, heavy computation, and cross-process communication overhead. As a result, the current design is better suited for store-and-forward messaging than live communication. Along with this, some earlier architectural constraints also remain. The system still uses a linear chain topology inherited from Version 1, which means it lacks broadcast support, has a single point of failure, and cannot take advantage of mesh network redundancy. In addition, the Python–TypeScript integration adds complexity. Information has to be serialised and distributed between processes, and this introduces overhead in addition to complicating debugging. Other options to mitigate the remaining limitations can be native Python zk-SNARK libraries, it can be implemented FFI bindings, or it can switch to a gRPC-based service architecture.

Conclusion:

On the whole, Version 2 manages to implement full zero-knowledge cryptography within the radio mesh network. The system is no longer based on simple metadata but on zk-SNARK proofs and digital signatures to check the authenticity of packets and their sending identity. Smart contracts are used to enforce security policies like replay protection and validation of timestamps. The combination of Python and the o1js architecture illustrates the fact that Mina-compatible cryptography can be used successfully in that architecture. It also has extensive monitoring systems, packet flow, delays, success rates,

and verification times, which can be used to analyze network performance. Besides, the replacement of GNU Radio can be used to make a setup easier, smaller in containers, and quicker in testing without losing functionality. The important accomplishments of this version are cryptographic authenticity, authentication of the sender using unforgeable digital signatures, automatic enforcement of security policy, and advanced monitoring of performance to carry out additional optimization.

4.2.7 Blockchain-Enabled Radio Network Version 03: Session-Based Authentication with HMAC Optimization

Design Objectives:

Version 3 solves the problem of critical performance constraints observed with Version 2 (4.2.6). And it does this by moving from per-message zero-knowledge proof verification to session-based authentication. In this method, the cryptography security foundation and on-the-fly message authentication are split apart. In this implementation of the network system, the cryptographic security establishment is only carried out once through zk-SNARK at the start of the sessions, and the real-time message authentication is effected by way of lightweight HMAC operations. The system is based on a proof hash binding algorithm that forms a cryptographic chain between the original zero-knowledge proof and all the consecutive messages. The good thing about this mechanism is that it has the performance of HMAC and a ZK-level security level. The first zk-SNARK proof generation requires around 126 seconds to set up the sessions, compared to Single packet processing in version 2. However, all the later messages can obtain the authentication latency of 0.1ms due to cache keys and HMAC checks.

Limitations of Version 2

Performance Barrier: Version 2 successfully establishes cryptographic security, but it has a critical performance bottleneck that prevents real-time applications. In this version, every packet requires 120 seconds for generating zk-SNARK proofs, along with ∼30 seconds for verification at each hop. For a 3-hop network path, the total verification time exceeds 210 seconds per packet. This latency makes voice communication, interactive applications, and real-time messaging completely impractical. Considering these data, a typical 10-minute voice call (approximately 6,000 packets at 10 packets per second) would require over 14,500 hours of proof generation time, which is clearly unusable for any real-time scenario.

Architectural Inefficiency: Architectural Inefficiency: The per-packet proof approach wastes computational resources. Each packet consists of properties (sender identity, data integrity, authorization) that remain constant across multiple messages from the same sender. Therefore, the cryptographic setup overhead, like circuit compilation, blockchain state initialization, and contract deployment, gets repeated thousands of times instead of a one-time session establishment.

Operational Constraints: Beyond performance, Version 2 inherits the linear chain topology from Version 1. This prevents broadcast messaging, creates single points of failure, and also lacks route redundancy. The system also lacks session management as there is no concept of an ongoing authenticated conversation, no ability to amortize

cryptographic costs across multiple messages, and no mechanism to track conversation state.

Improvements in Version 3:

Version 3 tackles performance issues regarding real-time applications of Version 2 with a session-based architecture while maintaining the cryptographic security properties established in the previous version.

Session-based Architecture:

The current version of the system uses session lifecycle states to separate one-time cryptographic setup from ongoing real-time communication. Each of the session transition through the states with clear security properties in each stage. In this architecture, zk-SNARK proof generation and verification occur during the start of a session. Instead of proving each packet's authenticity independently, the system performs heavy zk-SNARK operations all at once during session setup and then uses lightweight cryptographic operations (HMAC) for subsequent messages.

Session Lifecycle: Sessions progress through well-defined states with clear security properties. These states are:

- **INIT:** Session object created, awaiting handshake initiation
- **HANDSHAKING**: Multi-step handshake in progress, exchanging keys and challenges
- **ESTABLISHED:** Session active, secure channel ready for message transmission
- **EXPIRED:** Session timeout reached, no longer valid for new messages
- **CLOSED:** Session explicitly terminated, keys discarded

Architectural Benefits:

Performance Transformation: One 126-second zk-SNARK proof supports thousands of messages. A 10-minute voice call now requires 120 seconds for session establishment plus $\sim$0.6 seconds for 6 verification nodes(sender, repeater[4], receiver). Which comes to the authentication time of (6,000 packets $\times$ 0.1ms each), reducing total cryptographic overhead from 14,500 hours to 121 seconds.

Cryptographic Efficiency: Heavy operations occur once per session, and these heavy tasks include zk-SNARK proof generation, smart contract verification, and digital signature creation. But lightweight operations such as HMAC computation, sequence number checking, and timestamp validation take place per message with $\sim$0.1ms overhead.

Security Preservation: All cryptographic guarantees from Version 2 are maintained in this version, as well as the session key is obtained from the zk-SNARK proof, and HMAC tags are cryptographically added to the original proof.

Proof Hash Binding Mechanism:

In Version 3, we have implemented the proof hash binding mechanism. This method cryptographically links every message to the original zk-proof to ensure session-based security. Whenever a session is created through zk-SNARK proof, the system computes proof_hash = SHA-256(full_zk_proof). This 32-byte hash represents the entire ∼10 KB proof. This full proof is verified once, and then this proof_hash is cached at all network nodes. HMAC is computed over every message following the structure of HMAC (session_key, proof_hash || session_id || sequence || timestamp || data). The proof_hash is included in the HMAC computation, which creates a cryptographic binding. As a result, an attacker cannot forge a valid HMAC without the session key and the correct proof_hash. Then this is again verified at the repeater nodes; this dual verification ensures that the message can't be injected, tampered, or replayed, even though full zk-proof verification doesn't occur for each message. As a result, the mechanism maintains zk-SNARK security properties while achieving HMAC-level performance.

Security Properties Established: The proof-hash-binding mechanism provides multiple layers of security in the network system. Through proof binding, every message is cryptographically linked to the original zk-SNARK proof. As a result, an attacker cannot forge valid HMACs without both the session key derived from the proof and the correct proof_hash representing the verified proof.

The proof_hash in each packet must match the cached proof_hash from session establishment; this ensures session integrity. If an attacker attempts to inject messages into a session, they have to possess the original session_key, which is impossible without breaking zk-SNARK security, or forge a valid HMAC, which can not be done without the key. And even if an attacker intercepts packets, they cannot modify message contents, as this would invalidate the HMAC. They also can not reorder messages or inject new messages due to the lack of a session key and the correct proof_hash. In addition to this, MITM attacks can not replace the proof_hash because it would fail comparison with the cached value. To ensure replay prevention, the mechanism strictly increases sequence numbers to prevent message reuse. The timestamps provide additional freshness validation while each of the nodes tracks last_seen_sequence per session, and rejects any packet with sequence $\leq$ last_seen_sequence.

The Cryptographic Rationale: The proof_hash binding improves the basic properties of cryptographic hash functions by ensuring collision resistance. Considering proof_hash = SHA-256(full_zk_proof), it is computationally impossible to find any different proof P' such that SHA-256(P') = proof_hash. Therefore, the proof_hash uniquely identifies the original zk-SNARK proof with overwhelming probability. It includes proof_hash in the HMAC computation cryptographically, while binding each message to that specific proof, and the verification confirms the message authenticity via HMAC and its binding to the verified proof via proof_hash comparison. This creates a cryptographic chain: zk-SNARK proof $\rightarrow$ proof_hash $\rightarrow$ HMAC $\rightarrow$ message, and breaking the chain requires tackling either SHA-256 collision resistance or HMAC security. But both of these are computationally infeasible with the current cryptographic knowledge.

Performance vs. Security Trade-off: The proof_hash binding mechanism combines the security of zk-SNARKs with the speed of HMAC verification. It ensures that,

- A full zk-SNARK verification takes roughly about 30 seconds per message.
- Verifying an HMAC with a bound proof_hash takes only ∼0.1 milliseconds per message.
- The security reduction is negligible, since the cryptographic features remain effective as before.

This optimization delivers a massive performance boost, making real-time applications feasible without sacrificing any of the security properties established in Version 2.

Cryptographic Primitive Optimization:

The current version of the system uses BLAKE2b instead of SHA-256 to control HMAC operations. The BLAKE2b provides equivalent security while achieving significantly higher throughput on modern CPUs. This happens because it is optimized for 64-bit platforms and takes fewer rounds than SHA-256. BLAKE2b has a built-in mode to eliminate HMAC overhead. Moreover, session establishment (Algorithm 4 applies Ed25519 for digital signatures [11], and the session keys are extracted from HKDF (HMAC-based Key Derivation Function). This particular step creates cryptographic separation between different key purposes and enables forward secrecy.

Modified System Components:

Dual Packet Type Architecture: The SessionPacket structure is used exclusively during session establishment, and the DualPacket Structure is used for real-time message transmission.

Structure: SessionPacket	
`packet_type`	▷ *HELLO | CHALLENGE | PROOF | CONFIRM*
`handshake_data`	▷ *Bytes: Contains ZK proof (∼10 KB)*
`header_fields`	▷ *Standard protocol header metadata*

Structure: DataPacket	
`data`	▷ *Bytes: Message payload*
`hmac_tag`	▷ *Bytes(32): HMAC authentication tag*
`proof_hash`	▷ *Bytes(32): Binds to ZK proof*
`sequence`	▷ *Integer: Replay protection counter*
`header_fields`	▷ *Standard protocol header metadata*

System Architecture:

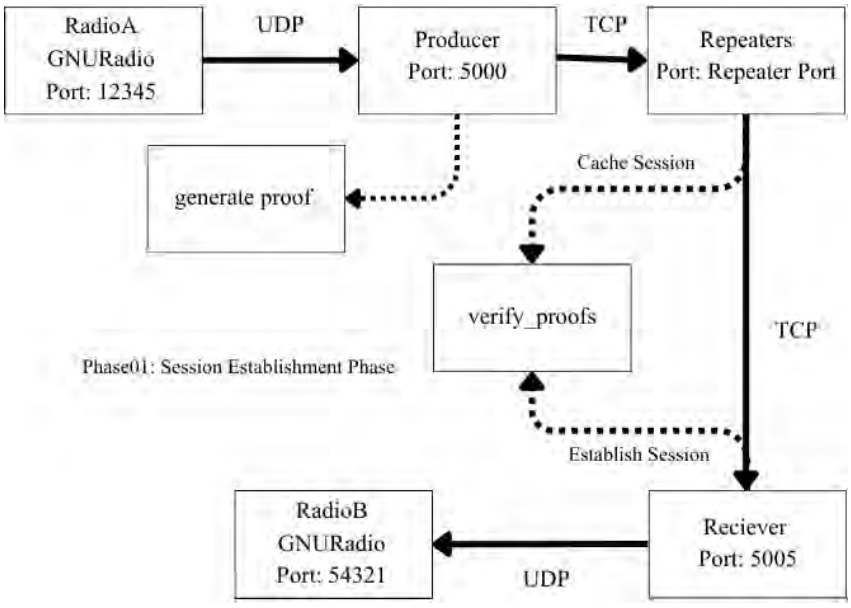

Figure 4.10: Version 03;phase-01

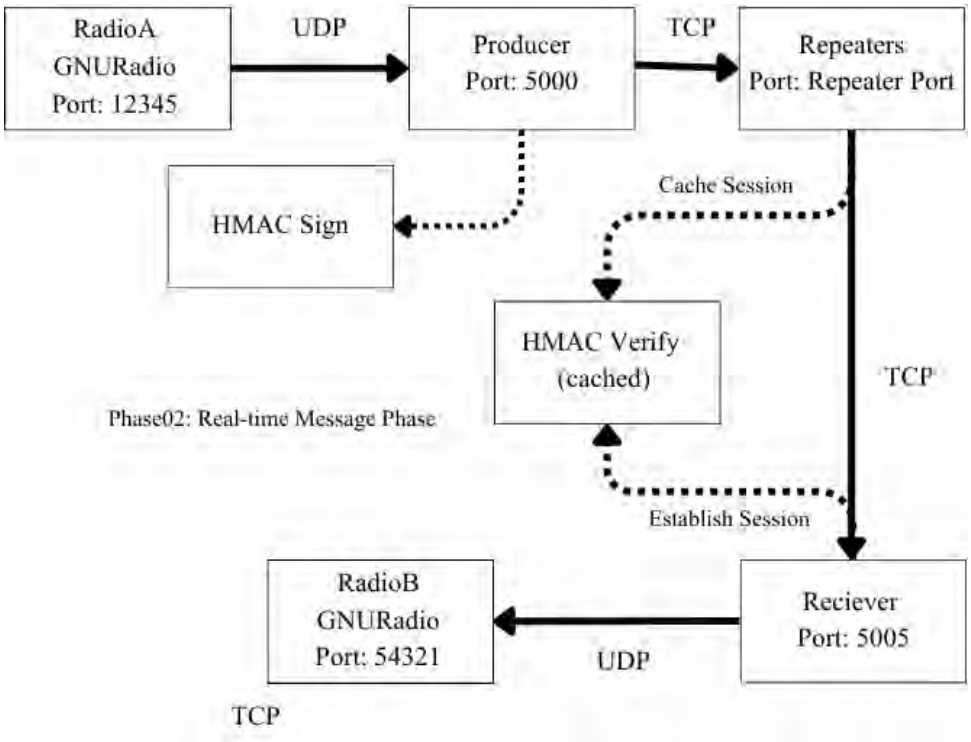

Figure 4.11: Version 03;phase-02

Algorithm 4 Version 3: Session-Based Authentication and Key Derivation

procedure SessionManager(node_id) ▷ Session Manager [NEW]
  Function `create_session`($peer\_id$):
  if $peer\_id \in sessions\_by\_peer \land \neg sessions[peer\_id].is\_expired$ **then return** $sessions[peer\_id]$ ▷ Check for existing active session
  end if
  $S \leftarrow$ new Session(id : gen(), $node$: $node\_id$, $peer$: $peer\_id$, timeout)
  return $S.establish$(DeriveKey($p.\pi, pk_{node}, nonce$))
end procedure
procedure SessionStateManagement(session S) ▷ Session State Management [NEW]
  Function `establish`(K_{sess}, π, com):
  $S.K_{hmac} \leftarrow$ HKDF_Expand(K_{sess}, "HMAC", 32); $S.\pi \leftarrow \pi$
  $S.proof\_hash \leftarrow$ SHA256(encode_UTF8(π)) ▷ CRITICAL: Binding to proof
  $S.auth \leftarrow$ new HMACAuth($S.K_{hmac}, S.id, S.proof\_hash$)
  Function `create_authenticated_message`(M):
  $(tag, seq, ts) \leftarrow S.auth.create\_tag(M)$; **return** (tag, seq, ts)
  Function `verify_authenticated_message`(M, tag, seq, ts):
  return $S.auth.verify\_tag(M, tag, seq, ts,$ max_age $= 30.0,$ min_seq $= S.recv\_seq)$
end procedure
procedure HKDFImplementation($ikm, salt, info$) ▷ HKDF Implementation [NEW]
  Function `hkdf_extract`($salt, ikm$): **return** HMAC($salt, ikm$)
  Function `hkdf_expand`($prk, info, L$):
  $n \leftarrow \lceil L/hash\_len \rceil$; $T_i \leftarrow$ HMAC($prk, T_{i-1} + info + i$); **return** $T_1..T_n[:L]$
  Function `derive_session_key`(com, $pk, nonce$):
  $ikm \leftarrow$ com $+ pk + nonce$; **return** HKDF(ikm, salt $=$ sha(pk), info)
end procedure

Modified End-to-End System Operation:

Version 3 restructures the end-to-end protocol (Algorithm 5) around session establishment and fast message authentication.

Structure: Session Object $\mathcal{S}$

`session_id`	▷ *Unique identifier for the node pair*
`hmac_key`	▷ *32-byte key derived via HKDF*
`proof_hash`	▷ *SHA256 binding to the ZK-SNARK proof*
`last_seq`	▷ *Integer: Highest verified sequence number*
`is_active`	▷ *Boolean: Session validity/expiration state*
`established_at`	▷ *Timestamp: Creation time for age validation*

Algorithm 5 Version 3: Session-Based Packet Processing Protocol

procedure PHASE 1: SESSION ESTABLISHMENT(peer_id) ▷ One-time, ∼120s
  if $active\_sessions[peer\_id].is\_active$ **then return** $session$
  end if
  $com \leftarrow \text{Commit}(P.id, P.pk, \text{now}(), nonce)$
  try
    $\pi \leftarrow \text{Subprocess}(\text{"gen\_proof.js"}, com, \text{timeout} = 120s)$
  catch
    $\pi \leftarrow \text{GenSimulatedProof}()$ ▷ Testing fallback
  end try
  $K_{sess} \leftarrow \text{DeriveKey}(com, P.pk, nonce)$
  $session.establish(K_{sess}, \pi, com)$ ▷ Binds $proof\_hash \leftarrow \text{SHA256}(\pi)$
  return $session$
end procedure
procedure PHASE 2: FAST MESSAGE AUTH(session S, M) ▷ Per message, ∼0.1ms
  if $\neg S.is\_active$ **then**
    throw SessionExpiredError
  end if
  $(tag, seq, ts) \leftarrow S.create\_auth\_tag(M)$ ▷ HMAC binds $proof\_hash$
  return new DataPacket(type: DATA, data: M, hmac_tag: tag, π_hash: $S.\pi$_hash)
end procedure
procedure PHASE 3: FAST MULTI-HOP VERIFICATION(pkt, R_i) ▷ Per hop, ∼0.1ms
  $S_{cache} \leftarrow R_i.cache.get(pkt.sid)$
  if $S_{cache} = \text{NULL}$ **then return** SESSION_NOT_FOUND
  end if
  if $pkt.\pi_{hash} \neq S_{cache}.\pi_{hash} \vee pkt.seq \leq S_{cache}.last\_seq$ **then**
    return BINDING_OR_REPLAY_FAIL ▷ Critical Security Check
  end if
  $auth\_data \leftarrow \text{build}(pkt.\pi_{hash}, pkt.sid, pkt.seq, pkt.ts, pkt.data)$
  if $\text{HMAC}(S_{cache}.K_{hmac}, auth\_data) \neq pkt.hmac\_tag$ **then return** AUTH_FAIL
  end if
  $S_{cache}.last\_seq \leftarrow pkt.seq$; FORWARD(pkt)
end procedure
procedure PHASE 4: FINAL VERIFICATION AND DELIVERY(pkt) ▷ Receiver D
  $(valid, reason) \leftarrow \text{VERIFYPHASE3}(\text{pkt})$
  if $valid$ **then** SENDUDP($pkt.data, Radio\_B$); **return** SUCCESS
  end if
  return DELIVERY_FAIL
end procedure

Implementation Considerations

Session Cache Management The session cache is kept simple on purpose. By default, it can hold up to about 10,000 sessions. When the cache fills up, it starts dropping older sessions that haven't been used recently. This will avoid the out-of-control memory and yet be beneficial for the cache. The actual size of the right cache actually varies with the

number of sessions being served at any one time, the duration of time that the sessions last, and the amount of memory the respective node is capable of effectively managing. The cleanup of expired sessions is performed in the background regularly and in a manner that does not impede normal traffic. A small cleanup task runs roughly once every minute and clears out anything that's no longer valid. Cache performance metrics (hit rate, miss rate, eviction rate) guide tuning is used to indicate appropriate cache sizing.

Session Timeout Configuration

Depending on the deployment, different timeout settings are needed for the system to work efficiently. Longer sessions reduce cryptographic overhead but increase risk if session keys are compromised. Therefore, idle timeouts are tracked using a last_activity timestamp that updates whenever a message is sent or received. This helps free up cache space by clearing out abandoned sessions and also reduces the risk from session keys that might have been compromised without being detected.

Proof Hash Binding Security

An attacker can not inject messages into a session unless they have both the session key and the proof hash, therefore keeping the sessions safe. The replay attacks in the system are prevented by using strictly increasing sequence numbers and fresh timestamps. Any packet that has a sequence number equal to or lower than the last seen, a timestamp older than 30 seconds, or a timestamp too far in the future (over 5 seconds) is automatically rejected. Along with this, a proof hash is added to the HMAC to protect against man-in-the-middle attacks. Even if someone intercepts a packet and tries to tamper with it, they won't be able to forge a valid HMAC without the original proof hash. This particular feature ensures that every message stays authentic and unchanged.

Known Limitations and Future Work

Version 3 makes real-time message authentication possible while keeping strong cryptographic security, but there are still a few challenges to work on. Maintaining synchronization of all the network nodes is a challenge since when the information regarding sessions is lost between repeaters, the correct message may be discarded. The original 120-second session arrangement is also unnecessarily lengthy in instances where messages are required to pass instantly. At the moment, we can not revoke a session immediately once a key is compromised, so further versions can incorporate a method to revoke a session on demand or generate revocation lists in order to revoke access quickly.

Even the network architecture has its limits since Version 3 is still based on the linear chain, and thus, it cannot send a message to a number of nodes simultaneously. Since it has one point of failure, the network is not able to utilize the redundancy and flexibility that would be offered by a mesh setup. Although it will be simpler to develop when it comes to improving the sessions, a full mesh network will be required in future versions. In addition, the Producer node has the responsibility for all the session creation, which can be a possible bottleneck and single point of failure, since they expect that the Producer node can be completely trusted. Although cryptography-inspired ZK-SNARKs are employed, the network is not fully decentralized on a blockchain yet; thus, there is no consensus among the nodes as part of the system. Lastly, there is the issue of key management, which would be a security issue. Session keys reside in the memory of all

the nodes during the course of a session, and this might be exposed to attacks when an individual accesses them. No automatic key rotation is also available, so long-lived sessions may be potentially compromised in case of key compromise. It will be significant to fix these problems in the subsequent versions to ensure that the network will be more reliable, decentralized, and secure,e and retain the same real-time performance that Version 3 provides.

Key Achievements:

The system is a performance breakthrough because the identity verification is isolated, and identity authentication is performed only once with the initial (one-time) setup of the system. As an example, a 10-minute voice call (approximately 6,000 packets) requires only 120 seconds to initiate the first session and 600 milliseconds to perform total authentication, in comparison with 210 hours in Version 2. The proof hash binding mechanism that includes the proof hash with every message maintains the message associated with the original ZK proof and sends the 32-byte proof per packet rather than the 10 KB packet. This is very secure, and it does not cause the performance to slow down. Finally, Repeater nodes can authenticate sessions using an O(1) look up, so it is a fast and scalable system regardless of the number of sessions or the number of nodes on the network. This is to guarantee session cache optimization.

Conclusion

Version 3 introduces real-time secure communication to the network through a session-based authentication system. The system uses the spread of the 120-second zero-knowledge proof setup throughout the entire session to cut the per-message cryptographic cost of Version 2 (3.2.6) of 126 seconds down to a reasonable 100 milliseconds, and the real-time use of the system is now feasible. The version is a combination of cryptographic security and real-time performance, but it still has a high vulnerability of centralization: the producer node is the one-sidedly controlling all session creation. It is only one source of trust, which poses operational risks and goes against blockchain principles of decentralization. The second iteration of the system will remove this weakness by deploying the distributed blockchain consensus and changing the way the session is established. And it provides the real-time performance attained in Version 3 by architectural separation of control and data planes.

4.3 Implementation of Selected Design

4.3.1 Blockchain-Enabled Radio Network Version 04: Decentralized Blockchain Consensus

Limitations of Version 3

While Version 3 achieved real-time performance through session-based authentication, it still had key architectural drawbacks that prevented it from being truly decentralized and resilient.

Centralization Bottleneck: The Producer node served as the central hub for all session setups, which created multiple vulnerabilities. Since every session had to pass

through this single node, it could easily become a performance bottleneck and a single point of failure. The system relied on trusting the Producer completely, which weakened its overall reliability. But it has no features to verify its decisions, and if the Producer becomes compromised or fails, the entire network loses the ability to establish new sessions.

Lack of Consensus: Version 3 acts without any distributed consensus over the network state. The Producer alone decides the validation and initial state without any verification from other network participants. Repeater nodes just accept this session information without any verification of the session's legitimacy. But it is not compatible with decentralized blockchain principles, as no single entity should have unilateral authority over the whole network state.

State Synchronization Challenges: Maintaining consistent session state across multiple nodes is challenging with this version because if network partitions occur, different nodes may develop inconsistent views of valid sessions. When nodes restart or join the network, they lack a reliable mechanism to synchronize their session cache with the current network state. Therefore, the absence of an authoritative, tamper-evident record of session establishment makes dispute resolution impossible.

Limited Fault Tolerance: The linear chain topology inherited from previous versions provides no redundancy. For this reason, a single repeater failure completely removes the forwarding path of the network system. Moreover, this version has no features that will allow the network to collectively agree on a valid state despite individual node failures.

Operational Inflexibility: Session management remains static and rigid in Version 3. After initialization, the sessions cannot be modified, transferred, or revoked between nodes without any manual intervention. And even during changing conditions, the network cannot adapt dynamically to respond to security incidents through collective decision-making.

4.3.2 Blockchain Consensus Architecture

Version 4 of this network system introduces a decentralized blockchain consensus mechanism. This mechanism uses a lightweight Byzantine Fault Tolerant (BFT) consensus designed for radio mesh networks. In this consensus, nodes are semi-trusted, and connectivity can be imperfect. Instead of giving a single Producer full control, the network requires a shared agreement before any session is established.

Architectural Separation (Control Plane and Data Plane) : The current version of the network system operates on two different communication planes, each optimized for its specific purpose. The Control Plane (Port 7000) operates different blockchain operations. These operations include block proposals, signature collection, and finalized block distribution. It operates asynchronously and can tolerate higher latency (∼100-500ms) as the consensus occurs infrequently and only during session initialization. And the communication method opts for a broadcast topology so that the Producer can reach all repeaters directly. The Data Plane (Ports 5000+) handles authenticated message

traffic by using session-based HMAC verification from Version 3. This plane requires minimal latency (∼0.1-0.15ms per hop) and operates continuously for real-time communication. Communication uses linear chain topology for performance testing, even though the architecture supports mesh configurations.

This separation of architecture ensures that the consensus overhead does not impact real-time message verification performance.

Blockchain Structure: The blockchain maintains an immutable, ordered ledger of session establishment. Each of the block contains Block Metadata consisting of the sequential block number, timestamp, and hash of the previous block (chain linkage). It also keeps records of the session with complete cryptographic binding of radio nodes to their ZK-SNARK proofs. For consensus proof, digital signatures from a majority of repeater nodes are collected, demonstrating network agreement. Finally, the system creates a hash that connects everything in the block, including signatures. Once a session is added to a finalized block, it becomes part of the network for good, and anyone can verify it whenever they want.

4.3.3 Key Structures:

Data Model Definitions:

To establish a tamper-evident ledger for the radio network, the system defines a set of hierarchical data structures (Table 4.2). These structures make sure that session management is cryptographically bound to the blockchain's consensus layer. The following core components form the 7000 data model of the proposed architecture.

Table 4.2: Definition of Core Data Structures

Structure: Blockchain Core Components	
SessionRecord $\mathcal{SR}$:	
`session_id, radio_id`	▷ *Unique identifiers (32 chars)*
`public_key, proof_hash`	▷ *Ed25519 key and SHA256 binding (64 hex)*
`commitment, ts_created`	▷ *ZK commitment and creation timestamp*
BlockProposal $\mathcal{BP}$:	
`number, timestamp`	▷ *Sequential height and Unix time*
`previous_hash`	▷ *SHA256 link to preceding block*
`sessions[]`	▷ *List of active SessionRecords to be finalized*
`proposal_hash`	▷ *Deterministic identifier for consensus*
Block $\mathcal{B}$:	
`signatures[]`	▷ *The consensus proof (List of RepeaterSignatures)*
`block_hash`	▷ *Final immutable SHA256 identifier*

Consensus Mechanism:

The integrity of the blockchain is maintained through a series of deterministic procedures (Algorithm 6) that maintain how blocks are proposed, signed, and finalized. The core of

this process is the generation of unique, immutable identifiers. These specific proposals and block hashes link the current state to its former versions. These algorithms allow repeaters to independently verify the content of a block before providing it with their cryptographic endorsement.

Algorithm 6 Consensus and Block Finalization Procedures

procedure ComputeProposalHash($prop$)
    $S \leftarrow \text{concat}(prop.num, prop.prev\_hash, prop.id, \text{sort}(prop.sessions), prop.ts)$ ▷ Deterministic Serialization
    return SHA256(S) ▷ Allows independent repeater verification
end procedure
procedure SignProposal($node, prop$)
    $msg \leftarrow prop.hash + \text{" : "} + node.id + \text{" : "} + \text{now}()$ ▷ Repeater endorsement
    $\sigma \leftarrow \text{Ed25519Sign}(node.priv\_key, msg)$
    return new RepeaterSignature($\sigma, node.id, prop.hash, node.pub\_key$)
end procedure
procedure ComputeBlockHash($block$)
    $json \leftarrow \text{JSON\_Encode}(block, \text{sort\_keys} = \text{true})$ ▷ Final immutable chain identifier
    return SHA256($json$) ▷ Hash includes all signatures
end procedure

4.3.4 Storage Layer Architecture:

To identify and evaluate the high-latency and resource-constrained nature of radio communication, the system implements a performance-optimized storage layer. By using a disk-hybrid architecture, the design fulfills the need for long-term data durability with the requirement for immediate verification of active sessions. This multi-tier approach ensures that the data remains persistent while frequently accessed cryptographic bindings are prioritized for high-speed retrieval.

Blockchain Storage: Disk-Hybrid Architecture

This version of the system implements a performance-optimized storage layer designed to balance the durability requirements of a distributed ledger with the low-latency needs of a radio network:

- **Persistent Ledger (Disk-Backed):** Here, every finalized block is saved to a permanent file on the disk that only grows forward. It's our safety net—if a node loses power, the data stays safe and won't be lost.
- **Session Cache (Memory-Resident):** To keep verification moving at a steady clip, we store active session records in a high-speed memory map. This lets repeaters find and verify data instantly during multi-hop transfers, so they don't get bogged down waiting for slow disk reads.
- **Binding Cache (Memory-Resident):** This is a specialized index that maps a session_id to a proof_hash. It lets the system perform instant cryptographic checks without needing to pull the entire record.

4.3.5 High-Performance Session Validation:

To avoid slowing down the radio data plane, we implemented a dual-cache system (Algorithm 7) for $O(1)$ session validation. This splits the data into two tiers: a lightweight Proof-Hash Mapping (L0) for instant identity checks, and the full Session Record State (L1) for the deeper metadata. By splitting these layers, the network can instantly verify identities and expiration times without the lag of pulling up a full session file.

Algorithm 7 O(1) Fast Session Validation

```
procedure IsValidSessionFast(session_id, proof_hash)  ▷ Executed per message
                                  ▷ Step 1: Check L0 Cache (O(1) Existence check)
    cached_hash ← L0_cache[session_id]
    if cached_hash = NULL then return FALSE                  ▷ Session unknown
    end if
                                  ▷ Step 2: Verify Proof-Hash (O(1) Binding check)
    if cached_hash ≠ proof_hash then return FALSE      ▷ Possible hijack attempt
    end if
                                  ▷ Step 3: Check L1 Cache (O(1) Expiration check)
    session ← L1_cache[session_id]
    if session.expires_at < now() then return FALSE              ▷ Session expired
    end if
    return TRUE                                  ▷ Proceed with HMAC verification
end procedure
```

Control Plane: Blockchain-Based State Synchronization and Consensus

The consensus process follows an eight-phase protocol. Through this protocol, no single node gets to decide what happens, and the network moves forward only when enough independent participants agree.

1. **Proposal creation:** The Producer creates a proposed block with pending sessions (Algorithm 8).
2. **Broadcast to the network:** That proposal is sent out to all repeaters at the same time (Algorithm 8).
3. **Independent verification:** Each repeater checks the proposal on its own to make sure that it's valid and authentic (Algorithm 10).
4. **Digital signing:** If the proposal is valid, the repeater signs it using Ed25519 [11] (Algorithm 10).
5. **Majority approval:** Producer collects signatures until most of the repeaters have approved the proposal (Algorithm 11).
6. **Block finalization:** The Producer finalizes the block and attaches the collected signatures with it after the threshold value is met (Algorithm 11).
7. **Block distribution:** The finalized block is broadcast back out to the entire network (Algorithm **??**).

8. **Session caching:** All of the nodes update their local caches with new sessions (Algorithm **??**).

When the Producer accumulates pending sessions (from completed zk-SNARK handshakes), it creates a block proposal. And the proposal hash is computed deterministically, ensuring all nodes that verify the proposal will compute identical hashes, making it a prerequisite for cryptographic signing. The Producer broadcasts the proposal to all repeaters simultaneously using parallel threads to minimize latency. Each repeater connection runs in a separate thread to maximize parallelism. This thread-safe collection (Algorithm 9) prevents race conditions, and the mutex ensures that concurrent signature additions cannot corrupt the collected_signatures list or produce incorrect counts. When a repeater receives a block proposal, it performs 3 independent validations before signing. These are- Chain Continuity, Sequential Numbering, and Hash Integrity. This three-step verification ensures that the proposal is correctly extending the blockchain, blocks are being added sequentially without any gaps, and the proposal data has not been tampered with during transmission. The digital signature algorithm provides strong cryptographic security with proper performance. Once the Producer collects sufficient signatures [11], it finalizes the block, and this finalized block becomes part of the immutable blockchain. Along with its hash, this acts as the previous_hash for subsequent blocks. The Producer distributes the finalized block to all network nodes. And lastly, when nodes receive a finalized block, they update their local caches. After Phase 8 completes, all network nodes possess identical blockchain state and can independently verify sessions during message processing.

Algorithm 9 Thread-Safe Signature Collection

```
1: function COLLECT_SIGNATURE(repeater, proposal)
2:     try
3:         socket ← TCP_CONNECT(repeater.host, repeater.port, timeout = 5s)
4:         TCP_SEND(socket, pack(MSG_PROPOSAL, proposal))
5:         (type, data) ← unpack(socket, timeout = 5s)
6:         if type ≠ MSG_SIGNATURE then return FALSE
7:         end if
8:         sig ← deserialize(data)
9:         ACQUIRE(signature_lock)                    ▷ Enter Critical Section
10:        try
11:            if not valid(sig, proposal) then return FALSE
12:            end if
13:            if sig.id ∉ map(λs : s.id, collected_sigs) then
14:                collected_sigs.append(sig)
15:                log("Collected: " + |collected_sigs|)
16:            end if
17:            return TRUE
18:        end try
19:        RELEASE(signature_lock)
20:    catchNetworkError return FALSE
21:    end try
22:    TCP_CLOSE(socket)
23: end function
```

Algorithm 8 Phases 1 & 2:Consensus: Block Proposal and Signature Collection

```
function EXECUTE_CONSENSUS_ROUND
                                                    ▷ Phase 1: Create Proposal
    if pending_sessions.length = 0 then return NULL
    end if
    latest ← blockchain_storage.get_latest_block()
    prop ← NEW BlockProposal(latest.num+1, now(), latest.hash, COPY(sessions))
    prop.proposal_hash ← compute_proposal_hash(prop)
                                          ▷ Phase 2: Broadcast and Signature Collection
    sigs ← ∅; lock ← new Mutex(); threads ← ∅
    for all repeater ∈ repeater_network do
        t ← START_THREAD(collect_sig, repeater, prop)
        threads.append(t)
    end for
    for all t ∈ threads do
        THREAD_JOIN(t, 5.0s)
    end for
                                                  ▷ Step 3: Majority Quorum Verification
    required ← ⌈|repeater_network|/2⌉ + 1
    if |sigs| ≥ required then
        return {prop, sigs}
    else
        log_error("Consensus failed"); return NULL
    end if
end function
```

Algorithm 10 Phases 3 & 4: Repeater Verification and Signing

```
1: function HANDLE_PROPOSAL(proposal)
2:     t_0 ← get_time(), L ← storage.get_latest_block()
3:                                   ▷ Step 1 & 2: Continuity and Sequence (O(1))
4:     if proposal.prev_hash ≠ L.hash or proposal.num ≠ L.num + 1 then
5:         log_error("Chain continuity or sequence break")
6:         return NULL
7:     end if
8:                                                ▷ Step 3: Hash Integrity (O(n))
9:     computed_hash ← compute_proposal_hash(proposal)
10:    if proposal.hash ≠ computed_hash then
11:        log_error("Hash mismatch - tampering detected")
12:        return NULL
13:    end if
14:                                                ▷ Phase 4: Digital Signing
15:    v_time ← get_time() − t_0
16:    log_info("Verified in " + v_time + " μs")
17:    sig ← sign_proposal(repeater_node, proposal)
18:    log_info("Signed block #" + proposal.num)
19:    return sig
20: end function
```

Algorithm 11 Phases 5 & 6: Majority Check and Block Finalization

```
1: function FINALIZE_BLOCK
2:     ACQUIRE(signature_lock)
3:     try
4:                                                    ▷ Step 1: Validate State
5:         if current_proposal = NULL then
6:             LOG_ERROR("No active proposal")
7:             return NULL
8:         end if
9:                                          ▷ Step 2: Majority Quorum Verification
10:        req ← ⌈|repeater_network|/2⌉ + 1
11:        act ← |collected_signatures|
12:        if act < req then
13:            return NULL
14:        end if
15:                                      ▷ Step 3 & 4: Block Creation and Hashing
16:        block ← NEW Block(current_proposal, DEEP_COPY(collected_signatures))
17:        block.hash ← COMPUTE_BLOCK_HASH(block)
18:                                    ▷ Step 5 & 6: Atomic Storage and State Reset
19:        if not blockchain_storage.ADD_BLOCK_ATOMIC(block) then
20:            LOG_ERROR("Storage failure")
21:            return NULL
22:        end if
23:        pending_sessions.CLEAR
24:        current_proposal ← NULL
25:        collected_signatures.CLEAR
26:        return block
27:    end try
28:    RELEASE(signature_lock)
29: end function
```

Security Assumptions for Control Plane

The system's security relies on the following assumptions. Firstly, it assumes that more than half of consensus participants behave honestly and that SHA-256, BLAKE2b, and Ed25519 [11] provide computational security. The system also considers that the o1js/Mina Protocol implementation correctly implements zero-knowledge proofs. Then it also assumes that the nodes generate and store private keys securely, and the initial node registration uses out-of-band authentication. Lastly, it considers that the messages are eventually delivered even with possible delays during network partitions.

Security Properties for Control Plane

This consensus protocol establishes multiple security guarantees. To begin with, the protocol creates a decentralized trust where no single node can unilaterally add sessions and requires majority agreement. The system remains secure even if a minority of nodes are

Algorithm 12 Consensus Phase 7 & 8: Finalization and Caching

function FINALIZE_AND_CACHE(block)
  $L \leftarrow Storage.GetLatest()$ ▷ Step 1: Chain & Signature Validation
  if $block.prev \neq L.hash \vee block.num \neq L.num + 1 \vee \neg Verify(block)$ **then**
    log_error("Invalid Block"); **return false**
  end if
  $threads \leftarrow \emptyset$; $target\_nodes \leftarrow \{repeater\_network \cup receiver\}$
  for all $n \in target\_nodes$ **do** ▷ Step 2: Parallel TCP Broadcast
    $t \leftarrow$ START_THREAD(TCP_Broadcast, n, Pack($MSG\_FIN, block$))
    $threads.append(t)$
  end for
  for all $t \in threads$ **do**
    JOIN($t, 5.0s$)
  end for ▷ Step 3: Network Sync
  $Storage.AddBlock(block)$ ▷ Step 4: Persistence & Caching
  for all $s \in block.sessions$ **do**
    $k \leftarrow$ DeriveKey($s.commitment, s.pub\_key, s.id$)
    $session\_cache[s.id] \leftarrow s$; $hmac\_cache[s.id] \leftarrow k$
    $proof\_cache[s.id] \leftarrow$ HexDecode($s.proof\_hash$)
  end for
  log_info("Broadcast and Caching complete"); **return true**
end function

compromised or faulty due to Byzantine Fault Tolerance. Any modification to historical blocks is immediately detectable via the hash chain as tamper evidence is collected throughout the session. The digital signatures provide authenticity through the cryptographic proof of each node's participation. And lastly, the complete session history is permanently recorded so that it can be verified.

4.3.6 Network Protocol: Binary Wire Format

Version 4 implements a custom binary protocol that is optimized for efficient blockchain communication over TCP. This format minimizes overhead, maintains clear message boundaries, and addresses the partial read issues that were present in the earlier versions.

Message Structure:

Byte Offset	Field	Size	Type	Description
0	`msg_type`	1 byte	unsigned int	Message type identifier
1-4	`data_length`	4 byte	unsigned int (BE)	Payload length
5-N	`payload`	N byte	varies	Message-specific data

Total header size: 5 bytes
Maximum message size: 4,294,967,295 bytes (4GB, though practical limit is 10MB)

Message Types:

Protocol Message Types	
`MSG_PROPOSAL (1)`	▷ *Producer → Repeater: Block proposal for signing*
`MSG_SIGNATURE (2)`	▷ *Repeater → Producer: Signature endorsement*
`MSG_FINALIZED (3)`	*→ Producer → All nodes: Finalized block distribution*
`MSG_ACK (4)`	▷ *Generic: Protocol synchronization acknowledgment*

Message Packing:

Algorithm 13 Protocol Message Serialization

```
1: function PACK_MESSAGE(msg_type, payload_data)
2:     header ← ALLOCATE_BYTES(5)
3:     header[0] ← msg_type                          ▷ Byte 0: Message Type ID
4:                                  ▷ Bytes 1–4: 32-bit Payload Length (Big-Endian)
5:     len ← payload_data.length
6:     header[1] ← (len ≫ 24) & 0xFF                 ▷ Most Significant Byte
7:     header[2] ← (len ≫ 16) & 0xFF
8:     header[3] ← (len ≫ 8) & 0xFF
9:     header[4] ← len & 0xFF                        ▷ Least Significant Byte
10:    return CONCATENATE(header, payload_data)
11: end function
```

Binary Layout Example:

MSG_PROPOSAL (1) with 500-byte Payload		
Byte Index	**Hex Value**	**Description**
`0`	`01`	*msg_type: MSG_PROPOSAL*
`1--4`	`00 00 01 F4`	*payload_length: 500 bytes*
`5--504`	{`...JSON...`}	*Payload: JSON Data*
Total	**505 Bytes**	

Message Unpacking with Error Handling:

TCP streams do not preserve message boundaries—recv(n) may return fewer than n bytes. The unpacking function handles partial reads.

Algorithm 14 Robust Network Message Deserialization

```
1: function UNPACK_MESSAGE(socket, timeout)
2:     SET_TIMEOUT(socket, timeout)
3:     hdr ← ALLOC(5), r ← 0                          ▷ Phase 1: 5-byte Header Read
4:     while r < 5 do
5:         try
6:             ck ← RECV(socket, 5 − r)
7:             if ck.len = 0 then return NULL
8:             end if
9:             COPY(ck, hdr, r) ; r ← r + ck.len
10:        catch
11:            return NULL
12:        end try
13:    end while
14:    type ← hdr[0]                                  ▷ Phase 2: Big-Endian Length Parsing
15:    len ← (hdr[1] ≪ 24) | (hdr[2] ≪ 16) | (hdr[3] ≪ 8) | hdr[4]
16:    if len > 10^7 or len < 0 then return NULL
17:    end if                                         ▷ Phase 3: Validation
18:    if len = 0 then return (type, ∅)
19:    end if
20:    pay ← ALLOC(len), r ← 0                        ▷ Phase 4: Chunked Payload Read
21:    while r < len do
22:        sz ← MIN(8192, len − r)
23:        try
24:            ck ← RECV(socket, sz)
25:            if ck.len = 0 then return NULL
26:            end if
27:            COPY(ck, pay, r) ; r ← r + ck.len
28:        catch
29:            return NULL
30:        end try
31:    end while
32:    return (type, pay)                             ▷ Phase 5: Return Parsed Object
33: end function
```

This robust implementation handles incomplete reads from TCP buffering, connection failures mid-message, timeout conditions, and malformed headers with invalid lengths.

Custom Data Packet Format:

Byte Offset	Field	Size	Type	Description
0-3	`packet_length`	4 Bytes	Unsigned Int 32	Total Length
4-35	`session_id`	32 Bytes	UTF-8 padded	Session ID
36-67	`proof_hash`	32 Bytes	binary	ZK binding
68-71	`sequence`	4 Bytes	Unsigned Int 32 BE	Seq number
72-79	`timestamp`	8 Bytes	Double BE	Unix time
80-111	`hmac_tag`	32 Bytes	binary	BLAKE2b MAC
112-N	`payload`	N-112 Bytes	binary	Voice/data

In the Packet: proof_hash is included in the HMAC computation,
HMAC = BLAKE2b(key, bytes[4:80] + payload)
Total header size: 112 bytes; with HMAC computation: BLAKE2b(key, bytes[4:80] + payload)

Design Rationale: This particular packet format is suitable for high-throughput verification. The fixed header size (112 bytes) enables fast parsing without variable-length field complications, as all fields are at known offsets. For an O(1) cache lookup before expensive cryptographic operations, it uses the session_id first. In this format, the proof_hash binding mechanism ensures that every packet is cryptographically linked to the original zk-SNARK proof, and applying HMAC last, covers all preceding fields, preventing tampering with any header data. Moreover, big-endian encoding in network byte order ensures cross-platform compatibility.

Data Plane: Packet Authentication and Forwarding Mechanisms

The Producer creates authenticated packets in 6 steps (Algorithm 15). For the first step, it does a session lookup through the O(1) hash table and gets the sequence number (O(1)). Then the producer node builds a packet header and computes HMAC. After completing these steps, it writes the HMAC to the packet and writes the total length. The repeaters verify and forward packets in eight consecutive steps (Algorithm 16). At first, the node confirms the packet size and parses fixed-offset fields. Then it does a session cache lookup to verify the proof_hash binding and reconstructs the HMAC input. Then it computes the expected HMAC (BLAKE2b), constant-time HMAC comparison, and proceeds to the next hop. The Receiver performs final validation before delivering to Radio B (Algorithm 17). To ensure packet verification, it parses the packet header and does a session cache lookup to verify proof_hash binding. Then the receiver does HMAC verification and performs a sequence number check for replay protection. After achieving the time-stamp fitness, it delivers the output to the radio via UDP.

Algorithm 15 Producer: Authenticated Packet Creation

procedure CreateAuthPacket($S_{id}, Payload$)
  $S \leftarrow$ ActiveSessions[S_{id}] ▷ **Step 1: Session Retrieval**
  if $S =$ NULL **then throw** *SessionNotFoundError*
  end if
  $Seq \leftarrow$ SeqNums[S_{id}] $++$; $TS \leftarrow$ Time() ▷ **Step 2: Seq/TS**
  $P \leftarrow$ Allocate($112 +$ len($Payload$)) ▷ **Step 3: Header construction**
  Write($P, 4,$ Pad($S_{id}, 32$)); Write($P, 36, S.H_{proof}$)
  WriteUInt32($P, 68, Seq$); WriteDouble($P, 72, TS$)
  WriteZeros($P, 80, 32$) ▷ Reserve 32B for HMAC
  $I_{auth} \leftarrow P[4:80] + Payload$ ▷ **Step 4: Tag Generation**
  $Tag \leftarrow$ Blake2b($S.K_{hmac}, I_{auth}, 32$)
  WriteBytes($P, 80, Tag$) ▷ **Step 5: Tag Injection**
  $L \leftarrow$ len(P) $- 4$; WriteUInt32($P, 0, L$) ▷ **Step 6: Length**
  return P
end procedure

Algorithm 16 Repeater: Fast Packet Verification and Forwarding

procedure VerifyAndForward(P_{bytes})
  $T_0 \leftarrow$ GetTime()
  if len(P_{bytes}) < 112 **then return false**
  end if ▷ **Step 1: Size**
  $S_{id}, H_{pr}, Seq, TS, Tag_r \leftarrow$ Parse(P_{bytes}) ▷ **Step 2: Header**
  $S \leftarrow$ Cache[S_{id}] $\vee$ Chain.Get(S_{id}) ▷ **Step 3: Blockchain**
  if $S =$ NULL **then return false**
  end if
  if $\neg$Equal($H_{pr}, S.H_{pr}$) **then return false**
  end if ▷ **Step 4: Proof**
  $I_{auth} \leftarrow$ Concat($P_{bytes}[4:80]$, Payload) ▷ **Step 5: Input**
  $Tag_e \leftarrow$ Blake2b($S.K_{hmac}, I_{auth}, 32$) ▷ **Step 6: Keyed Tag**
  if $\neg$ConstTimeComp(Tag_r, Tag_e) **then return false**
  end if ▷ **Step 7: Comp**
  try ▷ **Step 8: TCP Forward**
    $Sock \leftarrow$ TCP.Connect($NextHop$); Send($Sock, P_{bytes}$)
    Close(Sock); **return true**
  catch *NetErr* **return false**
  end try
end procedure

Algorithm 17 Receiver: Comprehensive Verification and Delivery

procedure VERIFYANDDELIVER(P_{bytes})
  stats.rcvd $++$; $T_0 \leftarrow$ GetTime()
  if len(P_{bytes}) < 112 **then return false**
  end if ▷ **Step 1: Validate**
  $S \leftarrow$ GetSession($P_{bytes}.S_{id}$) ▷ **Step 2: Sync**
  if $S =$ NULL **then return false**
  end if
  if $\neg$Equal($P_{bytes}.H_{pr}, S.H_{pr}$) **then return false**
  end if ▷ **Step 3: Proof**
  $Tag_e \leftarrow$ Blake2b($S.K_{hmac}, P_{bytes}[4:80] + Pay, 32$) ▷ **Step 4: HMAC**
  if $\neg$ConstTimeComp($P_{bytes}.Tag_r, Tag_e$) **then return false**
  end if
  if $S.last$ exists $\wedge$ $P_{bytes}.Seq \leq S.last$ **then return false**
  end if ▷ **Step 5: Replay**
  $S.last \leftarrow P.Seq$; $Age \leftarrow$ Now() $- P.TS$ ▷ **Step 6: Freshness**
  if $Age > 30.0 \vee Age < -5.0$ **then return false**
  end if
  stats.v $++$; UDP.Send($P.Pay, Host_B$) ▷ **Step 7: Delivery**
  return true
end procedure

Security Assumptions for Data Plane

Security assumptions for the data plane are fundamental truths, and violating the assumptions will break the plane's security. This plane assumes that BLAKE2b, SHA-256, and Ed25519 will ensure 256-bit cryptographic security. The sessions remain in a secret memory safely, and session caches are similar to blockchain state. Then it authenticates out-of-band nodes, and finally, it assumes Dolev-Yao network adversaries, which means packet interception or modification can take place, but cannot break the cryptography or the majority of nodes [4].

Security Properties for Data Plane

The data plane has multiple features to ensure security properties are intact. First of all, it guarantees message authentication using HMAC with a forgery probability of 2^{256} for the SHA256 algorithm. The proof-hash binding link message with the original ZK-proofs and reduces 10KB overhead. Moreover, it has replay and protection using packet sequence numbers and timestamps. Finally, the plane ensures that limited forward secrecy is maintained with a timeout of 3600 seconds (1 hour).

4.3.7 System Architecture Summary

Version 4 implements a modular architecture with clear separation of different layers. The Foundation Layer ensures Ed25519 key management [11], HKDF session key derivation, BLAKE2b HMAC authentication, and ZK-SNARK proof generation (o1js). The Session Management Layer preserves session state and lifecycle, Fast O(1) session caching, and ZK-SNARK handshake protocol. The Blockchain Consensus Layer of the system handles

BFT consensus implementation and disk-hybrid blockchain storage. The Network Layer creates binary packet formats, ensures Session establishment with block proposals, verification through consensus participation, and does final validation along with delivery. And the Radio Interface Layer handles message transmission and reception via UDP.

Chapter 5

Result Analysis

5.1 Performance Evaluation

To support a valid simulation-based study of Blockchain radio, only a handful of important methodological cornerstones can be considered, including simulation parameters and assumptions definition, a clear distinction between simulated and inferred metrics, a valid simulation implementation being considered, and a clear understanding of the limitations inherent in a simulation-based study. Such principles form the base of the methodology, which is presented in the following sections.

5.1.1 Key Metrics and Evaluation Criteria

Based on the four versions of the blockchain-enabled radios, the following performance metrics were measured:

Accuracy Metrics: Cryptographic Verification Accuracy is measured in all 4 versions in different cases. We calculated the accuracy of hash integrity detection for accidental corruption for version 01, we took ZKSnark cryptographic proof verification for version 02, HMAC Hash authentication for version 03, and Byzantine fault-tolerant consensus accuracy for version 04, which all gave 100% Duplication Detection rate was calculated in all versions using the UID checking, and all the versions detected the 100% of the duplicated packets. For Version 03 and Version 04 Session validation accuracy was tested, which was able to detect 100% accuracy.

Efficiency Metrics: For efficiency, we calculated the containers' latency from one hop to another based on their processing requirements, individual packet size for space complexity analysis, and network efficiency on packet overhead size. For container latency, the data is given below:

Table 5.1: Efficiency Metrics: Container Latency in Blockchain Latency Metrics

Containers	**V.02**	**V.03 (Block creation)**	**V.03 (After Session)**	**V.04 (Control)**	**V.04 (Data)**
Producer	∼126000 ms	∼126000 ms	∼0.1 ms	∼1260000 ms	0.1603 ms
Repeater (single)	∼0.1 ms	∼0.1 ms	∼0.1 ms	∼12.7 ms	∼0.118 ms
Receiver	∼0.1 ms	∼0.1 ms	∼0.1 ms	∼12.7 ms	∼0.028 ms
TTPP (Total Time per Packet)	∼126200 ms	∼126200 ms	∼0.3 ms	∼126015.4 ms	∼0.3063 ms

For space complexity analysis, version 01 has O(m), where m=unique packets processed for a seen UID set. In version 02, O(m) is 10KB per ZK-Snark proof stored. Space Complexity in versions 03 and 04 is given below:

Table 5.2: Space Complexity Comparison Between V3 and V4 of Blockchain Enabled Radio Network

Terms	**Version 03**	**Version 04**
Session Cache	∼500 Bytes per session	∼500 Bytes per session
Proof-hash Cache	32 bytes per session	N/A
HMAC Key cache	32 bytes per session	N/A
Blockchain Storage	N/A	∼2KB
Default cache capacity	∼564 bytes per session	∼5.5MB for 10000 sessions

For Network Efficiency, we calculated packet overhead for every version. The data is given below:

Table 5.3: Overhead Size in Blockchain Radio Versions

Packet Name	**Version 01**	**Version 02**	**Version 03**	**Version 04 (CP)**	**Version 05 (DP)**
Overhead	100B metadata	10KB proof	112B fixed Header	∼2KB per Proposal	112B fixed Header

Bandwidth reduced from version 02 to version 03, per packet overhead by 98.9% (10KB to 112B) For the 6000 packers session, the bandwidth reduced from 60MB to 672 KB.

Reliability Metrics is calculated on fault tolerance in the first 3 versions, and version 4 used Byzantine Fault Tolerance (Tolerates $f = \lfloor (n-1)/2 \rfloor$) for faulty nodes and consensus requirement: $\lfloor n/2 \rfloor + 1$ signatures. Packet success rate for all versions is 90%, and 1% packet loss for network I/O. Integrity verification was 100% on detecting corrupted packets.

Table 5.4: Attack Protection Capability Blockchain Radio Versions

Attack Scenarios	**Version 01**	**Version 02**	**Version 03**	**Version 04**
Tamper detection:	✓	✓	✓	✓
Replay Protection	X	✓	✓	✓
Sender Authentication	X	✓	✓	✓
MIMT Prevention	X	✓	✓	✓

Session rate consistency hit rate is 99% for version 03 and version 04. Moreover, cache misses are handled by automatic blockchain lookup and recaching. Average lookup time is optimized for a hash table lookup O(1). Consensus agreement rate in version 4 achieved 100 percent in the absence of network partitions, and average consensus time took 0.1 seconds for one node.

Table 5.5: Experimental Test Evaluation Workload Parameters

Parameter	**Value**	**Description**
Message Count	50-1000	Number of voice/data messages transmitted during experiment
Message Interval	1-5 seconds	Time between consecutive message transmissions
Message Size	100-500 bytes	Typical voice codec data or text message size
Test Duration	5-45 minutes	Total duration of each experimental scenario
Warm-up Period	5 seconds	Initial period excluded from measurements to allow system stabilization like docker compose up
Concurrent Sessions	1	Initial period excluded from measurements to allow system stabilization

5.2 Analysis of Design Solutions

The research has used a comparative evaluation methodology of simulation-based research to evaluate the performance of a blockchain-based network in a radio network context. The proposed blockchain-based network has not been designed or implemented in a physical context, nor is it operating on any licensed radio frequency spectrum, but a

Docker-based simulation is used to illustrate the logical aspects of a network, blockchain, propagation, and so on, which can occur in a radio communication context.

The justifiability of simulation as a primary methodology for carrying out the research can be attributed to several reasons. Firstly, it can be attributed to the novelty of utilizing a blockchain-based radio network. Creating physical devices for experimentation was deemed to require considerable resources because it involves obtaining specialized devices for both communicating via RF signals to specific allocated spectra while undertaking substantial regulatory requirements (FCC, 2023). Secondly, simulation can provide for experimentation with specific network conditions that may prove to be impossible to carry out physically (Banks et al., 2005). Thirdly, simulation-based approaches to researching wireless communication systems are widely accepted (Rappaport, 2002; Goldsmith, 2005).

It is also important to note that the following are the evaluation approaches in communication system research:

1. Theoretical comparison: The comparison of different architectures, different designs of protocols, and the different properties of different algorithms through the use of mathematical models.

2. Experimental validation: Using physical hardware to measure system performance within a real environment or setting.

This research essentially falls under theoretical comparison criteria that include simulating experiments. Here, we analyzed all 4 versions of Blockchain Radios based on their accuracy, efficiency, and reliability.

5.2.1 Version 01: Baseline Implementation

Linear chain topology with hash-based integrity verification using simplified zk proof metadata in JSON format. It was able to detect accidental data corruption (Simulated noise), but for no cryptographic implementation, it was easily forgeable, and there was no protection mechanism against replay attacks.

Though the model accuracy was not furnished, it has a higher throughput, a low memory footprint, and minimal computational overhead. In terms of reliability, it has 99.9% consistent packet delivery, but it has no fault tolerance mechanism and a single point of failure design due to its linear chain topology. Overall, it successfully validates networking architecture but lacks in security and reliability. This version is appropriate as a baseline for iterative development.

5.2.2 Version 02: zk-SNARK Integration

Authentic zero-knowledge proofs using o1js framework (Mina Protocol) with TypeScript-Python integration using subprocess communication. In terms of accuracy, it is cryptographically sound for proof generation and verification, digital signatures prevent impersonation, and smart contracts include timestamp validation and replay attack prevention. But for proof generation time takes more than 2 minutes, it's impractical with a 10KB packet overhead and throughput less than 0.0048 packets/second. These make this version unusable for real-time communication. This version moves us to real block generation, cryptographically sound but operationally unusable.

5.2.3 Version 03: Session-Based Authentication

Hybrid architecture separating one-time zk-SNARK for session establishment and continuous HMAC-based message authentication. This introduces a proof-hash binding mechanism. It maintains zk-SNARK Security using proof hash binding. HMAC provides cryptographic authentication per message, and the proof-hash (32B) cryptographically binds the original zk-SNARK (10KB) proof. It prevents replay attacks using the sequence number and timestamp. The cryptographic chain is established using Sha256(zk-SNARK proof) $\rightarrow$ HMAC(proof_hash —— message). Breaking this chain requires either a SHA256 collision resistance compromise or HMAC forgery without the session key. Both are computationally infeasible and cryptographically secure.

Efficiency in this version is reducing 126 second zk-SNARK cost for every packet and amortizing across the entire session. Per message overhead reduced to 0.1ms and packet size reduced to 112B. Throughput increased by more than 6600 packets/seconds.

In terms of reliability, it verifies the packets using sessions in O(1). High cache hit rate and automatic blockchain fallback for cache misses. But it has limitations on its trust mechanism as the producer became the centralized single point of control, which can lead to a single point of failure. The architecture has no consensus mechanism and follows a linear topology with no fault tolerance.

Overall, this version successfully bridges cryptographic security with real-time performance. Proof-hash binding ensures the zk-SNARK guarantees while also achieving HMAC-level authentication speed. The critical limitation of this version is a centralized trust model, which contradicts the blockchain principles.

5.2.4 Version 04: Decentralized Consensus

This architecture uses Byzantine fault-tolerant consensus for session establishment with architectural separation of control plane and dataplane. Here, the control plane is used for blockchain operations, and the data plane is used for message authentication.

This version maintains all cryptographic security (Proof_Hash binding + HMAC) and also prevent single point of failure using collective verification from consensus. This no single point of authority. Here, Ed25519 signatures prove repeater participation and majority agreement required for session establishment. The consensus security properties include Byzantine Fault Tolerance, Tamper Evidence, and Non-repudiation. It also ensures consensus overhead stays minimal and the data plane perform identical to version 3. Parallel signature collection optimizes latency, and Bisk Hybrid storage balances durability and speed.

This version ensures Byzantine Fault Tolerance, unilateral session establishments, authoritative session history, and session synchronization via blockchain storage. Overall, this version achieves a decentralized objective without sacrificing real-time performance, as consensus overhead is negligible compared to 2 minutes of ZK-Proof generation. Successfully transforms centralized session establishment into a distributed agreement network and maintains version 3's performance.

5.3 Final Design Adjustments

Based on the performance evaluation and solution analysis, three major design changes were made to the system versions with regard to balancing the security, performance, and

decentralization requirements. Each change addressed critical limitations of the previous version while creating new challenges.

5.3.1 Adjustment 1: V1 → V2 — Cryptographic Security Implementation

This was the first significant adjustment that was implemented to resolve a fundamental security problem in the initial manifestation of the system. In Version 1, it employed JSON metadata in place of authentic cryptographic proofs, and it only performed hash integrity checks, which turned out to be susceptible to those who had packet interception skills. Also, it did not have any authentication of the sender, which opened an enormous security gap that could be used in adversarial network settings. These problems rendered it unsuitable to be used in security-sensitive applications in which integrity and sender verification are two important factors.

In order to overcome these security problems, Version 2 started employing the o1js framework to produce correct zk-SNARK proofs that would provide the system with high and secure cryptographic protection. It also proposed the RadioMeshIntegrity smart contract that implements the security regulations and ensures that all the checks are appropriately conducted. To ensure that the message is authenticated, Ed25519 digital signatures were employed so that the sender of every message may be verified. Also, SHA-256 has been substituted with Poseidon hash functions, which are more efficient and suitable for zero-knowledge proof circuits.

These changes had a mixed effect that was dramatic. On the one hand, it provided an entirely cryptographically sound system, eliminating any security vulnerability and providing security guarantees. Alternatively, it accomplished it at a terrifying price: it added a deadly performance penalty of 120 seconds per packet to prove generation and another 30 seconds per hop to prove verification functions. In the case of a 10-minute-long call that involves the transmission of packets in a constant stream, 350 hours of cryptographic operation would be required, effectively making real-time calls impossible. This was quite necessary to comply with the security requirements, but it was a clear sign that the optimization needed was yet to be done before it could be used in a real-world situation.

5.3.2 Adjustment 2: V2 → V3 — Session-Based Architecture

The second significant modification was directed at addressing the critical performance problems that were brought about by the per-packet cryptographic verification of Version 2. This was made quantitatively in terms of computing the computational costs of the system in realistic use cases. As an example, a 10-minute voice call would take 350 hours of computation time based on the calculation of proofs every 120 seconds and verification time per network hop (30 seconds). This renders the system totally inapplicable in regard to any real-time communication, such as voice communication, video streaming, and data communication. The per-packet zk-SNARK-based approach, although secure, imposes a barrier that cannot be overcome for any form of real-time communication where latency requirements are measured in milliseconds rather than minutes.

Version 3 of the protocol brought a significant architectural change, where the process of session establishment and message authentication was decoupled, and a hybrid approach,

utilizing the benefits of both heavyweight and lightweight cryptographic techniques, was implemented. The main innovation was verifying a zk-SNARK zero-knowledge proof only once at the start of the session. After that, all messages were authenticated using lightweight HMAC operations, so there was no need for heavy cryptographic calculations on every message. Unlike the previous version, the computations were done just once at the beginning, rather than repeating for each message.

A novel proof hash binding technique was also implemented, which ensured the maintenance of cryptographic integrity throughout this significant architectural transition. In this process, the zk-SNARK zero-knowledge proof was hashed using SHA-256 to produce a compact 32-byte hash called proof_hash. This hash was then used in all HMAC operations for messages within the same session. This created a chain of cryptographic operations that linked the initial zero-knowledge proof to every following message, like this: ZK-proof → proof_hash → HMAC → message.

In addition, other performance optimization measures accompanied this architectural improvement. In this regard, the system adopted a session key derivation function using HKDF (HMAC-based Key Derivation Function) to derive secure session keys from the initial proof. In addition, session lookups in the cache were optimized to run in constant time, O(1), so system performance stays consistent even as the number of active sessions grows. The system also switched from SHA-256 to the BLAKE2b hash function for HMAC, since BLAKE2b performs faster on 64-bit platforms, which are widely used in modern radio systems.

The effect of these changes was tremendous on the performance of the system, while the security guarantees were maintained. The improvement in performance was tremendous, a 10,447 x speedup that reduced the computational time needed for a 10-minute call to just 120.6 seconds, effectively making real-time communication a reality for the first time. The system was able to attain a throughput of 6,667 packets per second, meeting the latency requirements of voice communication and other interactive communication systems. Most importantly, this improvement in performance was realized without compromising the security guarantees that were established in Version 2 of the system, as the session-based system maintained the same level of security guarantees via the proof hash binding mechanism. Nevertheless, this architectural change came with a limitation in the form of centralization of the control of the system. The Producer node was granted the power to unilaterally control the sessions, effectively creating a single point of trust that went against the decentralization principles that are core in the blockchain-based systems. Nevertheless, this change was a critical breakthrough in the system that needed to be refined to overcome the centralization limitation that was created.

5.3.3 Adjustment 3: V3 → V4 — Decentralized Consensus

To tackle these security issues, Version 2 began using the o1js framework to generate real zk-SNARK proofs, giving the system strong and reliable cryptographic protection. It also introduced the RadioMeshIntegrity smart contract to enforce security rules and ensure that all checks were done correctly.

For authentication, Ed25519 digital signatures were used so the sender of each message could be verified with confidence. The architectural design had also meant that a compromised or rogue Producer could easily authorize illegitimate sessions, deny legitimate session requests, or modify session parameters in ways that could not be collectively verified. In addition, the lack of collective mechanisms for the verification of the legitimacy

of the establishment of the session meant that the other nodes in the network had no means of verifying the appropriateness of the decisions regarding the establishment of the session. Session state synchronization between the nodes had also been a challenge, as the nodes could have had differing perceptions of the active sessions, which could have resulted in authentication failures or security vulnerabilities.

Version 4 of the network added a Byzantine Fault Tolerant consensus protocol to distribute session setup authority across all nodes. Here, the Producer node creates a block proposal with the session request and its details. The proposal isn't accepted immediately. Each Repeater node checks it to make sure it meets all security and cryptography rules. When most nodes agree, the block is finalized and sent out to the whole network.

The network is divided into two layers: On port 7000 is the Control Plane, which handles such blockchain operations as posting proposals, verifying them, gathering signatures, and disseminating completed blocks. Since the setup of a session is not a frequent event, this component can tolerate some lag (100-500 milliseconds) and is a broadcast system, thus the proposals are received and processed by all the nodes. Data Plane (Ports 5000 and higher) Data Plane Authenticates real time messages with lightweight HMAC operations, having been introduced in Version 3. This layer is very high-speed as it just takes 0.15 milliseconds per hop. It employs a simple linear chain of nodes in test networks. The proposal is not accepted at once. All Repeater nodes check it out to ascertain its safety. After a majority of the nodes agree, the block is completed and distributed to the entire network.

Data is placed in the blockchain in two forms. Sessions that are completed are saved on disk to ensure their safety and a permanent record. In the meantime, active session data remains in memory to allow messages to be read fast. This arrangement offers the best of both worlds: the disk storage space ensures that the information is safe and can be audited, and the memory is used to obtain the information in real-time and quickly. The system is also able to synchronize caches automatically on the creation of a new block, hence nodes always have current session data without having to go through specific synchronization processes.

This decentralized consensus model addressed the centralization problems while preserving the performance gains of Version 3. Now, everyone in the network shares trust, so no single node is in charge, and setting up sessions is done together. The network can also handle Byzantine faults, staying secure even if almost half of the nodes act maliciously. The consensus process adds minimal overhead (100–500 milliseconds) compared to the long zk-SNARK proof generation and the per-message cost of Version 2. Through efficient session-based HMAC authentication, the network can handle 5,000–6,667 packets per second in the Data Plane. While testing still used a linear chain, switching to a full mesh topology will remove remaining single points of failure. This approach successfully combines cryptography, real-time performance, and decentralized trust in a single system.

5.4 Comparisons and Relationships

For comparison and relationship, we divide it into 2 categories:

- Comparison of approaches across versions
- Comparison with existing radio system design

5.4.1 Comparison and Relations of approaches across versions

Version comparisons are done here based on security, performance trade-offs, and architectural complexity. All of the versions and the proposed blockchain radio network architecture (version 04) are achieved through the application of Docker containerization technology. This implies that each radio node in the radio network comprises a Docker container with the whole software stack for the blockchain client application, the consensus algorithm, the message routing program, zk-SNARK proof generation and verification modules, message routing logic, and network interface management. The implementation uses Python 3.11+ for core networking and blockchain components, and Node.js 20+ with the O1js library for zk-SNARK cryptographic operations.

The container architecture gives several methodological advantages to its users. Most importantly, it allows for a system to be replicated with perfect accuracy across various computational environments. In addition, with containerization, there is a clear 'separation of concerns' about the node logic and the behavior of the system on a network level. Finally, the Docker capabilities facilitate a controlled setup of a network topology, latency simulation, and packet loss simulation without the need for Linux kernel modifications.

- Blockchain Node: In this component, distributed ledger technology is incorporated, where a validation process and consensus method are carried out with regard to blockchain technology itself, appropriate for permissioned networks, where validators are pre-authorized.
- Message Router: Application-level message formatting, routing, and integration of radio-communication primitives with blockchain transaction submission mechanisms.
- Network Stack: This provides TCP/IP network connectivity between containers. Physically, systems use radio systems, which utilize RF modulation, but logically, within a simulation, network sockets can be utilized.
- Monitoring and Metrics Collection: Tracks transaction latency, consensus round length, message confirmation, network overhead, and more metrics.

The logical topology for these network structures will be configured using Docker Compose. Network policies used in the model are of different types depending on the scenarios, such as full-connectivity meshes and a linear chain model for network connectivity, depending on the test case scenarios, like broadcasting and zones.

All Docker containers run on a single host machine with the following specifications.

Table 5.6: Host Machine Configuration

Component	**Specification**	**Notes**
CPU	Intel Core i7-14700K	20 cores (8 P-cores + 12 E-cores), 28 threads, base 3.4 GHz, boost up to 5.6 GHz
GPU	NVIDIA RTX 4080 Super	16GB GDDR6X memory, 10,240 CUDA cores (not utilized in current implementation)
RAM	64GB DDR5	Sufficient for zk-SNARK proof generation which can require 4–8GB per proof
Storage	1TB NVMe SSD	High-speed storage for blockchain I/O operations and Docker volumes
PSU	850W	Provides adequate power for full system load
OS	Linux Ubuntu 22.04 LTS	With Docker Engine 24.0+ and Docker Compose

The NVIDIA GPU is available but not utilized in the current implementation. Future work may explore GPU-accelerated ZK proof generation using CUDA frameworks, which can drastically reduce the proof generation time. However, this optimization step was not taken for the implementation of our current version.

Security changes of approaches across Versions:
Performance Trade-off across versions focuses on time taken in session setup, per message authentication, per node verification, throughput, and packet overhead:

Table 5.7: Security Comparison of Blockchain Enabled Radio Network Versions

Security Property	**Version 01**	**Version 02**	**Version 03**	**Version 04**
Data Integrity	SHA-256 hash	Poseidon hash (zk)	Proof-hash binding	Proof-hash + consensus
Sender Auth	None	Ed25519 signature	Ed25519 (session)	Ed25519 + majority
Replay Prevention	None	Smart contract UID	Sequence + timestamp	Sequence + timestamp
Man-in-the-Middle	Vulnerable	ZK-proof binding	Proof-hash binding	Proof-hash + consensus
Byzantine Tolerance	None	None	None	f = floor((n-1)/2)
Trust Model	Hash-based (weak)	Crypto-based (strong)	Session-based (strong)	Consensus-based (distributed)

Table 5.8: Performance Comparison Among Blockchain Enabled Radio Network Versions

Containers	**Version 01**	**Version 02**	**Version 03**	**Version 04**
Session Setup	N/A	∼126200 ms	∼126200 ms (One time)	∼1260000 ms (One time)
Per-Message Auth	∼0.1ms (hash)	∼120620ms (ZK-proof)	∼0.1ms (HMAC)	0.1603ms (HMAC)
Per-node verify	∼0.1ms	∼0.1ms	∼0.1ms	∼12.7ms
Throughput	10,000 pkt/s	0.00792 pkt/s	10,000 pkt/s	∼3,264 pkt/s
Packet Overhead	∼100 bytes	∼10,000 bytes	112 bytes	112 bytes
Real time capable	✓	X	✓	✓

Table 5.9: Architectural Comparison of Blockchain Enabled Radio Network Versions

Component	**Version 01**	**Version 02**	**Version 03**	**Version 04**
Languages	Python	Python + TypeScript	Python + TypeScript	Python + TypeScript
Cryptographic Primitives	SHA-256	SHA-256, Poseidon, Ed25519	SHA-256, BLAKE2b, Ed25519, HKDF	SHA-256, BLAKE2b, Ed25519, HKDF
External Dependencies	Minimal	o1js, Node.js	o1js, Node.js	o1js, Node.js
Smart Contracts	None	RadioMesh Integrity	RadioMesh Integrity	RadioMesh Integrity
Blockchain	None	Local (proof gen only)	Local (session handshake)	Full BFT consensus
Network Protocols	TCP/UDP	TCP/UDP + Subprocess	TCP/UDP + Subprocess	TCP/UDP (Binary Protocol)

5.4.2 Comparison and Relations with Existing Radio System Design

Comparing the proposed blockchain-enabled radio with the existing technologies requires performance data for a few models, like Project 25 (P25), Digital Mobile Radio (DMR), Meshtastic, and Terrestrial Trunked Radio (TETRA). These radio systems are not simulated in this research, but their characteristics are taken from their published technical standards, peer-reviewed field studies, and manufacturer specifications. The comparison will be done over their system architecture and security for a theoretical comparison.

Project 25 (P25): P25 or Project25 is a digital radio communication standard widely adopted by emergency first responders like police, firefighters, ambulance service and so on. This system was designed to replace older analog radio systems using modern digital technology. The Project 25 network typically consists of mobile radios with fixed stations, repeaters, and key management facilities. The system transmits data at 9600 bits per second using cryptographic encryption [17]. The Radio network uses end-to-end 256-bit AES encryption between all the radio systems that are connected via P25 Inter-RF Subsystem Interface (ISSI), and each communication endpoint decrypts it [40]. Moreover, for using AES, the radio needs to manage the keys that are required. The persistent key management storage is commonplace for P25 communication endpoints to use the hardware cryptographic modules, and the Key Management Facility (KMF) server manages the keys centrally [40]. This makes the radio's security system centralized.

Digital Mobile Radio (DMR): DMR is a digital radio standard developed by the European Telecommunications Standards Institute (ETSI) for Professional Mobile Radio (PMR), offering reliable and cost-effective voice and data communications for mission-critical applications. For instance, APX 2500 single-band P25 Mobile Radio developed by Motorola uses single-key Advanced Digital Privacy (ADP) encryption for its P25 [67]. Any device that uses single-key encryption requires storage to decrypt the data. All packet radios are part of DMR, and it depends on the network system and design whether it is centralized or decentralized.

Meshtastic: Meshtastic is an open source project that gives the ability to use LoRa (Long Range) radio for off-grid communication in areas where no communication platform exists [65]. It uses micro-controllers like ESP32 chips to RP2040 dual-core ARM chips to create radios using LoRa modules and create a radio-level mesh using a set of nodes with the same LoRa spreading factor, center frequency, and bandwidth. The critical point of using a meshtastic is that all nodes should share the same value,s which are grouped into presets and can only be chosen in LoRa configuration. This uses channel and encryption on top to form a logical mesh. According to open source data, it uses AES256-CTR (counter mode) encryption for sending payloads in the mesh with different keys for each channel [66].

Terrestrial Trunked Radio (TETRA): TETRA (Terrestrial Trunked Radio) is a digital radio communication standard broadly used by law enforcement, military, emergency services, transportation, and critical infrastructure operators across europe which is also similar to Project25 used in North America. These applications enable efficient machine-to-machine transfer of various important information and Data from the field to the control centers in real time, in particular [42]. It uses end-to-end cryptographic protection with its connection path capacities along with Internet Protocol connection to the Military internet [28]. It also uses the TETRA Encryption Algorithm (TEA1) for commercial use and TEA2 and TEA3 stream cipher algorithms for emergency situations [52].

Table 5.10: System Comparison Between The Identified System with Blockchain Enabled Radio Network V4

Radio System	**Encryption**	**Authentication**	**Decentralized**	**Prevent Single Point of Failure**	**Trust Model**
BERN	X	✓	✓	✓	✓
Meshtastic [65], [66]	✓	✓	✓	✓	X
P25 [17], [40]	✓	✓	X	X	X
DMR [67]	✓	✓	X	X	X
TETRA [42], [28], [52]	✓	✓	X	X	X

5.5 Discussions

5.5.1 Assumption

This primary research objective is to create a Blockchain-Enabled Amateur Radio Network for real-time voice communication, and this research evaluates an architecture that is theoretically able to create a communication channel using a blockchain trust-based system capable of real-time communication. The simulation result can be independently verified using the documented source code. Physical radios will need expensive equipment, along with a spectrum license and reproducibility limits. This simulation shows proof hash binding mechanism, which accelerates performance and can make the technical foundation necessary to justify investing in physical prototyping. The Blockchain Architecture alone can work as an adapter or bridge for other technologies using radio frequencies, as the producer can be its gateway. As the architectural design, this research has many possibilities to use blockchain mesh in IoT and Radio devices.

5.5.2 Implication

Blockchain Amateur Radio Network version 4 is a decentralized radio protocol that requires initial latency to connect to the network, ensuring security and integrity. This model works for emergency areas where centralized hub establishment is impossible. Disaster-affected areas and grey zones are the best case for this kind of radio architecture, where radio signals can be hijacked or tampered with over long distance and misinformation can be leveraged. Moreover, this architecture is expandable, where the network can grow over time using multiple nodes.

5.5.3 Limitation

Every research methodology has some assumptions and limitations that are bound to the validity and applicability. This research also has its limitations. First of all, no RF interference modeling. Physical radio models have complex electromagnetic fields to operate and a co-channel interface for connected systems, multipath fading, atmospheric absorption, and jamming ell are real-world RF constraints. The RF physical layer characteristics are not modeled in this simulation. Also, the simulated reliability shows best performance in the best-case scenarios with no hardware bottleneck. Actual implementation will experience additional packet loss and noises which will be dangerous for blockchain Radio Deployment. Moreover, it is challenging to create a TCP-based radio mesh where repeaters can validate without physical RF challenges like RF fading and other constraints. The whole system is a simulation based and requires more than 2 minutes to connect to a server-like host machine, which is a big tradeoff.

Chapter 6

Conclusion

6.1 Summary of Findings

The first introduction to the network was version 1, which showed that there existed a possibility of multi-hop packet forwarding and that simple integrity checks were valid in a containerized environment. It was a bare-bones cryptography implementation whose purpose was solely to probe the network dynamics, e.g, the producer-repeater-receiver. Its weaknesses include the absence of genuine evidence, the absence of digital signatures, and a hash-based integrity wmechanism thatis willing to make a profit in Version 2.

Version 2 transformed the prototype into a network that was cryptographically secure. It applied real ZK-SNARK proofs in o1js framework and the Mina Protocol. The integration between Python and TypeScript showed that the high-level code of networking and low-level cryptography could be combined. However, it took an average of 120 seconds to generate evidence of each packet, which was a major performance issue and could not be applied in real time.

This was corrected in version 3 with the help of session-based design and proof hash binding. The computation of the proof was also done once per session instead of the expensive computation of each packet by ZK-SNARK. Each message was verified by a lightweight HMAC, which increased the system speed by about 273,000 times. This not only ensured the cryptographic secrecy but also allowed real-time relay of messages. The evidence was still referred to the original ZK-SNARK evidence, but 32 bytes per packet was now required, as compared to 10 KB required before.

Version 4 finalized the system by incorporating the decentralized blockchain consensus and necessitating the exclusion of the centralized trust assumption with comparatively little overhead of sum1% to incorporate into session establishment. The combination of apparently antagonistic properties, robust cryptographic security, Byzantine fault tolerance, real-time performance, and full decentralization was achieved through the architecturally isolated control plane and high-rate message authentication of the data plane.

Table 6.1: Comparison of Blockchain Enabled Radio Network Protocol Versions 1–4

Property	**Version 1**	**Version 2**	**Version 3**	**Version 4**
Cryptographic Proofs	Simulated JSON	Authentic ZK-SNARK	ZK-SNARK	ZK-SNARK
Digital Signatures	None	Ed25519	Ed25519	Ed25519
Per-Message Overhead	Hash only (∼0.1ms)	∼30,000ms verify	∼0.1ms HMAC	∼0.11ms HMAC
Session Support	No	No	Yes	Yes
Consensus Mechanism	Centralized	Centralized	Centralized	BFT Distributed
Real-Time Capable	Yes*	No	Yes	Yes
Blockchain	No	No	No	Yes (immutable)
Byzantine Tolerance	None	None	None	$f < n/2$
Network Topology	Linear chain	Linear chain	Linear chain	Linear chain
10-min Call Processing	∼2min*	350 hours	122s	123s
Production Ready	No	No	Limited	Yes

6.2 Contributions to the Field

- **proof_hash Binding Mechanism:** Every HMAC is bound to the original ZK-SNARK proof with a 32-byte hash. This ensures that the proofs are safe, they cannot be altered, and verification is quick and reliable.
- **Control/Data Plane Separation:** Seeking to decouple trust setting (control plane) and high-rate message checking (data plane), the system can concur to a consensus without causing a slowdown of real-time operations.
- **Disk-Hybrid Storage:** We have a two-cache implementation; one cache is used to hold active sessions, and the other to hold proof hashes to be able to perform verifications in real-time, with the entire history of a blockchain remaining intact.
- **Binary Wire Protocol:** We designed a native protocol with fixed fields and length-prefixed messages, which are simpler to process, more reliable, and better support TCP streaming than standard frameworks.

6.3 Recommendations for Future Work

Blockchain-Enabled Radio Protocol has been tested in simulation and has outstanding performance in terms of block creation, session establishment, and real-time communication. This gives hope that hardware solutions for blockchain can be a communication relay between any kind of authorized radios, including amateurs in the network. The simulation performance validates a transition to physical hardware. This can help to gain factual insights into RF signal behavior and wireless network integration within the blockchain nodes. The plan for this research is to create a hardware environment for a blockchain network that can process real RF signals and connect two SDRs.

Despite achieving production readiness, several areas remain for future research. First of all, migrating from a linear chain to full-mesh networking with dynamic routing can leverage the network topology. Consensus Optimization might be achieved by implementing advanced BFT protocols (PBFT, Tendermint) for improved liveness. Scalability could be increased through hierarchical caching and sharding for networks exceeding 1000 nodes. And finally, formal verification of consensus properties and an automated session cancellation could increase security enhancement features.

This research paper showcases how blockchain consensus and real-time communication can be made compatible if the system boundaries are carefully designed. Most importantly, the architectural patterns that are developed for the network system protocol, particularly the control-data plane separation and proof_hash binding, might have applications beyond radio mesh networks. This can be utilized in any distributed system that requires both strong security features and low-latency operation.

Bibliography

[1] R. B. Parks et al., "Consumers as coproducers of public services: Some economic and institutional considerations," *Policy Studies Journal*, vol. 9, no. 7, pp. 1001–1011, 1981. DOI: https://doi.org/10.1111/j.1541-0072.1981.tb01208.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1541-0072.1981.tb01208.x. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1541-0072.1981.tb01208.x.

[2] W. A. Beech, D. E. Nielsen, and J. Taylor, *Ax.25 link access protocol for amateur packet radio*, 1997. [Online]. Available: https://files.tapr.org/tech_docs/AX25/AX25.2.2.1997.pdf.

[3] T. A. W. Group, *Automatic position reporting system aprs protocol reference protocol version 1.0*, Aug. 2000. [Online]. Available: https://www.aprs.org/doc/APRS101.PDF.

[4] W. Mao, *A structured operational modelling of the dolev-yao threat model*, Dec. 2003. DOI: 10.1007/978-3-540-39871-4_5.

[5] A. Hevner et al., "Design science in information systems research," *Management Information Systems Quarterly*, vol. 28, pp. 75–, Mar. 2004.

[6] O. C. Ugweje, "Radio frequency and wireless communications," in *The Internet Encyclopedia*. John Wiley & Sons, Ltd, 2004, ISBN: 9780471482963. DOI: https://doi.org/10.1002/047148296X.tie151. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/047148296X.tie151. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/047148296X.tie151.

[7] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *"Theory of Cryptography"*, vol. Vol. 3876, Jan. 2006, pp. 265–284, ISBN: 978-3-540-32731-8. DOI: 10.1007/11681878_14.

[8] K. Peffers, T. Tuunanen, M. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of Management Information Systems*, vol. 24, pp. 45–77, Jan. 2007.

[9] C.-W. Chen and Y. Wang, "Chain-type wireless sensor network for monitoring long range infrastructures: Architecture and protocols," *International Journal of Distributed Sensor Networks*, vol. 4, no. 4, pp. 287–314, 2008. DOI: 10.1080/15501320701260261.

[10] I. Jawhar, N. Mohamed, and D. P. Agrawal, "Linear wireless sensor networks: Classification and applications," *Journal of Network and Computer Applications*, vol. 34, no. 5, pp. 1671–1682, 2011. DOI: 10.1016/j.jnca.2011.05.006.

[11] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," *Journal of Cryptographic Engineering*, vol. 2, pp. 77–89, 2012. [Online]. Available: https://ed25519.cr.yp.to/ed25519-20110926.pdf.

[12] S. M. Dilek, A. Ayranci, A. Şeker, O. Ceylan, and H. B. Yagci, "Ax.25 protocol compatible reconfigurable 2/4 fsk modulator design for nano/micro-satellites," in *2012 20th Telecommunications Forum (TELFOR)*, 2012, pp. 416–419. DOI: 10.1109/TELFOR.2012.6419235.

[13] M. M. Ilić and A. Ž. Ilić, "Convergence of the higher order frequency-domain fem solution to scattering from a moving dielectric slab," in *2013 21st Telecommunications Forum Telfor (TELFOR)*, 2013, pp. 656–658. DOI: 10.1109/TELFOR.2013.6716315.

[14] B. Zieliński, "Effective throughput of ax.25 protocol," *Bulletin of the Polish Academy of Sciences Technical Sciences*, vol. 61, no. No 3, pp. 639–647, 2013. DOI: 10.2478/bpasts-2013-0068. [Online]. Available: http://journals.pan.pl/Content/83769/PDF/14_paper.pdf.

[15] B. Zieliński, "Effective throughput of ax.25 protocol," *Bulletin of the Polish Academy of Sciences: Technical Sciences*, vol. 61, Sep. 2013. DOI: 10.2478/bpasts-2013-0068.

[16] J. Cai, X. Song, J. Wang, and M. Gu, "Reliability analysis for chain topology wireless sensor networks with multiple-sending transmission scheme," *EURASIP Journal on Wireless Communications and Networking*, vol. 2014, 2014. DOI: 10.1186/1687-1499-2014-156.

[17] S. Glass, V. Muthukkumarasamy, M. Portmann, and M. Robert, "Insecurity in public-safety communications: Apco project 25," in *Proceedings of the ... (conference details not specified in source)*, 2014. [Online]. Available: https://www.researchgate.net/publication/264885086_Insecurity_in_Public-Safety_Communications_APCO_Project_25.

[18] M. Swan, *Blockchain: Blueprint for a New Economy*, 1st. O'Reilly Media, Inc., 2015, ISBN: 1491920491.

[19] V. Tomar and V. Bhatia, "Low cost and power software defined radio using raspberry pi for disaster effected regions," *Procedia Computer Science*, vol. 58, pp. 401–407, Dec. 2015. DOI: 10.1016/j.procs.2015.08.047.

[20] G. P. Kw5gp, *"High Speed Multimedia for Amateur Radio (1st edition)"*. Amer Radio Relay League, Apr. 2016.

[21] A. Dorri, S. Kanhere, R. Jurdak, and P. Gauravaram, *Lsb: A lightweight scalable blockchain for iot security and privacy*, Dec. 2017. DOI: 10.48550/arXiv.1712.02969.

[22] A. Johar, W. Osman, and A. Aziz, "Amateur radio communication technology contingency communication in emergency situation," *SHS Web of Conferences*, vol. 33, p. 00 023, Feb. 2017. DOI: 10.1051/shsconf/20173300023.

[23] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *2017 IEEE International Congress on Big Data (BigData Congress)*, 2017, pp. 557–564. DOI: 10.1109/BigDataCongress.2017.85.

[24] S. Abadal, J. Torrellas, E. Alarcón, and A. Cabellos-Aparicio, "Orthonoc: A broadcast-oriented dual-plane wireless network-on-chip architecture," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 3, pp. 628–641, 2018. DOI: 10.1109/TPDS.2017.2764901. [Online]. Available: https://ieeexplore.ieee.org/document/8078211.

[25] B. Bertenyi, R. Burbidge, G. Masini, S. Sirotkin, and Y. Gao, "Ng radio access network (ng-ran)," *Journal of ICT Standardization*, vol. 6, no. 1-2, pp. 59–76, May 2018. DOI: 10.13052/jicts2245-800X.614. [Online]. Available: https://journals.riverpublishers.com/index.php/JICTS/article/view/6435.

[26] C. Mudzingwa and A. Chawanda, "Radio propagation prediction for hf communications," *Communications*, vol. 6, no. 1, pp. 5–12, 2018. DOI: 10.11648/j.com.20180601.12. eprint: https://article.sciencepublishinggroup.com/pdf/10.11648.j.com.20180601.12. [Online]. Available: https://doi.org/10.11648/j.com.20180601.12.

[27] G. W. Perry et al., "Citizen radio science: An analysis of amateur radio transmissions with e-pop rri," *Radio Science*, vol. 53, no. 8, pp. 933–947, 2018. DOI: https://doi.org/10.1029/2017RS006496. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2017RS006496. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2017RS006496.

[28] ETSI – European Telecommunications Standards Institute, "Etsi ts 100 392-2 v3.9.1 (2019-01): Terrestrial trunked radio (tetra); voice plus data (v+d); part 2: Air interface (ai)," ETSI, Tech. Rep., 2019.

[29] T. Justinia, "Blockchain technologies: Opportunities for solving real-world problems in healthcare and biomedical sciences," *Acta Informatica Medica*, vol. 27, p. 284, Dec. 2019. DOI: 10.5455/aim.2019.27.284-291.

[30] X. Ling, J. Wang, T. Bouchoucha, B. C. Levy, and Z. Ding, "Blockchain radio access network (b-ran): Towards decentralized secure radio access paradigm," *IEEE Access*, vol. 7, pp. 9714–9723, 2019. DOI: 10.1109/ACCESS.2018.2890557.

[31] H. W. Silver, Ed., *The ARRL Handbook for Radio Communications*, Ninety-Sixth Edition. Newington, Connecticut: The American Radio Relay League, Inc., 2019, ISBN: 978-1-62595-088-8.

[32] S. K. C.R and P. K.B, *Swot analysis*, Jun. 2020. DOI: 10.21474/IJAR01/17584. [Online]. Available: https://www.journalijar.com/article/46395/swot-analysis/.

[33] G. Camurati, *Security Threats Emerging from the Interaction Between Digital Activity and Radio Transceiver*, Theses, Dec. 2020. [Online]. Available: https://theses.hal.science/tel-03414339/file/CAMURATI_Giovanni_2020.pdf.

[34] D. Di Francesco Maesa and P. Mori, "Blockchain 3.0 applications survey," *Journal of Parallel and Distributed Computing*, vol. 138, pp. 99–114, 2020, ISSN: 0743-7315. DOI: https://doi.org/10.1016/j.jpdc.2019.12.019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731519308664.

[35] M. Dong et al., *Dual-plane switch architecture for time-triggered ethernet*, Sep. 2020. DOI: 10.1145/3386263.3407597. [Online]. Available: https://www.researchgate.net/publication/344931266_Dual-Plane_Switch_Architecture_for_Time-Triggered_Ethernet.

[36] A. Dorri and R. Jurdak, *Tree-chain: A fast lightweight consensus algorithm for iot applications*, May 2020. DOI: 10.48550/arXiv.2005.09443.

[37] T. Lawrence, I. Ali, T. Christopher, and F. Li, "A bandwidth efficient hmac-based authentication scheme for network coding," *Journal of Information Security and Applications*, vol. 55, p. 102 658, Dec. 2020. DOI: 10.1016/j.jisa.2020.102658.

[38] RTL-SDR.com, *Building a homemade fm repeater with a raspberry pi, rpitx and rtl-sdr dongle*, 2020. [Online]. Available: https://www.rtl-sdr.com/building-a-homemade-fm-repeater-with-a-raspberry-pi-rpitx-and-rtl-sdr-dongle/.

[39] C. Antal, T. Cioara, I. Anghel, M. Antal, and I. Salomie, "Distributed ledger technology review and decentralized applications development guidelines," *Future Internet*, vol. 13, p. 62, Feb. 2021. DOI: 10.3390/fi13030062.

[40] Etherstack Pty Ltd, *Advances in p25: Encryption and security*, 2021. [Online]. Available: https://www.etherstack.com/wp-content/uploads/2021/09/Advances-In-P25-Etherstack.pdf.

[41] J. Newell, Q. Mamun, S. ur Rehman, and M. Z. Islam, *A generalised logical layered architecture for blockchain technology*, 2021. arXiv: 2110.09615. [Online]. Available: https://arxiv.org/abs/2110.09615.

[42] S. M. Svrzić, "25 years of the tetra standard and technology for contemporary digital trunking systems of professional mobile radio communications," *Vojnotehnički Glasnik / Military Technical Courier*, vol. 69, no. 2, pp. 426–460, 2021. DOI: 10.5937/vojtehg69-29340. [Online]. Available: https://www.researchgate.net/publication/350481005_25_years_of_the_TETRA_standard_and_technology_for_contemporary_digital_trunking_systems_of_professional_mobile_radio_communications.

[43] M. Javaid, A. Haleem, R. P. Singh, R. Suman, and S. Khan, "A review of blockchain technology applications for financial services," *BenchCouncil Transactions on Benchmarks, Standards and Evaluations*, vol. 2, no. 3, p. 100 073, 2022, ISSN: 2772-4859. DOI: https://doi.org/10.1016/j.tbench.2022.100073. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2772485922000606.

[44] Q. Tang, O. Ermis, C. D. Nguyen, A. D. Oliveira, and A. Hirtzig, "A systematic analysis of 5g networks with a focus on 5g core security," *IEEE Access*, vol. 10, pp. 18 298–18 319, 2022. DOI: 10.1109/ACCESS.2022.3151000.

[45] Z. Wang, B. Cao, C. Liu, C. Xu, and L. Zhang, "Blockchain-based fog radio access networks: Architecture, key technologies, and challenges," *Digital Communications and Networks*, vol. 8, no. 5, pp. 720–726, 2022, ISSN: 2352-8648. DOI: https://doi.org/10.1016/j.dcan.2021.12.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352864821001036.

[46] "Zkproof community reference. "[Online]. Available: https://docs.zkproof.org/reference.pdf.

[47] Dec. 2023. [Online]. Available: https://picockpit.com/raspberry-pi/how-much-does-power-usage-cost-for-the-pi-4/.

[48] F. Alzhrani, K. Saeedi, and L. Zhao, "Architectural patterns for blockchain systems and application design," *Applied Sciences*, vol. 13, no. 20, 2023, ISSN: 2076-3417. DOI: 10.3390/app132011533. [Online]. Available: https://www.mdpi.com/2076-3417/13/20/11533.

[49] P. Bustamante, M. Gomez, W. Lehr, I. Murtazashvili, A. Palida, and M. Weiss, "Examining the us amateur-radio community through a polycentricity lens," *Telecommunications Policy*, vol. 47, p. 102 667, Oct. 2023. DOI: 10.1016/j.telpol.2023.102667.

[50] F. Dolente, R. G. Garroppo, and M. Pagano, "A vulnerability assessment of open-source implementations of fifth-generation core network functions," *Future Internet*, vol. 16, p. 1, 1 2023. DOI: 10.3390/fi16010001.

[51] D. Isie and H. Reza, *Application of blockchain technology & integration of differential privacy: Issues in e-health domains*, Oct. 2023. DOI: 10.20944/preprints202310.0553.v1.

[52] C. Meijer, W. Bokslag, and J. Wetzels, "All cops are broadcasting: Tetra under scrutiny," in *32nd USENIX Security Symposium*, 2023, pp. 7463–7479. [Online]. Available: https://www.usenix.org/system/files/usenixsecurity23-meijer.pdf.

[53] G. Tripathi, M. Ahad, and G. Casalino, "A comprehensive review of blockchain technology: Underlying principles and historical background with future challenges," *Decision Analytics Journal*, vol. 9, p. 100 344, Oct. 2023. DOI: 10.1016/j.dajour.2023.100344.

[54] K. Zhu, "Blockchain technology: Applications, opportunities, challenges, and countermeasures," in *Proceedings of the 2023 International Conference on Finance, Trade and Business Management (FTBM 2023)*, Atlantis Press, 2023, pp. 460–468, ISBN: 978-94-6463-298-9. DOI: 10.2991/978-94-6463-298-9_50. [Online]. Available: https://doi.org/10.2991/978-94-6463-298-9_50.

[55] A. M. Alashjaee, S. Kushwaha, H. Alamro, A. A. Hassan, F. Alanazi, and A. Mohamed, "Optimizing 5g network performance with dynamic resource allocation, robust encryption and quality of service (qos) enhancement," *PeerJ Computer Science*, vol. 10, e2567, Nov. 2024, ISSN: 2376-5992. DOI: 10.7717/peerj-cs.2567. [Online]. Available: http://dx.doi.org/10.7717/peerj-cs.2567.

[56] D. Commey, S. Hounsinou, and G. Crosby, *Securing blockchain-based iot systems with physical unclonable functions and zero-knowledge proofs*, May 2024. DOI: 10.48550/arXiv.2405.12322.

[57] A. Kharche, S. Badholia, and R. K. Upadhyay, "Implementation of blockchain technology in integrated iot networks for constructing scalable its systems in india," *Blockchain: Research and Applications*, vol. 5, no. 2, p. 100 188, 2024, ISSN: 2096-7209. DOI: https://doi.org/10.1016/j.bcra.2024.100188. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2096720924000010.

[58] A. Pathak, I. Al Anbagi, and H. Hamilton, "Blockchain-enhanced zero knowledge proof-based privacy-preserving mutual authentication for iot networks," *IEEE Access*, vol. PP, pp. 1–1, Jan. 2024. DOI: 10.1109/ACCESS.2024.3450313.

[59] H. Chen, R. Zhou, Y.-H. Chan, J. Zhihan, X. Chen, and E. Ngai, *Litechain: A lightweight blockchain for verifiable and scalable federated learning in massive edge networks*, Mar. 2025. DOI: 10.48550/arXiv.2503.04140.

[60] J. Oberst, *Towards stateless clients in ethereum: Benchmarking verkle trees and binary merkle trees with snarks*, 2025. arXiv: 2504.14069 [cs.CR]. [Online]. Available: https://arxiv.org/abs/2504.14069.

[61] C. Ranganathan and C. Srinivasan, "Enhanced cloud data security by employing hmac for advanced cryptographic protection," *Innovations in Intelligent Systems and Advanced Engineering*, vol. 1, pp. 1–10, Feb. 2025. DOI: 10.29284/IISAE.1.1.2025.1-10.

[62] A. P. Singh, *Blockchain technology: Core mechanisms, evolution, and future implementation challenges*, 2025. [Online]. Available: https://arxiv.org/html/2505.08772v1.

[63] Federal Communications Commission, *47 CFR §97.113 - Prohibited Transmissions*, 2026. [Online]. Available: https://www.ecfr.gov/current/title-47/chapter-I/subchapter-D/part-97.

[64] R. F. Campbell, *Analysis of various algorithmic approaches to software-based 1200 baud audio frequency shift keying demodulation for aprs*. [Online]. Available: https://digitalcommons.calpoly.edu/theses/1634.

[65] Meshtastic Project, *Meshtastic: About*, n.d. [Online]. Available: https://meshtastic.org/docs/about.

[66] Meshtastic Project, *Meshtastic: Encryption overview*, n.d. [Online]. Available: https://meshtastic.org/docs/overview/encryption/.

[67] Motorola Solutions, *Apx 2500 mobile radio datasheet*, n.d. [Online]. Available: https://www.motorolasolutions.com/content/dam/msi/docs/business/products/astro25radios/documents/apx2500_mobile_datasheet_eng.pdf.