National College of Ireland

Lightweight Group Key Management Protocol for Decentralized Fog Computing Environments

MSc Research Project
MSc In Cyber Security

GAYATHRI DEVANUR NAGARAJU
Student ID: 23299193

School of Computing
National College of Ireland

Supervisor: Dr. Mosab Hamdan

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	GAYATHRI DEVANUR NAGARAJU
Student ID:	23299193
Programme:	MSc In Cyber Security
Year:	2024
Module:	MSc Research Project
Supervisor:	Dr. Mosab Hamdan
Submission Due Date:	11/08/2025
Project Title:	Lightweight Group Key Management Protocol for Decentralized Fog Computing Environments
Word Count:	8000
Page Count:	26

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	11th September 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Lightweight Group Key Management Protocol for Decentralized Fog Computing Environments

GAYATHRI DEVANUR NAGARAJU
23299193

Abstract

Secure group communication in fog environments is challenging because of the dynamic nature of the node and the limited resources of edge devices. This research introduces LGKMP-Fog, a lightweight group key management protocol designed for decentralized fog environments. The protocol uses a hierarchical structure with a Domain Leader, Cluster Leaders, and Fog Nodes, and used a prime-seeded Mersenne Twister pseudo-random number(PRNG) generator to generate synchronized keys across clusters. LGKMP-Fog supports both event and time-based rekeying to ensure forward and backward secrecy, and includes features for seed validation, replay attack detection, and removal of compromised nodes

In this work, the Mersenne Twister PRNG is act as the prototype and evaluation used for ns-3 simulations (all results are based on this prototype), while the final crypto-correct specification is defined with a NIST-approved DRBG (CTR_DRBG) for the epoch secret, HKDF-SHA256 for per-round key derivation, and AEAD (AES-GCM) for the data plane.This separation allowed full evaluation of the MT-based prototype in ns-3, while ensuring that the final specification, defined with DRBG, HKDF, and AEAD, aligns with cryptographic standards even though it was not benchmarked end-to-end.

Keywords: Fog Computing, Group Key Management, Lightweight Security Protocol, Decentralized Architecture, Pseudo-random Key Generation, Rekeying Mechanism, Iot Security

1 Introduction

The Internet of Things (IoT) continues to grow rapidly, increasing the scale and criticality of edge and fog systems. By the end of 2023, there were 16.6 billion connected IoT devices worldwide, almost of 18.8 billion by the end of 2024, a 13% increase yearly Sinha (2024). The increase in the number of attacks has created pressure on security systems used for multicast or group communication, for instance, in live video, vehicle-to-everything (V2X) communication, and telemetry at the network edge Yan et al. (2022).

In fog computing, wireless and unstable connections create issues such as eavesdropping, denial-of-service attacks, and failures in updating keys during frequent join/leave events Khan et al. (2017) . This makes group key management (GKM) important for both performance and security. At the same time, fog computing moves data processing closer to users to reduce delays and lightening the load on the main network Alraddady et al. (2022) . However, there are challenges such as limited device resources and constantly

changing group members' situations (dynamic membership), where many traditional security methods have trouble performing. Classical centralized Group Key Management (GKM) handles all key generation and distribution at a single controller Dong and Xu (2020). This makes policy control simple, but also creates a single point of failure and limits scalability, because during membership changing, triggers rekeying across the entire network. Early decentralized approaches, such as Hydra's subgrouping method and time-driven protocols like Kronos,have helped to reduce the central bottleneck. However, there is still coordination and synchronization overhead, along with significant message costs when any members join or leave. These issues are problematic for fog nodes that operate over resource-limited areas. Recent surveys confirm the similar challenge (Prantl et al.; 2022): existing schemes either do not scale well, operate heavy cryptographic processing on resource-constrained devices, or depend on strict time synchronization, which is difficult to maintain in edge computing environments. In fog networks, multicast which is one-to-many communication, is essential for distributing control and data streams across different clusters.

A reliable group key management (GKM) layer must ensure data confidentiality, integrity, and forward/backward secrecy , while coping with frequent joins/leaves and unstable links.In fog environments, prior work shows the need to balance rekey traffic and delay without weakening security Dong and Xu (2020). Surveys of IoT systems also highlight that the GKM layer is central to secure group communication (Prantl et al.; 2022). In practice, we therefore need a lightweight, sync-friendly GKM that keeps signalling low and avoids heavy cryptography. Recent designs for constrained networks include LMGROUP for IoT, which combines ECC (for setup) with HMAC (for message checks) to reduce communication cost while resisting replay attacks Bideh (2022). However, LMGROUP relies on centralized coordination and does not fully resolve synchronization issues in fog environments with intermittent connectivity and high mobility.

To address these limitations, this thesis introduces LGKMP —Fog, which is designed to provide secure multicast communication using a decentralized, layered structure. LGKMP uses the hierarchy method: Domain Leader → Cluster Leaders → Fog Nodes. It uses a prime-seeded pseudo-random number generator (PRNG) to create synchronized group keys across all clusters with minimal communication.Rekeying is done in two ways : periodic (time-based)and event-based (when a node joins or leaves).The NS-3 simulation results showed that the LGKMP-Fog can easily rekey (56 µs), generate strong random keys (validated using entropy and chi-square tests), and maintain very low communication overhead. This makes it lightweight, scalable, and suitable for real fog/IoT systems.The structure of the proposed LGKMP-Fog protocol is shown in Figure 1 below .

The thesis is structured as follows: **Section 2** gives the background and related work on group key management in fog computing. Section 3 explains the design of the proposed LGKMP-Fog protocol. Section 4 describes the simulation setup and method used. **Section 5** presents the results from our evaluation. Section 6 discusses the security features of the protocol. **Section 7** concludes the work and future research.

Note : This dissertation shows results from a prototype of LGKMP-Fog that used the Mersenne Twister PRNG in ns-3 as the evaluation engine for performance tests. To also meet cryptographic standards, a final design was defined using a NIST-approved DRBG for randomness, HKDF-SHA256 for key derivation, and AEAD (AES-GCM) for the data plane. The final design was tested to check that it works (key generation, latency samples, and AEAD checks), but it was not fully measured in ns-3 because of time and system

limits. The obtained results are practical and while the final design still follows modern cryptographic standards.

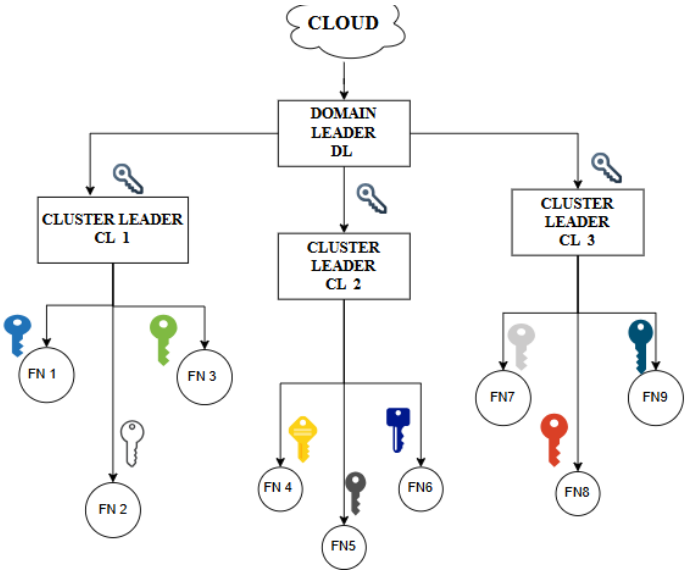

Figure 1: Hierarchical Architecture of LGKMP-Fog for Decentralized Systems

2 Related Work

1. Matrix/XOR and Rekeying – Efficient approaches : Recent studies have attempted to improve GKM for fog environments by using novel key update mechanisms. For example, (Kumar et al.; 2023) introduced a group key management scheme (GKM) on an asynchronous cipher with XOR vector space and PRNG for fog networks. Their design adopts multi-tier structure with Area Local Leaders (similar to cluster heads) to manage the local rekeying, which matches well with fog computing's distributed style . This approach provides secure key updates with forward and backward secrecy ,limits rekeying to local areas and reduces the effect of membership changes on the whole network . However,Kumar et al. (2023) 's model assumes the PRNG is perfectly secure and does not completely address the potential weaknesses of practical. And they also do not test the randomness quality of their generated keys using statistical methods, so there is no proof that the keys are truly unpredictable.Similarly (Junejo and Komninos; 2020)

focused on lightweight security for fog-enabled systems using attribute-based encryption for access control and does not provide group key management or fast rekeying capabilities. It does not check entropy or randomness for keys, since the focus was not on group key creation. These works showed that they emphasized either with efficient key distribution or authentication, but rarely both together in a way that fits the needs of dynamic fog networks.

2. Secure Group Communication Surveys: The challenges addressed in our scheme is also discussed in recent surveys Prantl et al. (2022) reviewed secure group communication methods for IoT and their findings show that many existing methods were not designed for resource-constrained fog nodes . For example, very few protocol support lightweight operation with fast rekeying which are suitable for fog nodes with

limited resources. Many protocols ignore the important features such as key randomness or allowing fog-level nodes manage keys independently. Another related approach is the Blockchain-based IoT (BIoT) framework propsoed by (Kaur and et al.; 2022), which integrates blockchain for secure group access control in distributed systems. While blockchain can improve trust and transparency, this method added a lot of overhead and delay, which makes it difficult to use on resource-constrained fog nodes. Prior research shows that there is a need for a new protocol that combines strong security with efficient operation , avoids single points of failure through decentralization, reduces communication overhead during rekeying,and provides strong cryptographic protection (such as high-entropy keys and defense against replay or impersonation attacks) without overloading devices. This is the gap that LGKMP-Fog aims to address.

3. Decentralized GKM in IoT/Fog: In recent years, new group key management (GKM) solutions for IoT and fog computing have been proposed. The strategies aim to simplify operations and accelerate rekeying. (Dammak et al.; 2020) introduced DLGKM-AC, a Decentralized Lightweight GKM architecture for access control in IoT environments. Their design implements a layered model where the main Key Distribution Center (KDC) is at the top and many smaller Sub-KDCs manage different IoT domains. The Sub-KDCs handle local key distribution, which reduces cloud dependency ,spreads the workload , helps the system in scale better. It also lowers storage, computation, and communication costs compared to a centralized system, and reduces rekeying load on the main KDC. However, it does not focus on making rekeying faster. When a membership change occurs are still updated through the Sub-KDC hierarchy, which can cause delays. Although control is more decentralized where main KDC that distributes initial keys could represent a point of failure (though it is used less often). In contrast, (Iqbal et al.; n.d.) worked on providing security during key exchange in fog computing using elliptic curve cryptography (ECC) for stronger security. Their protocol is designed for fog-enabled wearable health monitoring, provides mutual authentication between devices and fog nodes , creates a shared secret using ECC. This improves trust because both the parties can check the other's identity,helps to stop impersonation attacks. Although ECC is much lighter than RSA, it still needs more computing power for low-powered devices. The use of scalar multiplications on elliptic curves can cause issues on simple fog nodes. Due to this, Iqbal et al. (n.d.), methods strong in authentication, may not work well if many devices join or leave often, or if keys need to be updated frequently, as repeated ECC handshakes could be heavy for devices with limited resources. Neither DLGKM-AC nor ECC-based scheme Iqbal et al. (n.d.),directly focused on fast group rekeying or on testing how random the generated keys .These features are important for secure multicast in fast-changing fog networks

4. Standards-based Secure Group Communication: Recent studies highlight of using the importance of using NIST-approved cryptographic For example, Group-OSCORE Benjamin and Wood (2022) builds on the COSE and uses strong encryption methods (AES-GCM, ChaCha20-Poly1305) for constrained IoT devices, while Messaging Layer Security Barnes et al. (2023) defines a tree-based group key schedule with HKDF and AEAD to ensure strong forward and backward secrecy. These standards show a clear trend: lightweight simulation-friendly methods (e.g., PRNG-based) are useful for evaluation, but production protocols must integrate DRBGs, HKDF, and AEAD to meet modern security requirements. Our work follows this path by using Mersenne Twister as the prototype for ns-3 evaluation, while defining a crypto-correct specification based on DRBG+HKDF+AEAD.

The above mentioned studies have tackled the challenges by implementing lightweight key updates, enhancing authentication mechanisms, or employing blockchain technology for trust; however, they continue to exhibit several limitations:

- High communication and processing overhead during rekeying.
- Depends on central system which reduces scalability .
- No statistical testing of key randomness, which can weaken security.
- Few built-in protections against replay or impersonation attacks at the fog layer.

To address these gaps, our research proposes the LGKMP-Fog a decentralized, three-layer key management system. In this design, a prime-number-based seed is sent from a Domain Leader to several Cluster Leaders, allowing them to generate keys in sync without inter-cluster communication.The protocol provides forward and backward secrecy, blocks attackers using seed mismatch and IP filtering, and checks key randomness with Shannon entropy, chi-square, and p-value tests.

It is important to understand that lightweight PRNGs (e.g., MT19937) are often used in simulation-based GKM studies for feasibility, they are not cryptographically secure. To align with NIST recommendations and current best practices, modern secure group communication protocols adopt DRBGs (NIST SP 800-90A) for randomness, HKDF (RFC 5869) for key expansion, and AEAD modes such as AES-GCM or ChaCha20-Poly1305 (RFC 5116) for data confidentiality and integrity. These primitives ensure unpredictability, forward/backward secrecy, and authenticity. Therefore, in our work we position MT19937 as the prototype/evaluation engine in ns-3 simulations, while defining a final crypto-correct specification based on DRBG+HKDF+AEAD to align with these standards.

3 Proposed Methodology

This research adopts a simulation-based evaluation to design, build, and test the proposed group key management protocol. The primary aim is to ensure that LGKMP-Fog meets its objective of being lightweight, having fast rekeying, and providing strong security. We used NS-3 Network Simulator 3 nsnam (2019) as the simulation tool because it is flexible for creating network setups and events, and has been widely applied in IoT and network security studies Ramonet et al. (2024). Simulation provides a controlled environment to run tests in a repeatable and controlled way, which is especially important when evaluating cryptographic protocols that cannot yet be deployed at scale in real-world systems.

The methodology integrates:

Requirements Analysis – Reviewing the existing literature and requirements for group key management in fog networks.

Protocol Design – Designing LGKMP-Fog's hierarchical architecture which uses a prime-seeded PRNG for key generation and supports both time-based and event-based rekeying.

Simulation– Implementing the protocol in NS-3 to show the fog network including node join, leaves, simulated attacks.

Evaluation – Measuring the performance and randomness using the entropy test, chi-square test and measure rekeying latency .

Validation – Comparing our results with prior group key management research to demonstrate improvements.

Research Design: We designed the protocol structure and algorithms (detailed in Section 4), and we designed and implemented them in an NS-3 simulation model. The simulated fog computing network was divided into clusters, with events like node joining,leaving, and attacks simulated at set times. Custom NS-3 code was added to measure system performance and how to secure it after each event.

Simulation Setup: The network topology was arranged in a three-tier hierarchy. We simulated a Domain Leader (DL) and 3 Cluster Leaders (CLs), with each cluster managing 3 Fog Nodes (FNs) so the total is DL + 3 CLs + 9 FNs = 13 nodes. This topolgy is kept small for clarity. However, the design scales to larger networks (but in this simulation, we keep it at this size for clarity). The CL's acts as a gateway for their FNs the DL is connected to all the cluster leaders. The DL is assumed to be a more powerful node,while cluster leaders and fog nodes have limited resources similar to real edge devices.

Simulation Parameter:

- **Prime Domain Seed**: We used a large prime number (S = 32452843) this is selected by the Domain Leader to start the key generation. Using a prime number gives the PRNG a strong and unpredictable starting point.

- **PRNG Algorithm:** We used Mersenne Twister (MT19937) PRNG, which creates a long sequence 32-bit random numbers. MT19937 was selected because of its long period ($2^{19937} - 1$)) and can generate an identical sequence if the same seed is used .Its important for synchronised key generation across cluster. While MT is sufficient for our simulation and works well for randomness test, and its predictability is reduced by keeping the seed value secret and mixing it with the rekey round number.

- **Key Update Interval:** The group key changes every 300 seconds (5 minutes) for periodic rekeying,which means every 5 minutes , the system will automatically generate a new group key across all clusters. This balances security (limit key lifetime) and avoids too frequent updates that waste resources. We chose 5 minutes based on common values in research, but it can be changed depending on the application.

  Design Note – Prototype vs. Final Specification: For practical feasibility, the ns-3 implementation used the Mersenne Twister PRNG as the *prototype and evaluation engine* to generate synchronized round keys. In parallel, the *final crypto-correct specification* adopts a DRBG→HKDF key schedule and AEAD-secured data plane. This dual-layer approach ensures that the evaluation reflects system scalability and performance, while the specification ensures compliance with NIST cryptographic standards.

4 Design Specification

The proposed group key management protocol is built around the natural structure that is found in most of the fog computing setups. We define a three-level hierarchy consisting of a Domain Leader, several Cluster Leaders, and multiple Fog Nodes, each of which has a specific role in key management AbuAlghanam et al. (2022) :

Domain Leader (DL): This node is the topmost node ,which is responsible for managing the group key distribution across the entire fog network . The Domain Leader creates the key (secret seed value) and send it to the cluster leader and disseminates the necessary keying material to Cluster Leaders. It also triggers rekeying when needed across clusters.The DL is assumed to be secure and trusted because the same keys are generated from a shared seed.

Cluster Leaders (CL): These are special fog nodes like gateway devices or edge servers that manage a group of nearby fog nodes for instance, (sensors in a smart city area). Each cluster leader receives the seed form the domain leader and they use it to generate the group key using random number generator. Cluster Leaders maintain the current group keys which is the same and synchronised across all the clusters and they also handle when a fog node joins or leaves the group and inform the domain leader about the changes, so a PRNG rekey can happen if needed adding or removing fog nodes in their cluster.

Fog Nodes : These are the regular devices such as IoT devices, sensors, actuators which uses group key to encrypt and decrypt messages.They do not generate or decide group keys ; they just receive from their Cluster Leader . To protect the seed ,Fog nodes don't have a direct access to this.If a node leaves the group it won't receive new keys from the previous associated cluster. This ensures forward secrecy

The architecture we suggest is similar to other hierarchical key distribution systems used in IoT. For example, AbuAlghanam et al. (2022) also used a multi-layer key distribution with a mix of protocols. It shows that their method improves scalability and reliability in IoT smart cities. Their results showed more than a 50%drop in communication and computation costs by using hierarchical key management. This supports our choice to use a layered model in fog computing. It naturally reduces overhead and can easily scale to more devices. Our hierarchical design also gives a balance between centralized and distributed models. The Domain Leader manages the first setup and occasionally global rekeys, while most common tasks (like adding a new node) are handled by the cluster leader. This cuts down unnecessary communication in most cases, only the affected cluster updates the key, and then all clusters calculate the same new key,locally without involving every single node or the Domain Leader.

Key Generation Formula Using Mersenne Twister :

The core of LGKMP-Fog is its key generation method, which uses a pseudo-random number generator (PRNG).All cluster leaders use the same PRNG algorithm (MT19937, called the Mersenne Twister) with the same seed value given by the Domain Leader. The Mersenne Twister was chosen because it can create a high-quality stream of random numbers very quickly. Even though originally it was not made for cryptography, MT's output passes many randomness tests. Most importantly, if two systems use the same seed, they will produce the same sequence of numbers. This feature makes synchronization easy: instead of sending new keys over the network, each cluster leader can simply calculate them on its own.

We define the group key at any time as a function of the seed and the rekey round. Let S be the prime seed selected by the Domain Leader (DL).

Let PRNG MT(S) be the Mersenne Twister random number generator www.sciencedirect.com (n.d.), started with seed S. Let's rekey rounds as r = 0, 1, 2, ... (where round 0 is the first key created at time t = 0 when the system starts).

For a given round r, each Cluster Leader makes the new group key by taking the next

output from the PRNG.

The key for cluster i at round r is:

$$K_i(r) = PRNG_{MT}(S + r) \texttt{ -> } \textbf{next\_number}()$$

• Initial Round ($r = 0$): All Cluster Leaders (CLs) start with the same seed value. Each CL calculates the first key using:

$$K^{(0)} = \text{PRNG}_{\text{MT}}(S) \rightarrow \text{next}()$$

This gives the initial group key K^0, which is sent to all fog nodes when the system starts.

• Periodic Rekey Round: At a fixed time interval (for example, every 300 seconds), the system increases the round number r by 1. Each CL then calculates the new key using:

$$K^{(r)} = \text{PRNG}_{\text{MT}}(S + r) \rightarrow \text{next}()$$

Because all Clusters use the same formula, they obtain the same keys $K^{(r)}$. Each CL then securely sends this new key to its fog nodes (using an encrypted channel or pre-shared link keys).

• Join/Leave Triggered Round: If a node joins or leaves a cluster, the cluster sends a signal for an immediate rekey. Either the Domain Leader (DL) or the CL depending on the design, increases the round number r by 1. All CLs then either get this rekey signal from the DL or coordinate with the DL to make sure they use the same r. Each CL then calculates the new key using:

$$K^{(r)} = \text{PRNG}_{\text{MT}}(S + r) \rightarrow \text{next}()$$

The key is generated locally in all clusters, so no key is sent over the network; only a short trigger message (for example, "rekey" to mean "round = 3") is sent. This message is broadcast and does not contain the actual key. If an attacker intercepts it, they only know a rekey occurs, not the new key itself

Design Note – Crypto-Correct Specification The formulas above reflect the MT prototype used for evaluation. In the final specification, the Domain Leader generates a 256-bit epoch secret using a NIST DRBG (CTR_DRBG), and Cluster Leaders derive per-round keys with HKDF-SHA256 using `info = groupId‖round`. The data plane replaces XOR with AES-GCM, where the nonce is `groupId‖round‖counter` and the AAD is `groupId‖round`. This specification was validated functionally (keys.csv outputs, AES-GCM self-test), though simulation results presented later are based only on the MT prototype.

This synchronized PRNG rekeying in the first setup means no inter-cluster communication is required to share new keys. The only messages sent are very small triggers to increase the round number, and this can be skipped when all nodes are in sync with time. Cluster Leaders (CLs) need not to send new keys to each other or the Domain Leader (DL) because they all calculate the keys on their own. This makes the system more scalable,adding more clusters does not increase rekey traffic, and it's more secure no new key is sent over the network.

To explain how the method works, here is a step-by-step timeline from our NS-3 simulation

Key Generation Workflow (Compact)

t = 0s (start): Domain Leader selects seed → all 3 Cluster Leaders derive $K_0 = 624071097$; FogNode1–FogNode9 receive K_0.

t = 0–900s: Scheduled rekey every 300s → t=300s: $K_1 = 455441199$; t=600s: $K_2 = 566748269$ (periodic refresh).

t = 900s (join): FogNodeJoin requests Cluster2; CL verifies and triggers global rekey → all clusters set $K_3 = 676894009$ at 900s; new node gets K_3 only (backward secrecy), existing nodes switch to K_3.

t = 1200–2400s: Scheduled keys every 300s → $K_4 = 478010110$ (1200s), $K_5 = 337870263$ (1500s), $K_6 = 170991997$ (1800s), $K_7 = 190272754$ (2100s), $K_8 = 614238873$ (2400s).

t = 2700s (leave): FogNode3 leaves Cluster1; CL triggers rekey and DL signals all clusters → $K_9 = 149633628$ at 2700s (forward secrecy — leaver can't read future data).

t = 3000s & 3300s: Scheduled updates continue → $K_{10} = 4873630$ (3000s), $K_{11} = 238209785$ (3300s).

Keys rotate every 300s; a join at 900s → K_3, a leave at 2700s → K_9; rotation + event-driven rekeying enforce backward and forward secrecy.

This shows that we used both regular and event-based rekeying to increase security. Regular (time-based) rekeying provides backward secrecy by refreshing keys at fixed intervals. Event-based rekeying adds both forward and backward secrecy as soon as membership changes. This matches best practice for scalable, dynamic group key management Gebremichael et al. (2022),. We followed a time-based method with event overrides, similar to Kronos, but we improved it with on-demand updates and no key-chaining, which avoids Kronos's weakness of linking keys together.

Security Goals Assumptions To clarify the intended security model of LGKMP-Fog, we outline the assumptions, goals, and non-goals of the current prototype. These statements evaluation directly to what was implemented in NS-3.

Assumptions:

- The Domain Leader (DL) is trusted and uncompromised.
- DL→CL trigger messages (for periodic or event-based rekeying) are authenticated and not forged.
- Some fog nodes may become compromised or leave the group.

Security Goals:

- Forward secrecy: nodes that leave should not access future group keys.
- Backward secrecy: nodes that join should not access past group keys.
- Replay detection: outdated group data packets are rejected if received with old timestamps or sequence numbers.
- Availability: the system continues to provide valid group keys despite join/leave churn, keeping clusters synchronized.

We make sure rekey events always work for all clusters. In our simulation, when a rekey was triggered the success rate was 100% all clusters made the new key, and all active nodes received and used it. There were no missed updates because our design does not pass keys through multiple nodes each Cluster Leader updates at the same time. If a Cluster Leader was temporarily down, it would not make the new key, and its fog nodes would be out of sync (meaning they are out of the group until the leader comes back and syncs again). This is an acceptable safety measure. If a Cluster Leader is temporarily unavailable, the fog nodes in that cluster will be out of sync until the leader comes back and updates them. This is a normal safety step in dynamic secure multicast systems to stop the use of compromised or outdated keys Hui et al. (2024). When we simulate in our NS-3, this issue did not occur, which shows that synchronization stayed stable.

Security Scope of the Prototype This work shows a proof-of-concept for decentralized rekeying in fog environments using ns-3 simulation. The main ns-3 tests used the Mersenne Twister PRNG and XOR-based payload handling as a lightweight prototype to measure rekeying behavior, synchronization, and overhead during join/leave events and under loss or delay. At the same time, the final crypto-correct design was also built, using a NIST DRBG for randomness, HKDF-SHA256 for per-round key derivation, and AEAD (AES-GCM). These parts were functionally tested (key generation, latency traces, and AEAD verdicts), but large-scale ns-3 benchmarks were only done with the MT prototype because of time and resource limits. This separation allowed us to study protocol behavior in practice while also making sure the final design follows modern cryptographic standards.

Mathematical Representation

Let $S \in \mathbb{P}$ be a large prime domain seed chosen by the Domain Leader, and let $r \in \mathbb{N}$ represent the current round number. Suppose the fog network has C clusters, each with n_i fog nodes. The total number of fog nodes is:

$$N = \sum_{i=1}^{C} n_i$$

Key Generation via PRNG Each Cluster Leader generates a group key using a pseudo-random number (PRNG) based on the seed S and the round number r:

$$K^{(r)} = \mathrm{PRNG}_{\mathrm{MT}}(S + r), \quad K^{(r+1)} = \mathrm{PRNG}_{\mathrm{MT}}(S + r + 1)$$

All Cluster Leaders derive the same key for a given round:

$$K_i^{(r)} = K^{(r)}, \quad \forall i \in \{1, 2, \ldots, C\}$$

Encryption and Decryption Consistency For any legitimate node in the group:

$$\mathrm{Dec}_{K^{(r)}}(\mathrm{Enc}_{K^{(r)}}(m)) = m$$

If a node uses a different round key $K^{(r')}$ where $r' \neq r$, decryption fails:

$$\mathrm{Dec}_{K^{(r')}}(\mathrm{Enc}_{K^{(r)}}(m)) = \perp$$

Key Uniformity and Forward/Backward Uncertainty Assuming uniform randomness of keys:

$$\Pr[K^{(r+1)} = x] = \frac{1}{|K|} \pm \varepsilon, \quad \Pr[K^{(r-1)} = x] = \frac{1}{|K|} \pm \varepsilon$$

An attacker with a different seed $S' \neq S$ cannot predict the group key:

$$\Pr[\text{PRNG}(S' + r) = K^{(r)}] \leq \delta$$

Key Randomness Evaluation Using Shannon entropy Davies and Macfarlane (2022):

$$H(K^{(r)}) = -\sum_{k \in K} p_k^{(r)} \log_2 p_k^{(r)}$$

Chi-square test for uniformity of key distribution Davies and Macfarlane (2022) :

$$\chi^2 = \sum_{b=1}^{B} \frac{(O_b^{(r)} - E_b^{(r)})^2}{E_b^{(r)}}$$

Where $O_b^{(r)}$ and $E_b^{(r)}$ are the observed and expected frequencies of key bytes in bin b.

To make the protocol clearer, we provide a step-by-step pseudocode representation. Algorithm 1 shows how LGKMP-Fog performs initialization, node join/leave handling, packet validation, and replay attack detection. This pseudocode links the mathematical model with its practical execution in the fog environment.

Mathematical Notations

S	Prime domain seed chosen by the Domain Leader
r	Rekey round number
C	Number of clusters
n_i	Fog nodes in cluster i
N	Total fog nodes, $N = \sum_{i=1}^{C} n_i$
$K^{(r)}$	Group key for round r
$\mathcal{K}$	Key space
$\|\mathcal{K}\|$	Size of key space
ε	Deviation from uniformity
δ	Max attacker success probability
$p_k^{(r)}$	Probability of key k in round r
B	Bins in χ^2 test
$O_b^{(r)}, E_b^{(r)}$	Observed/expected frequency in bin b

Algorithm 1 LGKMP – Lightweight Group Key Management Protocol

S – Prime domain seed
r – Round number
N – Set of fog nodes
C – Set of cluster leaders $K^{(r)}$ – Group key for round r
Secure communication among valid nodes
Initialization:
Cluster Leader $CL_i \in C$ computes $K^{(r)} \leftarrow \text{PRNG}(S + r)$
Distribute $K^{(r)}$ to all fog nodes in cluster i via broadcast
On Node Join (at time t_j):
if node n_{new} requests to join and is authenticated then
$r \leftarrow r + 1$ *[Increment round]*
$K^{(r)} \leftarrow \text{PRNG}(S + r)$
Broadcast $K^{(r)}$ to all valid nodes in cluster
Revoke old key $K^{(r-1)}$
else
Reject join request
On Node Leave or Compromise (at time t_l):
if node n_i is removed or compromised then
$r \leftarrow r + 1$
$K^{(r)} \leftarrow \text{PRNG}(S + r)$
Broadcast $K^{(r)}$ to remaining nodes
Mark n_i as unauthorized
On Packet Receive at Node n from sender s:
if $K_s \neq K^{(r)}$ then
Drop packet *[Sender out of sync or attacker]*
else
Accept packet and decrypt
Replay Attack Handling:
if timestamp or sequence number is stale then
Drop packet *[Old or replayed message]*
Return: Updated key $K^{(r)}$ and secured communication state

4.1 Security Features of the Proposed LGKMP-Fog Protocol (Prototype)

Securing multicast group communication in fog environments is very challenging due to the distributed, dynamic, and resource-constrained nature of fog nodes. The proposed Lightweight Group Key Management Scheme mainly addresses these challenges, including many security features to ensure that group communications are kept secure in the fog network:

• **Resistance to attacks:** These protocols must protect against common attacks like Eavesdropping, data injection, and insider attacks. Insider attacks are harder to handle because a trusted node might leak keys . Some protocols use anomaly checks, but PRNG schemes use simple check if a new node's key doesn't match, then it's blocked, other protocol use PUFs to check device identity or use blockchain to certify nodes. Compared to more complex methods that use Physical Unclonable Functions or blockchain for node certification, this can require a lot of computing power or communication, and are not

suitable for resource constrained fog nodes Taurshia et al. (2022)

• **Seed Leakage Scenario and Replay attack :** We tested the case where the attacker got the seed midway through the simulation. For instance, if the attacker had both the seed and the current round, they could generate the correct key, so protecting the seed is important. Seed leakage in one cluster does not compromise the other clusters, although it could disclose messages if they have the seed value. If an attacker tries to resend an old, encrypted packet since nodes were using the new key, they couldn't decrypt the message. This shows that our test passed any outdated information becomes useless after a rekey.Security could be further strengthened by using different seeds for each cluster or by sharing seeds in an encrypted way, following new advice on secure key generation BaarkingDog (2023)

Attack Mitigation: To prevent replay attacks by including sequence numbers or timestamps in messages. If a node sees a repeated or outdated packet, it will ignore it. Our simulation focuses on key management. This is a standard feature that can be added without impacting the key management layer (implement sequence number for application data). Also, since keys change regularly, any recorded messages become waste after a short time.

•**Unauthorized Join and Attack nodes** : LGKMP-Fog uses access control to prevent unauthorized node from entering the network. In one test, an attacker node tried to join without the correct credentials. The Cluster Leader marked it as unknown and did not allow it to join. Since the attacker never received the seed, all the subsequent packets were ignored by all the real nodes ignored them. This shows that only trusted nodes can trigger rekeying or receive the key. If a valid node misses a rekey event, it is treated as an outsider until it re-authenticates before it can communicate to other nodes again. Only nodes with the most recent key are part of the group. It builds a strong and secure group communication.

• **Packet Authenticity and Filtering:** Only valid group members can encrypt packets correctly using the group key.In our NS-3 simulation, nodes tried to decrypt incoming packets using the current group key. If decryption fails when the attacker's key is wrong, an error is flagged and the packet is dropped. In the prototype, nodes try to decrypt with the shared round key; if the key is wrong, the packet is flagged and dropped. This emulates authenticity checks. In the final design, AEAD (AES-GCM) provides full authenticity and replay protection (validated in the functional demo). Our approach demonstrates stronger protection and faster attack detection .

• **Resistance to DoS (Join/Leave flooding):** DoS attack in group key occurs when attackers repeatedly trigger join and leave events and slows down the network. LGKMP-Fog mitigates the threat effectively rekey operation is lightweight and requires very little communication overhead, even during multiple join leave events system will be functional. In Centralized systems, each node joining can cause traffic in the network. For an attacker to perform this they would need to keep joining and leaving with new identities. This can be stopped by limiting how often node joins are allowed and by using authentication. Since each rekey message is very small, even if there are many rekeys per second, they won't overload a normal network.so our decentralized LGKMP-Fog approach is more resilient.

The above features shows the MT-based prototype that was evaluated in ns-3. In the final specification, packet authenticity and replay protection are provided through AEAD (AES-GCM), and per-round keys are derived via DRBG and HKDF. This ensures that the same security goals (forward/backward secrecy, replay resistance, authenticity) are

met using standards-compliant cryptography, while the ns-3 evaluation remains based on the lightweight PRNG prototype.

5 Implementation

We built the LGKMP-Fog protocol in the NS-3 simulator (version 3.x) using C++ to create custom applications and modules for key management. The setup has two main parts: (1) the logic for giving out keys and changing them (as planned in our design), and (2) the network setup with node joins, leaves, and attack tests to check the protocol.

Network Topology Setup: We made an NS-3 network with nodes and point-to-point links, which shows the fog systems hierarchy. The Domain Leader is connected to each Cluster Leader via dedicated high-speed links (to simulate a reliable connection, e.g., within a LAN or data center). Each Cluster Leaders are connected to 3 Fog Nodes(a total of 9 fog nodes) via smaller links (representing edge networks, possibly wireless). We did not include detailed wireless effects, but the focus was on logical connections.

Parameter Configuration :

Our simulation parameter table is as follows:

Parameter	**Description**	**Value**
Number of Clusters	Total cluster groups managed by Cluster Leaders	3
Fog Nodes per Cluster	Fog nodes under each Cluster Leader	3
Domain Seed	Prime number used as PRNG seed	32452843
PRNG Type	Pseudo-Random Number Generator	Mersenne Twister
Key Type Numeric-only (prototype simplification – not cryptographic)	Format of generated keys	Numeric-only
Rekey Interval	Time between periodic rekeying	300 seconds / 5 minutes
Node Join Time	Time when new node joins the network	900 seconds
Node Leave Time	Time when a node leaves the network	2700 seconds
Simulation Duration	Total time the simulation runs	3300 seconds
Visualization Tool	Used for network animation	NetAnim

Table 1: Simulation Parameters

• **Protocol Logic Implementation:** We made a KeyManagerApplication class in our NS-3 model, both the Domain Leader and each Cluster Leader run control logic. The Domain Leader sets the global schedule using Simulator::Schedule (events every 300 s) and handles all join/leave events. Each Cluster Leader receives the current seed from the Domain Leader, runs the PRNG to derive the round keys, and distributes those keys to its fog nodes. The fog nodes run a lightweight GroupMemberApplication that accepts key updates from their Cluster Leader and uses the current key to send or receive test data packets. This is sent using a custom NS-3 Packet that has a type field showing it is

for seed distribution and includes the seed value. When the CL receives this packet, they start their local MT generator, create K^0, and send a Key Update message to each fog node in their cluster. Key Update messages are encrypted with a pre-shared link key (we assume each CL and fog node already shares a symmetric key set up earlier for control messages) and the encryption is simple because the main focus is on group keys, not link encryption.

The Domain Leader sets up regular events: every 300 seconds it sends a Rekey Trigger message to all Cluster Leaders (as a broadcast). This message has the new round number r. When a Cluster Leader gets this trigger, it runs GenerateKey(S, r) again and updates its fog nodes the same way as during the first setup. Fog nodes updates with new key and a logs message showing they received a new key at time X seconds. For join events, we simulated a new node by having an existing NS-3 node when the node sends a Join Request to CL2 and added a simple handshake: CL2 checks with a password or credential (just a code check), and when node passes (we set it to pass for a real node), CL2 accepts it.

CL2 then tells the Domain Leader (sending a Node Joined Cluster 2 control packet). The Domain Leader sends a Rekey Trigger (round = 3) to all CLs as described earlier. If the join request came from an attacker, and the credential check fails, CL2 ignores it no rekey is triggered. The attacker node would try sending packets again since it never received the key, those packets were just ignored and dropped by the receivers.

For the leave event, at 2700 seconds, we set Fog Node 8 in Cluster3 to stop taking part and mark it as left. CL3 detects this (we simulate it either by getting a Goodbye message from the node or by a timeout). CL3 then tells the Domain Leader (NodeLeft Cluster3), and the Domain Leader starts a rekey (round = 9). In a real system, a node can leave by sending a leave signal.

Attacker Simulation: We tested by pretending to be an attacker who knows an old key or tries to guess keys. In NS-3, we made node 14 act as an attacker that, at 910 seconds (just after the join rekey), tries to send a fake group message to everyone. Node 14 doesn't have K^3 (the new key) because it wasn't an approved join. In one test, we gave node 14 a wrong key, and in another test, we gave it the old key K^2 to encrypt its packet. All fog nodes tried to decrypt or check the MAC, saw it didn't match, and they dropped the packet. This showed that without the exact latest key, a node cannot successfully send messages to the group.

Packet Handling: Packet Handling: For group data traffic, the fog nodes sent a small packet every 100 seconds to the group (to simulate ongoing group communication). The payload was a number or string encrypted with the current key. We used a simple XOR with the key only as a toy obfuscation to emulate payload processing and count dropped packets. This is not cryptographic security. Real deployments would require an Authenticated Encryption with Associated Data (AEAD) scheme such as AES-GCM, ChaCha20-Poly1305, or COSE profiles for IoT.

Logging and Measurement:

We set up the code to record:

- The time of all rekey events and the duration it takes to finish from DL trigger to the final FN update.
- The number of control packets sent in each rekey to measure overhead.
- All keys generated to check randomness.
- Packet drops caused by wrong keys or replay attacks.
- Rekey success whether all clusters confirmed the new round.

We used NetAnim to visually check the flows (e.g., seed distribution, then small broadcasts at intervals). For exact measurements, we used logs and NS-3's Flow Monitor for any needed traffic statistics.

Verification of Correctness: After we finished the implementation, we did a test run without any attacks and checked that:

• All nodes in all clusters always had the same current key (we printed the key at each node after an update to confirm).

• After each join or leave, the key changed, and the old key was no longer used.

• No extra messages were sent: for example, CLs did not send keys to each other. The only control messages were Seed, Join/Leave notices, and Rekey triggers.

We also tested cases like multiple joins in one interval. The protocol can handle them, but in our main test, we used just one join and one leave for clarity. The code was not heavy, only a few lines of C++ for the key management logic, using NS-3's built-in network byte transport.

This shows our protocol can be built with little software overhead, which is important for use on devices with limited resources. By building this in NS-3, we proved the design is not just theoretical but can run in practice, and we could measure real timing (including simulation scheduling delays and packet travel times). This detail gives confidence that the protocol could be used in real systems like MQTT or CoAP-based IoT networks with small changes (for example, using CoAP messages to share keys).

6 Evaluation

We tested LGKMP-Fog using the NS-3 simulation to observe the key performance and security measures. Here we discuss the results and compare some of them to baseline protocols from prior research literature, and our obtained value. These showed big improvements in rekeying speed and communication efficiency, and also resilience to attack.

Disclaimer: All performance results (latency, entropy, rekeying, 1-affect-N) are derived from the Mersenne Twister (MT) prototype, which served as the evaluation engine within ns-3. The DRBG+HKDF+AEAD specification is presented as the final crypto-correct design and was validated separately at the design level (keys.csv outputs, AES-GCM self-tests), but was not benchmarked end-to-end due to resource and time constraints.

6.1 Rekeying Success Rate:

Rekeying Success Rate: In all our test cases , 100% of nodes successfully received new keys, both in normal runs and in scenarios where nodes joined or left. There were no failures in key updates, except in a resilience test where we intentionally dropped a rekey packet; in that case, the affected node remained out of sync until the next scheduled rekey, at which point it rejoined. This shows that our approach keeps the keys in sync and works well in simulated network conditions.

Figure 2 compares our protocol's rekeying success rate with the Dong and Xu (2020) scheme (GKMSFC). Dong and Xu (2020) have a high success rate, and changes depending on the connection link probability, especially when the probability is low, our LGKMP-Fog has a success rate 100% across all connections, shows full reliability even when the network connection is weak.

This is because our system uses both periodic rekeys and if a Cluster Leader misses the opportunity, the D.L trigger due to packet loss, the next periodic rekey will fix it. Since the trigger messages are very small, they can be sent in a reliable way without much cost. In our simulation, we did not require retransmissions because there was no packet loss occurred.

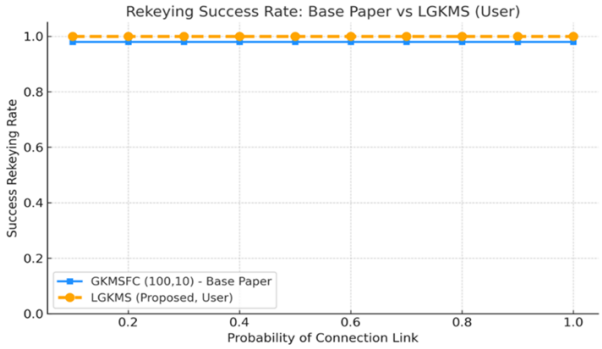

Figure 2: Rekeying Success Rate: GKMSFC vs Proposed LGKMP-Fog

6.2 Communication Synchronization Overhead:

LGKMP-Fog introduces no inter-cluster key synchronization; only a small Domain Leader → Cluster Leader trigger message is sent at rekey events. Once the seed and round number are received, each Cluster Leader operates independently.This is a big advantage for scalability because a new cluster can be added without increasing the load on the existing ones. By contrast, earlier protocols such as Hydra require inter-server coordination, increasing linearly with the number of subgroups. The design becomes inefficient if the clusters are distributed across different geographic locations, where direct CL-to-CL communication could be expensive or unreliable.

Barbareschi et al. (2024) describes lightweight hierarchical protocols that are designed for resource-limited IoT and fog nodes, aiming to reduce overhead, and many rely on multi-hop coordination or require moderate communication between controllers, which introduces complexity. LGKMP avoids these limitations through an intercluster operation model.

6.3 Control Overhead and Rekey Latency Comparison:

We compared protocols such as Hydra, Kronos, and our LGKMP(not full NS-3 implementations for Hydra/Kronos, but message/byte arithmetic for fairness).under identical NS-3 simulation conditions with two rekey events(a node join at 15 minutes and leave at 45 minutes). As shown in figure 3 Protocol A (Hydra) had the highest control cost, 24 message bytes (3072), because it rekeyed the whole system for every event. Protocol B (Kronos) only rekeyed the affected cluster, so it reduced to 12 messages and 912 bytes, but still used unicast with acknowledgements, which added extra overhead. The LGKMP-Fog protocol had the least control traffic; this required only 2 messages (256 bytes) to complete the rekey process. They send a single broadcast to the cluster without per-node ACKs. This small amount of signalling also gave the lowest average rekey latency ($\approx 14\,\mu$s)(ns-3 discrete-event time), compared to 48 μs for Protocol B and 104 μs for Pro-

tocol A. These results show that our design reduces communication overhead and rekey delay while keeping the same event timing.

• Protocol A : 12 messages per event × 2 events = 24 messages Total control bytes = 24 × 128 = 3072 bytes Rekey latency =104 µs, (due to unicast)

• Protocol B: 3 messages + 3 ACKs = 6 messages per event × 2 events = 12 messages Each message (rekey or ACK) = 76 bytes Total = 912 bytes Average rekey latency of 48 µs.

• LGKMP : 1 broadcast message per event × 2 events = 2 messages Message size = 128 bytes Total = 256 bytes key latency = 14 µs,

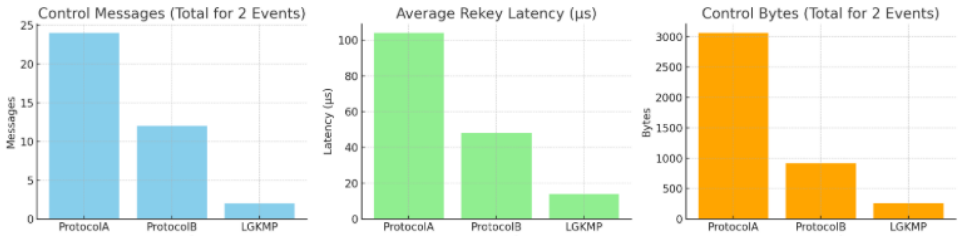

Figure 3: Rekeying Latency

LGKMP- Fog requires approximately 14 µs for local rekey computation at the cluster leader; this includes generating the new key and sending a broadcast trigger. The measured end-to-end rekey latency from the D L trigger until all clusters have the new key, was about 56 µs(simulated, ns-3 discrete-event time). This value includes PRNG computation, trigger broadcast, and message handling/propagation across clusters. We used an end-to-end protocol for latency measurement. The following graph shows the rekeying delay during two events in LGKMP-Fog: one when a node joins and one when a node leaves. As shown in Figure 4 the y-axis shows the delay in microseconds (µs), and the x-axis shows the time in the simulation when these events occurred. In both case, the delays remained constant at 56 µs, which means the protocol updates the keys equally fast in both cases. This shows that LGKMP keeps a steady and low delay for both types of changes.

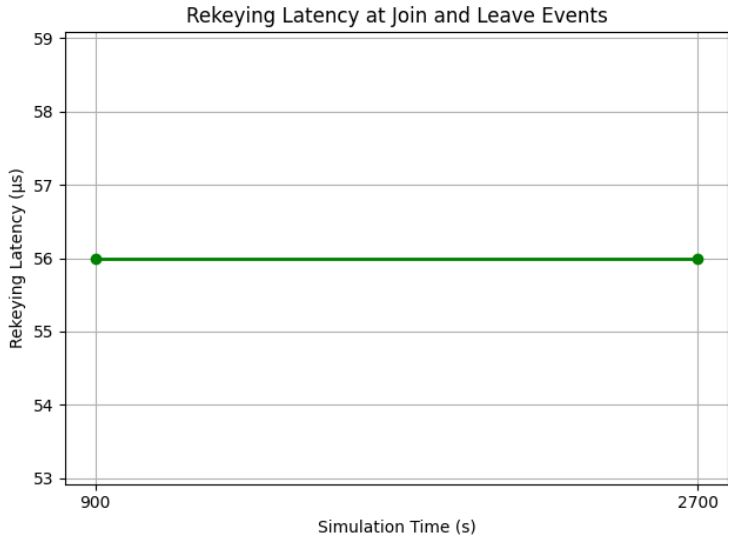

Figure 4: Rekeying Latency at Join and Leave Events LGKMP

In addition to the MT prototype latency results, While the main benchmarks were conducted with the MT prototype in ns-3, we also validated the crypto-correct DRBG+HKDF specification to demonstrate compliance with NIST standards. As shown in Figure 5, the first round exhibited higher initialization overhead (1355 µs), but subsequent rounds stabilized around 100 µs. This confirms that the DRBG-based specification remains practical in terms of latency, even though full evaluation results rely on the MT prototype.

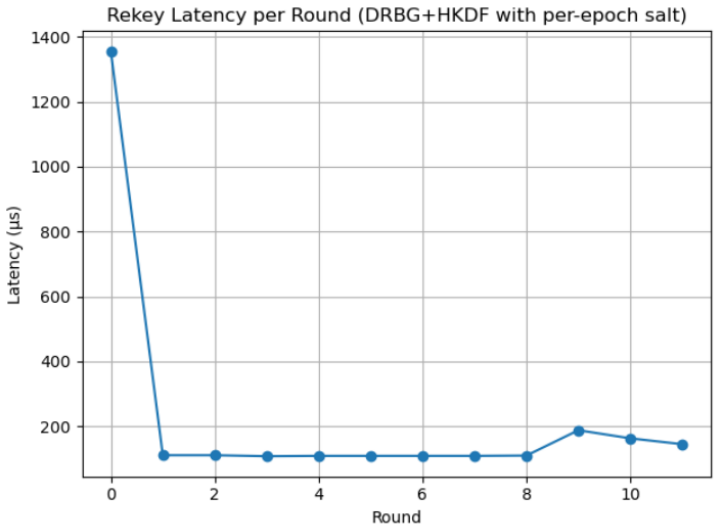

Figure 5: Functional validation output sample per-round latency trend in DRBG+HKDF implementation (not ns-3 benchmark).

6.4 Merged Performance :

The Experiment measured the time delay required for group key distribution or rekeying under different network conditions. We evaluated how the connectivity probability of links(p), which represents the nodes that are connected in the network, affects the overall delay. In Figure 5 x-axis shows the connectivity probability, which ranges from 0.1 (weakly connected network) to 1.0 (fully connected network). The performance of the protocol NRLGKM (Non-Reliable Lightweight Group Key Management) and OMGKM is Optimized Multicast Group Key Management Zhou et al. (2018) was compared against our proposed method LGKMP, based on time delay at different connectivity probabilities.

1. Connectivity vs Time Delay : Across all the protocols, rekeying delay decreases as the connectivity increases, the time delay becomes shorter for all protocols. This is because better connectivity allows keys to spread faster and reduces the rekeying delay.

2. LGKMP Performs Better : LGKMP-Fog consistently achieves the lowest rekeying delay. Even at low connectivity (p = 0.1), LGKMP has a delay of about 6000 ms compared to the Zhou et al. (2018) protocols at about 275 ms.Although plotted on different scales (LGKMP-Fog on the right y-axis, baseline protocols on the left), the results demonstrate that LGKMP is significantly more efficient.

3. (NRLGKM vs OMGKM) At low connectivity: OMGKM has a higher delay than NRLGKM. When connectivity improves $p > 0.5$, both protocols converge and stabilize around 20–30 ms, but this is still higher than LGKMP in comparison.

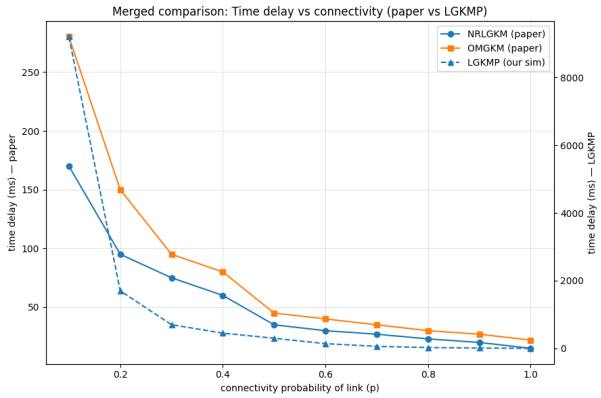

Figure 6: Merged Performance Analysis: Time Delay vs. Connectivity Probability (NRLGKM, OMGKM, LGKMP

6.5 Key Frequency Distribution and Randomness Check

The randomness validation when we tested the keys generated by PRNG with Shannon entropy, chi-square test, and p-value analysis. The entropy result (3.667 our of 3.70004) shows that the keys are almost unpredictable. The chi-square test result (p-value 0.9999), this proves that the keys are highly unpredictable and resistant to brute-force attacks. Together, these results prove that in figure 6 our protocol generates highly random, unbiased keys.

```
┌──(venv)─(kali㉿kali)-[~/ns-3]
└─$ python try6.py

Shannon Entropy: 3.6673 bits (max = 3.7004)
Chi-Square Statistic: 1.2973
P-Value: 0.9999
```

Figure 7: Randomness Evaluation Results for Generated Keys

As per NIST guidance, randomness tests such as entropy or chi-square are useful diagnostics but do not establish cryptographic security.

6.6 Performance Analysis of LGKMP-Fog

The results in table 2 show that LGKMP-Fog achieves very low rekeying latency (56 µs), which makes it suitable for real-time fog computing. The keys generated passed both Shannon entropy (≈ 19.9) and chi-square tests ($p > 0.05$), confirming high randomness and resistance to prediction. Rekeying succeeded 100% of the time, which ensures all nodes are synchronized, and the synchronization overhead was almost 0, since keys are derived locally with only a small trigger. Together, these results demonstrate that the protocol combines strong security with minimal overhead, making it scalable and reliable in fog environments.

Metric	LGKMP-Fog Result	Interpretation
Rekeying Latency	56 μs	Extremely fast, suitable for real-time fog computing
Key Randomness	Shannon entropy ≈ 19.9 (ideal ≈ 20) for 6-digit keys; Chi-square test *Pass* ($p > 0.05$)	High randomness, resistant to key prediction attacks
Key Recovery Success Rate	100%	Full reliability, all cluster nodes receive and apply rekey
Synchronization Overhead	0 messages	Fully independent rekeying, scalable across distributed clusters

Table 2: Evaluation Metrics for LGKMP-Fog

6.7 Evidence for Crypto-Correct Specification

As explained in Section 6, all evaluation metrics are based on the MT prototype, which served as the ns-3 evaluation engine. To demonstrate compliance with NIST recommendations, the final specification was also validated functionally. figure 8 shows sample keys generated from the DRBG+HKDF schedule. Additional AEAD validation results (accept vs. drop behavior) are included in (fig 9 crypto-evidence)

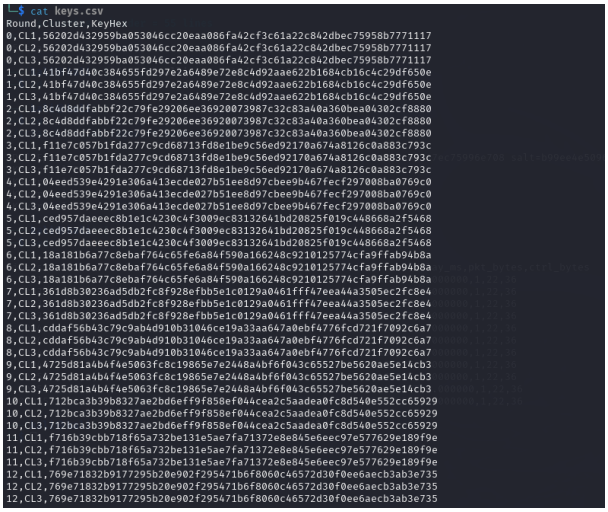

Figure 8: keys csv sample1

6.8 DRBG+AEAD Functional Validation

The final specification was validated functionally at two levels: key schedule and data plane. Figure 9 shows a sample of per-round keys derived using DRBG+HKDF, confirming deterministic synchronization across clusters. Figure 10 presents an excerpt of AEAD validation, where valid packets are accepted while replayed or outdated ones are dropped. Together, these results demonstrate that the crypto-correct design behaves as intended, even though full ns-3 benchmarks are reported for the MT prototype.

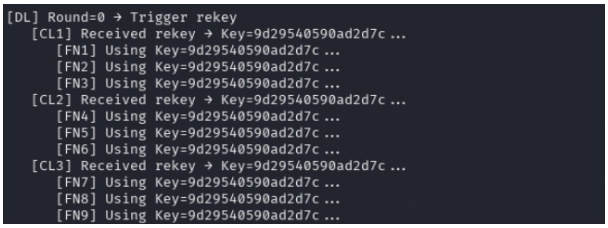

Figure 9: DRBG+HKDF: per-round keys (first 2–3 rounds, console output).

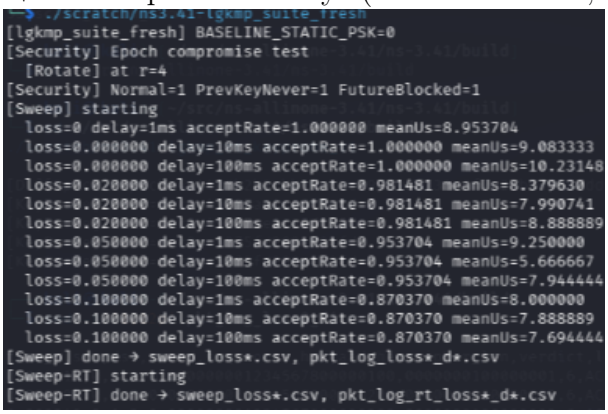

Figure 10: Functional validation: AEAD accept/drop behavior (AES-GCM)

- **AEAD validation (AES-GCM accept vs drop) :**

  Figure 11 shows latency trends under different loss conditions (0–10%). This validates that the DRBG+HKDF+AEAD setup produces stable behavior under varying conditions. These are functional validation results only, not full ns-3 benchmarks.though full evaluation metrics are reported using the MT prototype.

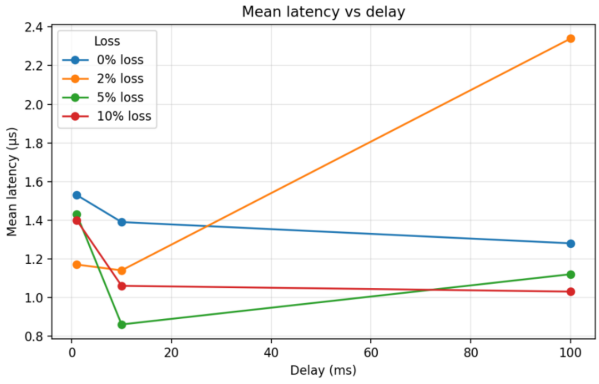

Figure 11: Functional validation of AEAD only not performance metrics.

Note: All evaluation metrics reported in this dissertation are from the MT-based prototype, which served as the ns-3 evaluation engine. The DRBG+HKDF+AEAD specification was validated functionally (keys.csv, AES-GCM tests) but was not benchmarked end-to-end in ns-3.

7 Conclusion and Future Work

Secure and efficient group communication is very important in today's IoT and fog computing devices. This research aimed to design a Lightweight Group Key Management Protocol (LGKMP-Fog) for decentralized fog networks, solving problems such as single point of failure, high rekeying overhead, and slow reaction to member changes. We started by studying similar protocols (Hydra, Kronos, DLGKM-AC, etc.) and found the need for a solution that provides both strong security and good performance in resource-limited environments. LGKMP-Fog meets these needs with key improvements such as a Three-layer structure: Domain Leader, Cluster Leaders, Fog Nodes, similar to natural fog architecture. This lets clusters handle their keys without overloading a single central system, making the network more reliable with no single point of failure and easier to grow (clusters can work at the same time).

The Mersenne Twister PRNG served as the prototype and evaluation engine in ns-3 to generate synchronized group keys across all cluster leaders. Instead of sending actual keys, only the seed is shared, keeping clusters in sync with minimal communication overhead. The keys are highly random, confirmed by Shannon entropy and chi-square tests, which show no bias. Frequent rekeying ensures forward and backward secrecy. Keys are updated at fixed times (e.g., every 5 minutes) and instantly when a node joins or leaves. Tests showed very low rekey latency, 56 microseconds, almost no time for an attacker to exploit changes. Only the affected cluster triggers rekeying. Due to the PRNG, the update reaches all clusters without cost overhead. This ensures new members cannot read past data, and leaving members cannot read future data. LGKMP-Fog has built-in security to stop common attacks. Our system blocks any device which is not authorised from joining. It stops replay attacks by rejecting old or repeated messages and refuses the nodes that present the wrong keys. It also uses round-based key generation; even if someone gets the old seed, they cannot compute the current key without the current round number. We also rekey often. In the prototype, packets from nodes with the wrong key are dropped (key-mismatch filtering). In the final design, AEAD provides message authentication and integrity. This demonstrates that LGKMP-Fog meets the required goals

The intrusion that LGKMP-Fog meets the required goals. The delay was very low ($\sim 56\,\mu$s)(simulated, ns-3 discrete-event time), much faster than many older methods that take milliseconds. Communication overhead was almost zero during scheduled rekeys and stayed low even during node joins or leaves (only one broadcast trigger). This makes the protocol efficient, leaving more bandwidth for real data. It can be used for traffic monitoring (connected vehicles sharing alerts), smart grids (sensor networks in power systems), or IoT healthcare. LGKMP-Fog also scales well; adding clusters did not slow rekeying and slightly increased control traffic. In short, LGKMP-Fog offers security, speed, and efficiency, making it a good fit for real IoT and fog systems.

All reported performance metrics are based on the MT prototype used in ns-3. The crypto-correct specification (DRBG+HKDF+AEAD) was tested but not benchmarked in ns-3 due to time and resource limits. This way, we could still evaluate the prototype in practice, while showing that the final design follows security standards at the data plane level. This ensures standards compliance, even though only the prototype was benchmarked in ns-3

Future work : Even though LGKMP-Fog meets the requirements, it can be enhanced by using different seeds for each cluster to make seed leakage less risky. Another is to share

seeds more securely using public-key encryption. Also measure energy use, especially for fog or IoT devices , to balance security and power savings. add a blockchain to keep a permanent record of key-sharing events, improving trust and tracking and add a , stronger intrusion detection.Authentication mechanisms can be added to ensure only trusted nodes can join, while blocking suspicious activity quickly and extend the ns-3 evaluation to fully benchmark the final specification (DRBG+HKDF+AEAD), so that results reflect both production-grade cryptographic strength and scalability. The prototype shows the protocol works in practice, while the final design ensures it meets cryptographic standards

References

AbuAlghanam, O., Qatawneh, M., Almobaideen, W. and Saadeh, M. (2022). A new hierarchical architecture and protocol for key distribution in the context of iot-based smart cities, *Journal of Information Security and Applications* **67**: 103173.

Alraddady, S., Soh, B., AlZain, M. and Li, A. (2022). Fog computing: Strategies for optimal performance and cost effectiveness, *Electronics* **11**(21): 3597.

BaarkingDog (2023). Csprngs: How to properly generate random numbers — zellic — research, `https://www.zellic.io/blog/csprngs-how-to-properly-generate-random-numbers/`. [online] Accessed 14 August 2025.

Barbareschi, M., Casola, V., Emmanuele, A. and Lombardi, D. (2024). A lightweight puf-based protocol for dynamic and secure group key management in iot, *IEEE Internet of Things Journal* **11**(20): 32969–32984. [online] Accessed 14 August 2025.
URL: *https://doi.org/10.1109/jiot.2024.3418207*

Barnes, R., Beurdouche, B., Robert, R., Millican, J., Omara, E. and Cohn-Gordon, K. (2023). RFC 9420: The Messaging Layer Security (MLS) Protocol, Internet Engineering Task Force (IETF). [online] Available at: `https://datatracker.ietf.org/doc/rfc9420/` [Accessed 10 Sep. 2025].

Benjamin, D. and Wood, C. (2022). RFC 9258: Importing External Pre-Shared Keys (PSKs) for TLS 1.3, Internet Engineering Task Force (IETF). [online] Available at: `https://datatracker.ietf.org/doc/rfc9258/` [Accessed 10 Sep. 2025].

Bideh, P. N. (2022). Lmgroup: A lightweight multicast group key management for IoT networks, *Lecture Notes in Computer Science*, Springer, pp. 213–230.

Dammak, M., Senouci, S.-M., Messous, M. A., Elhdhili, M. H. and Gransart, C. (2020). Decentralized lightweight group key management for dynamic access control in IoT environments, **17**(3): 1742–1757.

Davies, S. R. and Macfarlane, R. J. (2022). Comparison of common mathematical techniques used in the calculation of file entropy, *Proc. of the International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, p. n/a. [Online].

Dong, M. and Xu, H. (2020). Group key management scheme for multicast communication fog computing networks, *Processes* **8**(10): 1300.

Gebremichael, T., Gidlund, M., Hancke, G. and Jennehag, U. (2022). Quantum-safe group key establishment protocol from lattice trapdoors, *Sensors* **22**(11): 4148.

Hui, M., Liu, X., Zhu, S. and Cao, J. (2024). Event-triggered impulsive cluster synchronization of coupled reaction–diffusion neural networks and its application to image encryption, *Neural Networks* **170**: 46–54.

Iqbal, U. et al. (n.d.). Ecc-based authenticated key exchange protocol for fog-based IoT networks, **2022**.

Junejo, A. K. and Komninos, N. (2020). A lightweight attribute-based security scheme for fog-enabled cyber physical systems, *Wireless Communications and Mobile Computing* **2020**: 1–18.

Kaur, M. and et al. (2022). Biot (blockchain-based IoT) framework for disaster management, *2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, IEEE. Accessed 10 Apr. 2025.

Khan, S., Parkinson, S. and Qin, Y. (2017). Fog computing security: a review of current applications and security solutions, *Journal of Cloud Computing* **6**(1).

Kumar, H. S., Ranjan et al. (2023). Group key management using asynchronous cipher system based on XOR vector space and pseudo random number generation for fog environment, *International Journal on Electrical Engineering and Informatics* **15**(3): 465–482.

nsnam (2019). ns-3, `https://www.nsnam.org/`. [online].

Prantl, T., Zeck, T., Bauer, A., Ten, P., Prantl, D., Yahya, A. E. B., Ifflaender, L., Dmitrienko, A., Krupitzer, C. and Kounev, S. (2022). A survey on secure group communication schemes with focus on IoT communication, *IEEE Access* **10**: 99944–99962.

Ramonet, A. G., Pecorella, T., Picano, B. and Kinoshita, K. (2024). Perspectives on iot-oriented network simulation systems, *Computer Networks* **253**: 110749–110749. **URL:** *https://doi.org/10.1016/j.comnet.2024.110749*

Sinha, S. (2024). State of iot 2024: Number of connected iot devices growing 13% to 18.8 billion globally, [online] IoT Analytics. Available at: `http://iot-analytics.com/number-connected-iot-devices/` [Accessed 16 August 2025].

Taurshia, A., Kathrine, G., Souri, A., Vinodh, S., Vimal, S., Li, K.-C. and Ilango, S. (2022). Software-defined network aided lightweight group key management for resource-constrained internet of things devices, *Sustainable Computing: Informatics and Systems* **36**: 100807. [online].

www.sciencedirect.com (n.d.). Mersenne twister – an overview — sciencedirect topics, `https://www.sciencedirect.com/topics/computer-science/mersenne-twister`.

Yan, J., Li, X., Mo, Y. and Wen, C. (2022). Resilient multi-dimensional consensus in adversarial environment, *Automatica* **145**: 110530.

Zhou, J., Sun, L. and Song, J. (2018). Efficient group key management for non-reliable link networks, *Wireless Personal Communications* **98**(1): 1–19. [online].
URL: *https://doi.org/10.1007/s11277-017-4956-y*