Fortified Control-Plane Encapsulation with Session-Key Derivation for Secure IP Mesh Routing

George Amponis
K3Y Ltd. and Dept. of Informatics
Democritus University of Thrace
Sofia, Bulgaria and Kavala, Greece
gamponis@{k3y.bg, cs.duth.gr}

Panagiotis Radoglou-Grammatikis
K3Y Ltd. and Dept. of Elec. & Comp. Eng.
University of Western Macedonia
Sofia, Bulgaria and Kozani, Greece
pradoglou@{k3y.bg, uowm.gr}

Thomas Lagkas
Dept. of Informatics
Democritus University of Thrace
Kavala, Greece
tlagkas@cs.duth.gr

Vasileios Argyriou
Dept. of Networks & Digital Media
Kingston University
London, UK
vasileios.argyriou@kingston.ac.uk

Antonios Sarigiannidis
K3Y Ltd.
Sofia, Bulgaria
asarigia@k3y.bg

Natasa Kazakli
MetaMind Innovations P.C.
Thessaloniki, Greece
nkazakli@metamind.gr

Thomas Boufikos
K3Y Ltd.
Sofia, Bulgaria
tboufikos@k3y.bg

Panagiotis Sarigiannidis
Dept. of Electrical and Computer Eng.
University of Western Macedonia
Kozani, Greece
psarigiannidis@uowm.gr

***Abstract*—Ad hoc routing protocols, widely used in mesh networks, lack inherent control plane security, making them vulnerable to classical attacks and future quantum threats. We present a security enhancement for the Optimized Link State Routing Protocol (OLSR) integrating post-quantum cryptography. Our approach embeds a Kyber512 key-encapsulation handshake within neighbor discovery (HELLO messages) to establish secure sessions. Subsequent HELLO and Topology Control (TC) messages are protected using ChaCha20-Poly1305 authenticated encryption (AEAD). This ensures neighbor authenticity, message confidentiality, and integrity against both classical and quantum adversaries without altering core OLSR routing logic. We detail the mechanism, including TLV-based packet extensions and cryptographic state management, using standard libraries (`liboqs`, `OpenSSL`). This provides a practical pathway towards quantum-resilient mesh networking, with performance evaluated through parameters like handshake latency and control plane overhead. Future work can explore protocol-agnostic abstractions.**

***Index Terms*—OLSR, Mesh Networking, Post-Quantum Cryptography, Kyber, Network Security, Control Plane Security,**

I. Introduction

Wireless mesh networks provide flexible communication for applications ranging from community networks to tactical systems, often employing ad hoc protocols like Optimized Link State Routing (OLSR) [1] for its proactive routing and fast path discovery. However, standard OLSR [1] exchanges unencrypted control messages (`HELLO` for neighbor discovery, `TC` for topology dissemination). This exposes the control plane to significant threats: passive eavesdropping reveals topology, while active attackers can disrupt routing via message forgery or replay attacks [1]. Traditional security overlays face key management challenges in dynamic ad-hoc settings and typically rely on classical public-key cryptography (RSA, ECC) for key exchange. These classical foundations are vulnerable to future quantum computers running Shor's algorithm [2]. The ”store now, decrypt later” strategy poses a serious risk, potentially allowing retroactive decryption of captured traffic, compromising topology and session keys. Therefore, securing OLSR requires both classical robustness and quantum-resistant forward secrecy. This paper proposes a security enhancement for OLSR by integrating Post-Quantum Cryptography (PQC). We embed a Kyber512 [3] Key Encapsulation Mechanism (KEM) handshake within OLSR `HELLO` messages using Type-Length-Value (TLV) extensions to establish shared secrets. Subsequent control messages (`HELLO`, `TC`) are protected using a derived symmetric key and the ChaCha20-Poly1305 [4] Authenticated Encryption with Associated Data (AEAD) scheme. Our contributions include the integrated PQC KEM handshake via TLVs, derived key usage for AEAD protection, detailed protocol extensions and operational sequences, and an implementation strategy leveraging standard cryptographic libraries (`liboqs` [5], `OpenSSL` [6]). This work provides a practical design for achieving quantum-resilient control plane security in Layer 3 mesh protocols.

XTRUST-6G is co-funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or Smart Networks and Services Joint Undertaking. Neither the European Union nor the granting authority can be held responsible for them.” ”This work has received funding from the Swiss State Secretariat for Education, Research and Innovation (SERI)

II. Background and Related Work

A. Optimized Link State Routing (OLSR)

OLSR [1], a proactive link-state protocol for Mobile Ad-hoc Networks (MANETs), uses periodic `HELLO` messages for neighbor discovery and link sensing. It employs Multipoint Relays (MPRs), a subset of neighbors selected to forward `TC` messages, reducing broadcast overhead. `TC` messages declare links to MPR selectors, enabling nodes to construct a network topology map for shortest path computation.

B. The Quantum Threat and Post-Quantum Cryptography

Classical public-key algorithms like RSA and ECC are vulnerable to Shor's algorithm [2] executed on future quantum computers. Post-Quantum Cryptography (PQC) aims to provide algorithms secure against both classical and quantum attackers, based on different mathematical foundations (lattices, codes, hashes, etc.). The NIST PQC standardization process [7] has selected algorithms like CRYSTALS-Kyber [3], a lattice-based KEM, recognized for its security and performance balance.

C. Existing MANET Security and Related Work

Research on secure MANET routing includes studies about OLSR variants using conventional cryptography as well as external key management solutions but such approaches lead to performance limitations including quantum protection challenges. Wireless communication networks require PQC implementations according to research that focuses on group messaging [8] and V2X and DTNs and specialized networks like underwater networks [9] and wireless systems [10]. Research is currently focused on lightweight KEM proposals and performance evaluation. The use of general PQC MANET security solutions together with Quantum Key Distribution (QKD) as an alternative or supplement has been studied [11]–[16] but implementing standardized PQC KEMs such as Kyber with AEAD protection directly into routing protocol control planes through transparent extensions receives less attention. Our research demonstrates the practical implementation of QKD with established libraries [5], [6] to enhance the OLSR protocol through its current framework.

III. Proposed PQC-Enhanced OLSR Mechanism

To guarantee confidentiality and integrity of OLSR routing (meta)data against both classical and quantum adversaries taht may want to exfiltrate network topology information or payload data, we integrate a Kyber512 key-encapsulation mechanism and ChaCha20-Poly1305 AEAD encryption directly into the OLSR packet flow, without altering its core route-computation or MPR selection algorithms.

A. System Architecture

Figure **??** illustrates the architectural placement of the proposed security enhancements within an OLSR node. The `PQC-Enhanced OLSR Control Plane` acts as a security shim between the standard IP layer and the core OLSR processing logic. All incoming OLSR control packets are intercepted by this layer. If the packet contains PQC handshake TLVs, the layer performs the necessary key exchange operations using `liboqs`. If the packet contains an AEAD TLV, it uses the appropriate key and `OpenSSL` to decrypt and authenticate the payload. Only successfully authenticated and decrypted payloads are passed to the `OLSR Core Logic`. Conversely, when the core logic generates a HELLO or TC payload to be sent, the security layer encrypts and authenticates it using the relevant session key and passes the AEAD TLV-packaged packet to the IP layer for transmission.

B. Cryptographic Primitives and Formalism

Before detailing the protocol phases, we formally define the cryptographic building blocks. Let $\mathcal{PK}$ be the public key space, $\mathcal{SK}$ be the secret key space, $\mathcal{CT}$ be the ciphertext space, and $\mathcal{SS}$ be the shared secret space (typically $\{0,1\}^{256}$ for 256-bit keys) for the chosen PQC KEM (Kyber512).

- **KEM Key Generation:** A probabilistic algorithm KEM.KeyGen() outputs a public/secret key pair (pk, sk), where $pk \in \mathcal{PK}$ and $sk \in \mathcal{SK}$.

$$(pk, sk) \leftarrow \text{KEM.KeyGen}()$$

- **KEM Encapsulation:** A probabilistic algorithm KEM.Encaps(pk) takes $pk \in \mathcal{PK}$ and outputs a ciphertext $ct \in \mathcal{CT}$ and a shared secret $ss \in \mathcal{SS}$.

$$(ct, ss) \leftarrow \text{KEM.Encaps}(pk)$$

- **KEM Decapsulation:** A deterministic algorithm KEM.Decaps(sk, ct) takes $sk \in \mathcal{SK}$ and $ct \in \mathcal{CT}$, and outputs $ss' \in \mathcal{SS}$. For correctly generated pairs, $ss' = ss$.

$$ss' \leftarrow \text{KEM.Decaps}(sk, ct)$$

- **Key Derivation Function (HKDF):** Let HKDF($salt, ikm, info, L$) be the HMAC-based Key Derivation Function (extract-then-expand) as defined in RFC 5869 [17]. We derive the session key K_{session} of length $L = 32$ bytes as:

$$K_{\text{session}} \leftarrow \text{HKDF}(\text{null}, ss, \texttt{"PQC-OLSR"} \| ID_A \| ID_B, 32)$$

where ss is the KEM shared secret, null indicates no salt is used initially, and $info$ includes a context string and unique node identifiers ID_A, ID_B (e.g., main IP addresses, consistently ordered).

- **AEAD Encryption/Decryption:** Let AEAD.Encrypt(K, N, P, A) take a key K, nonce N, plaintext P, and associated data A, outputting ciphertext C and tag T. Let AEAD.Decrypt(K, N, C, A, T) take the key, nonce, ciphertext, associated data, and authentication tag T, outputting plaintext P if the tag verifies, or $\perp$ (failure) otherwise. We use ChaCha20-Poly1305 where the tag T is typically appended to C.

The mechanism operates in two phases:

C. Phase 1: PQC Key Establishment via HELLO Handshake

Upon first encountering a potential neighbor (or periodically for rekeying), nodes initiate a key establishment handshake piggybacked onto standard `HELLO` messages using Type-Length-Value (TLV) extensions (see Figure 1). We utilize the **Kyber512** KEM [3]. The detailed message flow is shown in Figure 3. The handshake involves exchanging KEM public keys (TLV `0x0F`) and ciphertexts (TLV `0x10`) to allow both nodes to derive an identical shared secret (`ss`) via KEM encapsulation and decapsulation using `liboqs` [5]. This shared secret is then processed through HKDF [17] to produce a symmetric session key ($K_{session}$).

D. Phase 2: AEAD Protection for Steady-State Control Traffic

Once the shared `SessionKey_AB` (derived as $K_{session}$) is established, all subsequent `HELLO` and `TC` messages exchanged *between Node A and Node B* are protected using ChaCha20-Poly1305 [4].

1) Before sending, the node generates a unique Nonce N. The original OLSR payload P (e.g., HELLO neighbor list or TC MPR selector list) is encrypted using AEAD.Encrypt($K_{session}, N, P, A$), where A is typically empty associated data, via OpenSSL's EVP interface [6]. This yields ciphertext C and tag T.
2) The original payload P is replaced by TLV `0x11` containing (N, C, T) (see Figures 1 and 2).
3) The receiving node extracts N, C, T from TLV `0x11`. It computes $P' \leftarrow$ AEAD.Decrypt($K_{session}, N, C, A, T$) using OpenSSL [6].
4) If the result $P' \neq \perp$ (authentication successful), the decrypted payload P' is passed to the standard OLSR processing logic. If $P' = \perp$, the packet is discarded, and a failure may be logged or counted towards blacklisting.

E. Enhanced Packet Structures

To accommodate the cryptographic data, we extend the OLSR message format using TLVs. OLSRv2 natively supports TLVs; for OLSRv1, they can be added after the main message content. Figures 1 and 2 show the proposed structures.

Key points regarding the packet structures:

- TLVs `0x0F` (Kyber Public Key) and `0x10` (Kyber Ciphertext) are only present during the initial handshake phase within `HELLO` messages.
- TLV `0x11` (AEAD Payload) is used in both `HELLO` and `TC` messages after the handshake is complete and the neighbor state is `SECURE`. It contains the nonce, the ChaCha20 ciphertext of the original OLSR fields, and the Poly1305 authentication tag.
- The length field within each TLV indicates the size of the subsequent value field. For TLV `0x11`, the length is variable depending on the size of the original OLSR payload being encrypted, plus the fixed nonce (12 bytes) and tag (16 bytes) sizes.
- Legacy OLSR nodes that do not implement these extensions will ignore unknown TLVs according to standard TLV processing rules.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------------------------------------------------------+
|          OLSR Common Message Header (Type=HELLO, etc.)        |
+---------------------------------------------------------------+
| Htime | Willing | Link Code | Resvd |       Link Message Size |
+---------------------------------------------------------------+
|             Neighbor Interface Address 1 (4 bytes)            |
+---------------------------------------------------------------+
| Optional Link Metric 1 / Padding                              |
+---------------------------------------------------------------+
~                              ...                              ~
+---------------------------------------------------------------+
|             Neighbor Interface Address N (4 bytes)            |
+---------------------------------------------------------------+
| Optional Link Metric N / Padding                              |
+---------------------------------------------------------------+
| TLV Type=0x0F| TLV Length=800| Kyber Public Key (pk)          |
| (KEM PubKey) | (800 bytes)   | (Value continues...)           |
+--------------+---------------+--------------------------------+
| TLV Type=0x10| TLV Length=768| Kyber Ciphertext (ct)          |
| (KEM Cipher) | (768 bytes)   | (Value continues...)           |
+--------------+---------------+--------------------------------+
| TLV Type=0x11| TLV Length=Var| Nonce (12 bytes)               |
| (AEAD Data)  | (Variable)    +--------------------------------+
|              |               | Encrypted HELLO Payload        |
|              |               | (Specific Hdr + Neighbor Info)|
|              |               +--------------------------------+
|              |               | Authentication Tag (16 bytes) |
+--------------+---------------+--------------------------------+
```

Fig. 1: Enhanced OLSR HELLO Message Structure. Shows standard fields followed by optional KEM TLVs (0x0F, 0x10) for handshake and the AEAD TLV (0x11) for steady-state protection.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------------------------------------------------------+
|           OLSR Common Message Header (Type=TC, etc.)          |
+---------------------------------------------------------------+
|           Advertised Neighbor Seq Num (ANSN)   | Reserved     |
+---------------------------------------------------------------+
|            Advertised Neighbor Main Address 1 (4 bytes)       |
+---------------------------------------------------------------+
| Optional Metric 1 / Padding                                   |
+---------------------------------------------------------------+
~                              ...                              ~
+---------------------------------------------------------------+
|            Advertised Neighbor Main Address N (4 bytes)       |
+---------------------------------------------------------------+
| Optional Metric N / Padding                                   |
+---------------------------------------------------------------+
| TLV Type=0x11| TLV Length=Var| Nonce (12 bytes)               |
| (AEAD Data)  | (Variable)    +--------------------------------+
|              |               | Encrypted TC Payload           |
|              |               | (Specific Hdr + Advert. Nbrs) |
|              |               +--------------------------------+
|              |               | Authentication Tag (16 bytes) |
+--------------+---------------+--------------------------------+
```

Fig. 2: Enhanced OLSR TC Message Structure. Shows standard fields followed only by the AEAD TLV (0x11) used during steady-state protection.

F. Sequence of Operations

Figure 3 details the sequence of operations for establishing a secure neighbor relationship and exchanging protected control messages.

The process, depicted in Figure 3, starts with a **Key Exchange Handshake**. Node A generates (`Q: liboqs`) a Kyber keypair and sends its public key in a HELLO TLV (`H: 0x0F`). Node B, upon receiving A's public key, generates its own keypair, encapsulates (`Q`) against A's key yielding ciphertext `ct_B` and shared secret `ss_B`, derives the session key via HKDF (`N`), and replies with its public

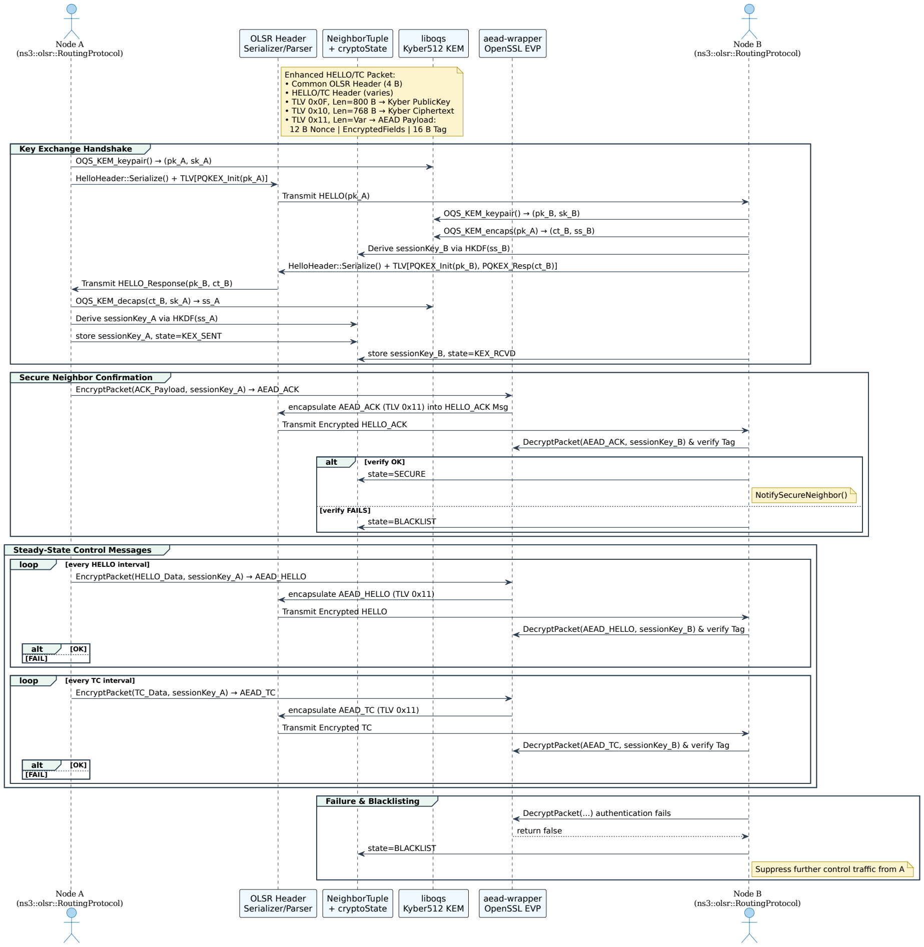
Node A
(ns3::olsr::RoutingProtocol)
OLSR Header
Serializer/Parser
NeighborTuple
+ cryptoState
liboqs
Kyber512 KEM
aead-wrapper
OpenSSL EVP
Node B
(ns3::olsr::RoutingProtocol)
Enhanced HELLO/TC Packet:
• Common OLSR Header (4 B)
• HELLO/TC Header (varies)
• TLV 0x0F, Len=800 B → Kyber PublicKey
• TLV 0x10, Len=768 B → Kyber Ciphertext
• TLV 0x11, Len=Var → AEAD Payload:
12 B Nonce | EncryptedFields | 16 B Tag
Key Exchange Handshake
OQS_KEM_keypair() → (pk_A, sk_A)
HelloHeader::Serialize() + TLV[PQKEX_Init(pk_A)]
Transmit HELLO(pk_A)
OQS_KEM_keypair() → (pk_B, sk_B)
OQS_KEM_encaps(pk_A) → (ct_B, ss_B)
Derive sessionKey_B via HKDF(ss_B)
HelloHeader::Serialize() + TLV[PQKEX_Init(pk_B), PQKEX_Resp(ct_B)]
Transmit HELLO_Response(pk_B, ct_B)
OQS_KEM_decaps(ct_B, sk_A) → ss_A
Derive sessionKey_A via HKDF(ss_A)
store sessionKey_A, state=KEX_SENT
store sessionKey_B, state=KEX_RCVD
Secure Neighbor Confirmation
EncryptPacket(ACK_Payload, sessionKey_A) → AEAD_ACK
encapsulate AEAD_ACK (TLV 0x11) into HELLO_ACK Msg
Transmit Encrypted HELLO_ACK
DecryptPacket(AEAD_ACK, sessionKey_B) & verify Tag
alt
[verify OK]
state=SECURE
NotifySecureNeighbor()
[verify FAILS]
state=BLACKLIST
Steady-State Control Messages
loop
[every HELLO interval]
EncryptPacket(HELLO_Data, sessionKey_A) → AEAD_HELLO
encapsulate AEAD_HELLO (TLV 0x11)
Transmit Encrypted HELLO
DecryptPacket(AEAD_HELLO, sessionKey_B) & verify Tag
alt
[OK]
[FAIL]
loop
[every TC interval]
EncryptPacket(TC_Data, sessionKey_A) → AEAD_TC
encapsulate AEAD_TC (TLV 0x11)
Transmit Encrypted TC
DecryptPacket(AEAD_TC, sessionKey_B) & verify Tag
alt
[OK]
[FAIL]
Failure & Blacklisting
DecryptPacket(...) authentication fails
return false
state=BLACKLIST
Suppress further control traffic from A
Node A
(ns3::olsr::RoutingProtocol)
OLSR Header
Serializer/Parser
NeighborTuple
+ cryptoState
liboqs
Kyber512 KEM
aead-wrapper
OpenSSL EVP
Node B
(ns3::olsr::RoutingProtocol)

Fig. 3: Sequence diagram detailing the post-quantum key-encapsulation handshake and subsequent AEAD protection steps in OLSR control messages between Node A and Node B, involving crypto libraries (Q: liboqs, C: OpenSSL/AEAD), Header processing (H), and Neighbor state management (N).

key (TLV 0x0F) and ct_B (TLV 0x10). Node A receives this, decapsulates (Q) ct_B using its secret key to get ss_B, derives the identical session key (HKDF), and updates its state (N). **Secure Neighbor Confirmation** may follow, where A sends an AEAD-protected ACK (C: OpenSSL, TLV 0x11). Node B decrypts/verifies (C) and transitions its state for A to SECURE (N). In the subsequent **Steady-State**, all HELLO/TC messages are protected. The sender encrypts/authenticates the OLSR payload using the session key (C) and places the nonce, ciphertext, and tag into TLV 0x11. The receiver extracts the TLV, decrypts/verifies (C); only verified payloads proceed to OLSR logic. A **Failure** in decryption/verification (C) results in packet discard and potential state transition to BLACKLIST (N).

Our PQC enhancement replaces standard OLSR processing with cryptographic operations integrated via TLVs:

- **KEM Handshake (HELLO TLVs):** A two-message exchange establishes a shared secret using Kyber512.
  Initiator Send: KeyGen (liboqs); Send Public Key (TLV 0x0F).
  Responder Rcv/Reply: KeyGen; Encapsulate (liboqs) against Sender PK; Derive Session Key (HKDF); Send own PK (TLV 0x0F) + KEM Ciphertext (TLV 0x10).
  Initiator Rcv: Decapsulate KEM Ciphertext (liboqs); Derive Session Key (HKDF) → SECURE State achieved.
- **AEAD Protection (Steady-State HELLO/TC):** Subsequent messages on secure links are protected.
  Send (SECURE State): AEAD Encrypt OLSR Payload (OpenSSL, ChaCha20-Poly1305) → Pack into TLV 0x11 (Nonce, Ciphertext, Tag).
  Receive (SECURE State): Extract TLV 0x11; AEAD Decrypt/Verify (OpenSSL). If OK → Pass plaintext to Standard OLSR Processing; Else → Discard Packet

IV. Implementation Details

For Post-Quantum key establishment, we select **Kyber512** [3] as the KEM, offering NIST Security Level 1 with relatively compact keys (800 B public key, 768 B ciphertext). Its soft-

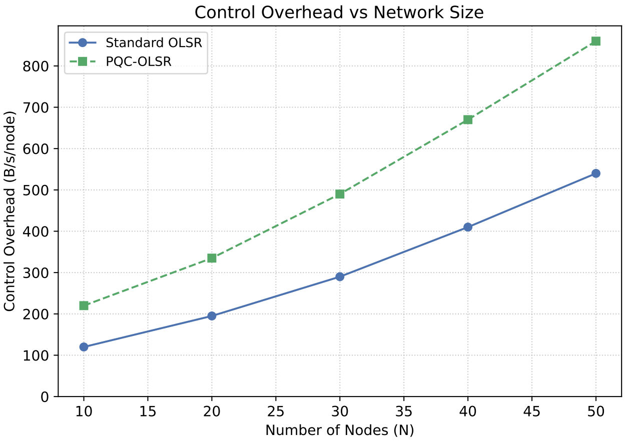

Fig. 4: Control Plane Overhead per Node vs. Network Size. Compares baseline OLSR traffic with the additional overhead incurred by system-wide PQC KEM and AEAD protection.

ware performance is suitable for target environments. Key generation, encapsulation, and decapsulation rely on the **`liboqs`** C library's KEM API [5]. For ongoing message protection, we choose the **ChaCha20-Poly1305** AEAD scheme [4], known for its 256-bit security and high software speed, particularly beneficial on platforms without AES hardware acceleration. We utilize the high-level EVP interface provided by the standard **`OpenSSL`** library [6] for these symmetric operations. The integration within ns-3 targets the existing OLSR module (`src/olsr`). The core PQC logic, state machines (managing `INIT` through `SECURE`/ `BLACKLIST` states per neighbor), and cryptographic operations are implemented by modifying classes like `ns3::olsr::RoutingProtocol`. OLSR packet/header classes require adaptation for TLV handling, and per-neighbor data structures (e.g., `NeighborTuple`) must be extended to store cryptographic state and derived session keys.

V. Evaluation

We evaluate the performance impact of the proposed PQC enhancement on OLSR using the ns-3 network simulator. The simulation environment models a mobile ad-hoc network with nodes moving according to the Random Waypoint model within a 500x500m area or placed on a static grid. We utilize the IEEE 802.11g WiFi model with default settings and log-distance propagation loss. The cryptographic overhead is measured by instrumenting the OLSR control packet flow. Handshake time is measured from the reception of the initial KEM public key (triggering KEX_RCVD state) to the successful processing of the corresponding KEM ciphertext (reaching SECURE state). A simulated delay of 1ms is added for each PQC operation (Encapsulate, Decapsulate, AEAD Encrypt/Decrypt) to approximate computational cost.

Figure 4 presents the average control plane overhead per node as the network size increases. As expected, the baseline OLSR overhead grows with network size due to increased HELLO traffic density and more extensive topology information disseminated via TC messages. PQC-OLSR demonstrates significantly higher overhead. This increase stems directly from the added cryptographic payloads: the 800-byte public keys and 768-byte ciphertexts exchanged during the initial KEM handshake between all neighboring pairs, and the smaller but persistent overhead of the AEAD nonce (12 bytes) and tag (16 bytes) appended to every secured HELLO and TC message thereafter.

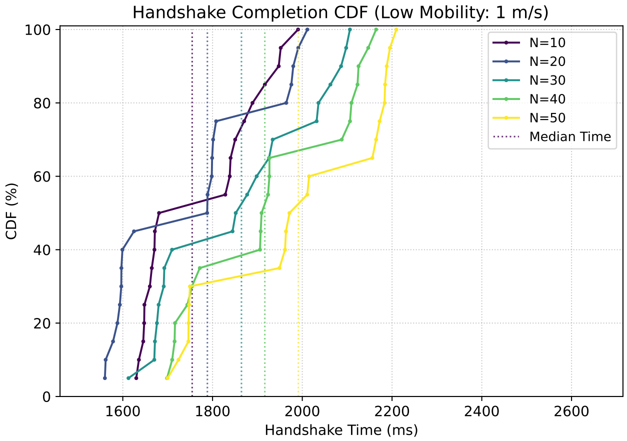

Fig. 5: Handshake Completion Time CDF (Low Mobility: 1 m/s). Shows the distribution of successful PQC KEM handshake durations for different network sizes in a low-mobility scenario. Vertical lines indicate median completion times.

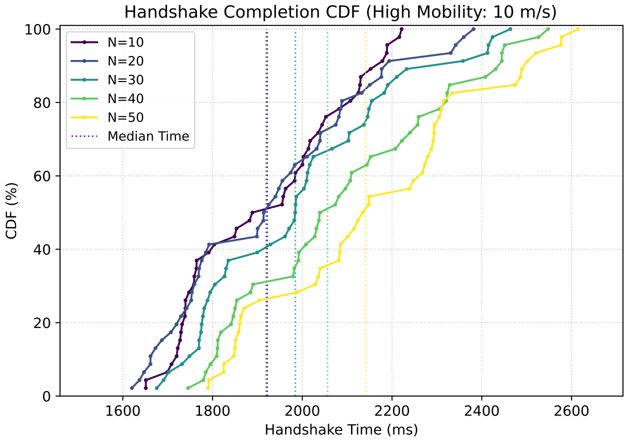

Fig. 6: Handshake Completion Time CDF (High Mobility: 10 m/s). Shows the distribution of successful PQC KEM handshake durations for different network sizes under higher mobility. Vertical lines indicate median completion times.

Figures 5 and 6 illustrate the Cumulative Distribution Function (CDF) of the PQC handshake completion times for varying network sizes (N). The handshake time represents the latency incurred in establishing a secure session key between two neighbors. In the low mobility case (Figure 5), handshake completion times exhibit some dependency on network size, with larger networks potentially showing slightly delayed completions, likely due to increased background traffic and processing load delaying the exchange of necessary KEM packets embedded within OLSR messages. The relatively tight distribution and distinct steps suggest that handshake progression is often tied to periodic OLSR events. In the higher mobility case , the handshake process is visibly impacted. The CDF curves are shifted to the right due to longer completion times. The distributions also appear more spread out, with shallower slopes between the steps and higher median values.

VI. Discussion

The evaluation highlights the feasibility of integrating post-quantum security into the OLSR control plane, albeit with performance trade-offs. The increased control overhead demonstrated by PQC-OLSR (Figure 4) is a direct consequence of the cryptographic additions. While significant, this cost provides essential resilience against advanced adversaries targeting the routing infrastructure. The impact must be weighed against network capacity and application requirements. Handshake latency analysis shows that secure session establishment is viable within typical OLSR operational timescales at no significant compute expence. However, the increased delay and variance under higher mobility will pose a challenge in dynamic environments. The simulated 1ms cryptographic processing delay also contributes non-negligibly to this latency, indicating that PQC computational costs, while manageable, are a relevant factor. Nevertheless, the (relatively) unaffected data plane performance serves as a confitmarion of the targeted nature of our proposed enhancement which only touches upon the L3 control-plane.

VII. Conclusion and Future Work

Our research developed and analyzed PQC-enhanced OLSR design that ensures confidentiality, integrity and neighbor authentication with resistance to both classical and quantum attacks. We implemented Kyber512 and ChaCha20-Poly1305 security through TLV extensions which allowed practical application to current mesh network protocols. Simulation findings measure both protocol overhead growth and explain the duration of handshakes while showing their responsiveness to mobile nodes. The next steps require detailed research on advanced key management techniques (rekeying and revocation) while developing universal security principles for protocols that extend this method past OLSR, following a similar development methdodology as the authors in [18]. Future quantum-resistant mesh networks need complete end-to-end security which requires an integration of Layer 3 security with data plane protection mechanisms.

Acknowledgement

This work has received funding from the European Union and the Swiss State Secretariat for Education, Research and Innovation (SERI) under the grant agreement No 101192749 (XTRUST-6G). Disclaimer: Funded by the European Union. Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] T. Clausen and P. Jacquet, Eds., ”Optimized Link State Routing Protocol (OLSR),” IETF, RFC 3626, Oct. 2003. [Online]. Available: https://www.rfc-editor.org/info/rfc3626. DOI: 10.17487/RFC3626.

[2] P. W. Shor, ”Algorithms for Quantum Computation: Discrete Logarithms and Factoring,” in *Proc. 35th Annu. Symp. Found. Comput. Sci.*, Santa Fe, NM, USA, Nov. 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700.

[3] R. Avanzi et al., ”CRYSTALS-Kyber Specification Version 3.02,” CRYSTALS Consortium, Tech. Rep., Aug. 2021. [Online]. Available: https://pq-crystals.org/kyber/data/kyber-specification-v3.02.pdf

[4] Y. Nir and A. Langley, ”ChaCha20 and Poly1305 for IETF Protocols,” IETF, RFC 8439, Jun. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8439. DOI: 10.17487/RFC8439.

[5] Open Quantum Safe Project, ”liboqs - C Library for Post-Quantum Cryptography,” 2024. [Online]. Available: https://openquantumsafe.org. (Accessed: May 5, 2024).

[6] OpenSSL Project, ”OpenSSL Cryptography and SSL/TLS Toolkit,” 2024. [Online]. Available: https://www.openssl.org. (Accessed: May 5, 2024).

[7] National Institute of Standards and Technology (NIST), ”Post-Quantum Cryptography,” 2024. [Online]. Available: https://csrc.nist.gov/projects/post-quantum-cryptography. (Accessed: May 5, 2024).

[8] J. Bobrysheva and S. Zapechnikov, ”Post-quantum Secure Group Messaging,” in *Proc. IEEE Conf. Russian Young Researchers Electr. Electron. Eng. (ElConRus)*, St. Petersburg and Moscow, Russia, Jan. 2021, pp. 2323–2326. DOI: 10.1109/ElConRus51938.2021.9396513.

[9] O. Sobolewski et al., ”Quantum-Resistant Key Management for Underwater Acoustic Multicast Communication,” in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Washington, DC, USA, Oct. 2024, pp. 196–201. DOI: 10.1109/MILCOM61039.2024.10773852.

[10] M. K. Misra, R. Mathur, and R. Tripathi, ”On Post Quantum Wireless Communication Security,” in *Proc. 5th Int. Conf. Inf. Syst. Comput. Netw. (ISCON)*, Mathura, India, Oct. 2021, pp. 1–6. DOI: 10.1109/ISCON52037.2021.9702489.

[11] M. Kara, K. Karampidis, S. Panagiotakis, M. Hammoudeh, M. Felemban, and G. Papadourakis, ”Lightweight and Efficient Post Quantum Key Encapsulation Mechanism Based on Q-Problem,” *Electronics*, vol. 14, no. 4, p. 728, Feb. 2024. DOI: 10.3390/electronics14040728.

[12] S. Hoque, A. Aydeger, and E. Zeydan, ”Exploring Post Quantum Cryptography with Quantum Key Distribution for Sustainable Mobile Network Architecture Design,” arXiv preprint arXiv:2404.10602, Apr. 2024. [Online]. Available: https://arxiv.org/abs/2404.10602

[13] O. Amer, W. O. Krawec, M. Z. Hossain, V. U. Manfredi, and B. Wang, ”Dynamic Routing and Post-Processing Strategies for Hybrid Quantum Key Distribution Networks,” in *Proc. IEEE 44th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jersey City, NJ, USA, Jul. 2024, pp. 1224–1235. DOI: 10.1109/ICDCS60910.2024.00116.

[14] J. Qian, J. Luo, and J. Chen, ”Hybrid Trusted Quantum Key Distribution Network Routing Scheme for Power Grid Environment,” in *Proc. 10th Int. Conf. Comput. Commun. (ICCC)*, Chengdu, China, Dec. 2024, pp. 422–427. DOI: 10.1109/ICCC62609.2024.10941900.

[15] M. Nie, G. Yang, and R. Wei, ”Quantum Self-organizing Network and Routing Protocol for Cooperative Communication of Intelligent Robot Soccer Game,” in *Proc. 5th Int. Conf. Control Robot. Eng. (ICCRE)*, Osaka, Japan, Apr. 2020, pp. 48–52. DOI: 10.1109/ICCRE49379.2020.9096471.

[16] D. Soler, I. Cillero, C. Dafonte, M. Fernández-Veiga, A. Fernández Vilas, and F. J. Nóvoa, ”QKDNetSim+: Improvement of the quantum network simulator for NS-3,” *SoftwareX*, vol. 26, p. 101685, 2024. DOI: 10.1016/j.softx.2024.101685.

[17] H. Krawczyk and P. Eronen, ”HMAC-based Extract-and-Expand Key Derivation Function (HKDF),” IETF, RFC 5869, May 2010. [Online]. Available: https://www.rfc-editor.org/info/rfc5869. DOI: 10.17487/RFC5869.

[18] P. Bajpai and P. K. Mishra, ”A Secure Peer to Peer Key Validation Protocol Using Symmetrical Group,” in *Proc. Int. Conf. Comput., Sci. Commun. (ICCSC)*, Greater Noida, India, Feb. 2024, pp. 1–6. DOI: 10.1109/ICCSC62048.2024.10830299.