

# Post-Quantum Key Agreement for CANsec over CAN XL: A Worst-Case Time-to-Key Analysis with ML-KEM

Omar Al-Daihani\*

\* Computer Engineer, B.Sc. in Computer Engineering  
Kuwait  
E-mail: aldaihaniali@gmail.com

DOI: 10.29322/IJSRP.16.10.2026.p17808  
<https://dx.doi.org/10.29322/IJSRP.16.10.2026.p17808>

Paper Received Date: 4th September 2026  
Paper Acceptance Date: 12th October 2026  
Paper Publication Date: 18th October 2026

**Abstract-** CAN XL supports payloads of up to 2048 bytes per frame. CANsec adds a MACsec-like data plane but leaves key agreement unspecified. Pre-shared keys alone provide no forward secrecy, and classical elliptic-curve exchange is vulnerable to cryptographically relevant quantum computers, a concern for long-lived vehicles. This paper proposes PQ-CKA, a CANsec key agreement based on ML-KEM (FIPS 203): the server broadcasts an ephemeral encapsulation key, members return ciphertexts authenticated with pre-shared keys, and the secure association key is distributed under KEM-derived key-encryption keys. A compositional worst-case time-to-key (WC-TTK) bound accounts for CAN XL frame durations, arbitration with transmission errors, first-in-first-out server processing and execution times derived from measured Cortex-M4 cycle counts. Evaluation uses a real-vehicle message set with 106 identifiers and 2,392 frames/s at 50% bus load, with 10 000-run simulations checking the bound. For 16 ECUs, ML-KEM-768 gives a WC-TTK of 122.7 ms versus 119.4 ms for X25519; all members hold the new key after 94.1 ms. The same protocol requires 363.5 ms on CAN FD at 2 Mbit/s. Mutual ML-DSA authentication increases the bound to 337.3 ms and makes it probabilistic because signing has an unbounded geometric tail. Sequential protocol rounds and the server's decapsulation rate dominate time-to-key, rather than post-quantum message size.

**Index Terms-** CAN XL, CANsec, in-vehicle networks, key agreement, ML-KEM, post-quantum cryptography, real-time systems, response-time analysis

## I. INTRODUCTION

In-vehicle networks are moving towards zonal architectures in which a few zone controllers aggregate communication from many electronic control units (ECUs). CAN XL, the third generation of Controller Area Network standardised in ISO 11898-1:2024 [1], fits this role: it retains the priority arbitration of classical CAN, switches to a faster data phase and permits payloads of 1–2048 bytes per frame [2]. CANsec adds authenticated encryption and freshness at the data-link layer, with its data plane specified in CiA 613-2 [3]. The specification reuses MACsec security functions [4] while explicitly excluding the key-exchange and distribution protocol [5]. Recent research mapping CAN XL frames onto a MACsec engine confirms this distinction between data and control planes and identifies an adapted MACsec Key Agreement as the missing control plane [6].

A CANsec key agreement must reconcile long-term confidentiality with timely completion. Shor's algorithm breaks the discrete-logarithm and factoring problems underlying elliptic-curve Diffie–Hellman and RSA [7]. Estimates of the resources required to factor RSA-2048 have declined from 20 million noisy qubits [8] to fewer than one million [9]. Mosca's criterion identifies risk when a secret's lifetime plus migration time exceeds the time until a cryptographically relevant quantum computer becomes available [10]. Vehicle platforms frozen today will remain in service well into that window, exposing recorded CAN XL traffic protected by elliptic-curve key exchange to later decryption. NIST has standardised ML-KEM for key establishment [11] and ML-DSA for signatures [12], and issued recommendations for key-encapsulation mechanisms [13]. Its draft transition plan proposes deprecating quantum-vulnerable public-key algorithms at the 112-bit level after 2030 and disallowing them after 2035 [14]. Automotive requirements add pressure through UN Regulation No. 155 [15] and ISO/SAE 21434 [16], which require cybersecurity management across the vehicle lifecycle.

The second requirement is timeliness. Secured communications within a CANsec zone cannot begin until all members have received the secure association key (SAK), while key agreement shares the bus with time-critical control traffic. Therefore, an automotive system integrator needs an assured upper limit for the time it takes for all nodes in a zone to receive and be able to use the newly

agreed upon key after the wake-up or rekeying trigger occurs. Average handshake latency does not provide such assurance. This worst-case time-to-key (WC-TTK) depends on the size of the frames being transmitted, how long it takes to acquire access to the bus (bus arbitration), errors during transmission and how much time it takes to perform the cryptographic operations on automotive-class processors. Each of these factors needs to be included as a contributor to the overall delay. Post-quantum messages are large: an ML-KEM-768 encapsulation key has 1184 bytes [11], which is prohibitive on classical CAN but fits into a single CAN XL frame. No published analysis quantifies what this means for the WC-TTK of a CANsec zone.

This paper closes that gap. Its contributions are:

- **Protocol.** PQ-CKA establishes keys for CANsec zones through a key server that broadcasts an ephemeral ML-KEM encapsulation key. Members return ciphertexts authenticated with pre-shared keys, and the SAK is distributed using key-encryption keys derived from the KEM secrets and pre-shared keys. Alternative designs use per-ECU key pairs, X25519, a hybrid combiner or ML-DSA signatures.
- **Analysis.** A compositional WC-TTK bound combines worst-case CAN XL and CAN FD frame durations with response-time analysis incorporating release jitter and sporadic errors, burst transmission of multi-frame protocol data units (PDUs), first-in-first-out key-server processing and execution-time bounds derived from measured cycle counts.
- **Evaluation.** Traffic from a public production-vehicle trace is combined with measured Cortex-M4 cycle counts for ML-KEM, ML-DSA and X25519. Discrete-event simulation checks the bound across 10 000 random phasings for each variant.
- **Design rules.** The evaluation provides quantitative guidance on key-server performance, zone size, priority assignment, bit rates and authentication.

Section II reviews the background and related work. Section III defines the system model and the protocol, Section IV the timing analysis, and Section V the evaluation. Section VI discusses the design implications and limitations, and Section VII concludes.

## II. BACKGROUND AND RELATED WORK

### A. CAN XL and CANsec

A CAN XL frame begins with arbitration at the nominal bit rate, where the 11-bit priority identifier resolves bus access bit by bit, as in classical CAN. After the arbitration-to-data switching sequence, the transceiver enters the fast data phase. This carries the service data unit type (SDT), simple/extended content (SEC) bit, 11-bit data length code, 13-bit preface CRC, 8-bit virtual CAN network identifier (VCID), 32-bit acceptance field, data field of 1–2048 bytes and 32-bit frame CRC [1], [2]. Dynamic stuffing applies in arbitration; fixed stuff bits in the data phase make its length independent of payload values. The large data field permits Ethernet tunnelling, supporting the hybrid CAN XL–Ethernet zonal architectures under evaluation [17]. For classical CAN and CAN FD, AUTOSAR Secure Onboard Communication (SecOC) [18] usually adds authentication above the data-link layer through a truncated message authentication code and optional freshness value, without confidentiality. CANsec provides protection at the data-link layer [6].

CANsec protects CAN XL frames using a MACsec-derived protocol data unit. In the mapping of Wiemer et al. [6], protocol control information occupies 8 bytes for the association number, 32-bit packet number and 16-bit secure-channel identifier. A 2-byte padding field aligns the encrypted data, and a 16-byte integrity check value completes the frame, adding 26 bytes to the payload. Each secure channel contains secure associations whose keys (SAKs) must be installed at every receiver before protected frames can be accepted. Payloads exceeding one frame are segmented through the frame fragmentation service in CiA 613-3 [19]. The data plane does not specify how these keys are distributed to the nodes [5].

### B. Key management in MACsec

Ethernet uses MACsec Key Agreement (MKA), defined in IEEE 802.1X-2020, to distribute secure association keys [20]. Participants sharing a connectivity association key elect a key server, which generates the SAK and wraps it with a key-encryption key derived from the association key. That association key may be pre-shared or established through EAP authentication, typically EAP-TLS using elliptic-curve or finite-field Diffie–Hellman. A pre-shared association key depends only on symmetric primitives and thus resists quantum attacks; its compromise, however, exposes every SAK previously distributed under it. EAP-TLS derivation provides forward secrecy but remains vulnerable to quantum attacks. Buruaga et al. [21] addressed these competing requirements for Ethernet through hybrid authenticated key exchange combining classical, post-quantum and quantum-key-distribution inputs. CAN XL has no comparable proposal, and its bus topology, arbitration and limited processing resources require a design assessed for worst-case timing.

### C. Post-quantum key encapsulation and signatures

ML-KEM, standardised in FIPS 203 [11], is a key-encapsulation mechanism derived from CRYSTALS-Kyber [22]. KeyGen creates an encapsulation key  $ek$  and a decapsulation key  $dk$ .  $\text{Encaps}(ek)$  generates a 32-byte shared secret and ciphertext  $ct$ , while  $\text{Decaps}(dk, ct)$  recovers that secret. Table 1 gives object sizes for the three parameter sets alongside ML-DSA [12] and X25519 [23]. KEMs also support authentication: KEMTLS replaces TLS handshake signatures with KEM-based authentication [24], while post-quantum WireGuard combines KEMs with an optional pre-shared symmetric key [25]. Hybrid combiners such as X-Wing join X25519 with ML-KEM-768, keeping the session key secure as long as either component remains secure [26].

**Table 1.** Object sizes (bytes) of the post-quantum and classical primitives considered (FIPS 203, FIPS 204, RFC 7748).

| Primitive   | NIST category | Public key | Ciphertext / signature | Shared secret |
|-------------|---------------|------------|------------------------|---------------|
| ML-KEM-512  | 1             | 800        | 768                    | 32            |
| ML-KEM-768  | 3             | 1184       | 1088                   | 32            |
| ML-KEM-1024 | 5             | 1568       | 1568                   | 32            |
| ML-DSA-44   | 2             | 1312       | 2420                   | –             |
| ML-DSA-65   | 3             | 1952       | 3309                   | –             |
| ML-DSA-87   | 5             | 2592       | 4627                   | –             |
| X25519      | – (classical) | 32         | 32                     | 32            |

The pqm4 project reports measured cycle counts for embedded implementations on the ARM Cortex-M4 at 24 MHz, avoiding distortion from flash wait states [27], [28]. The fastest published Cortex-M4 implementation of X25519 requires 625 358 cycles per scalar multiplication [29]. These measurements provide the execution-time inputs for the analysis in Section IV-D. Existing automotive research includes Fritzmann et al.'s study of post-quantum cryptography [30], Castiglione and Elia's combination of such cryptography with network segmentation for in-vehicle communication [31], and Lotto et al.'s survey of CAN authentication security properties [32]. Cena et al. examined multicast origin authentication in MACsec and CANsec [33]. None of these studies analyses CANsec key agreement or its worst-case timing.

#### D. Timing analysis of CAN

Tindell et al. introduced CAN response-time analysis [34]. Davis et al. later showed that the original test could be optimistic and provided the revised busy-period formulation now used [35]. Error models extend the analysis to include frame retransmissions caused by bit errors [36], [37]. Ikumapayi et al. investigated authentication's impact on schedulability [38], Berisa et al. compared the timing of classical CAN, CAN FD and CAN XL [39], and Laufenberg et al. developed a simulation framework covering these generations [40]. Table 2 positions the present study within this literature.

**Table 2.** Positioning of this work.

| Work                                             | Post-quantum | CAN XL / CANsec | Key agreement | Worst-case timing  |
|--------------------------------------------------|--------------|-----------------|---------------|--------------------|
| Davis et al. [35]; Punnekkat et al. [36]         | –            | –               | –             | ✓                  |
| Ikumapayi et al. [38]                            | –            | –               | –             | ✓ (authentication) |
| Berisa et al. [39]                               | –            | ✓ (timing)      | –             | ✓                  |
| Wiemer et al. [6]; Cena et al. [33]              | –            | ✓               | –             | –                  |
| Buruaga et al. [21]                              | ✓            | – (MACsec)      | ✓             | –                  |
| Fritzmann et al. [30]; Castiglione and Elia [31] | ✓            | –               | partial       | –                  |
| This work                                        | ✓            | ✓               | ✓             | ✓                  |

### III. SYSTEM MODEL AND PROTOCOL

#### A. Architecture

A CANsec zone consists of a key server (KS), usually the zone controller, and  $N$  member ECUs connected to one CAN XL segment (Fig. 1). Periodic control frames and lower-priority bulk traffic, such as tunnelled Ethernet frames or logging data, share this segment. Each member  $i$  holds a long-term pre-shared key  $PSK_i$  with the KS, provisioned at the end of the production line and stored in its hardware security module, as in MKA with a pre-shared association key [20]. At start-up and association renewal, the KS generates the zone SAK and distributes it to every member.

#### B. Threat model and security goals

The adversary controls the bus under the Dolev–Yao model: it can record, inject, delay and drop frames, but cannot read hardware security modules. It may later acquire a cryptographically relevant quantum computer to attack recorded traffic or compromise a long-term key. The protocol must provide (i) authenticated SAK delivery to legitimate members only; (ii) key confirmation before activation, ensuring every member holds the new association key; (iii) SAK confidentiality against quantum adversaries; and (iv) forward secrecy, so later compromise of  $PSK_i$  does not expose past-session SAKs. As with CANsec itself, bus-flooding denial of service is outside scope, while random transmission errors are included in the timing analysis.

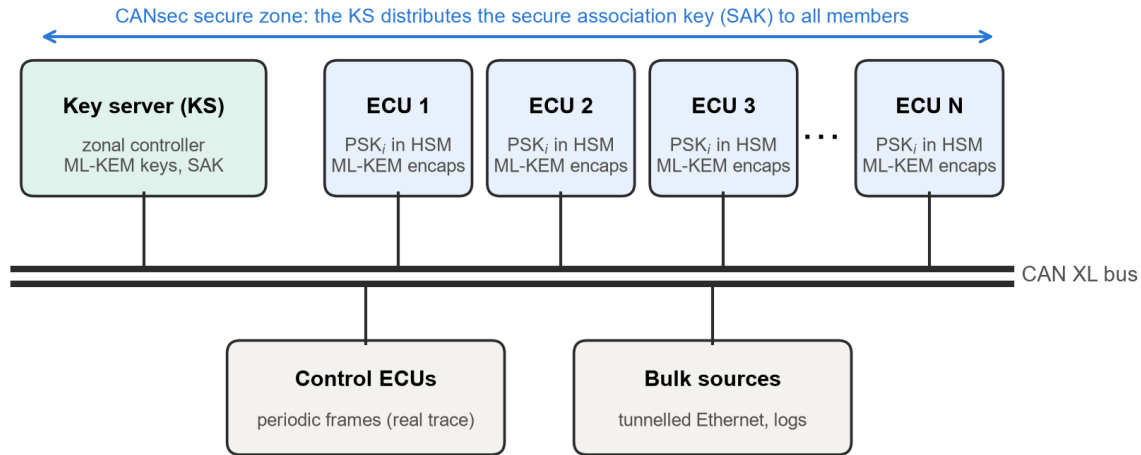


**Fig. 1.** System architecture: a key server and  $N$  member ECUs form a CANsec zone on a CAN XL segment shared with control and bulk traffic.

### C. PQ-CKA protocol

PQ-CKA follows five steps (Fig. 2). Each key-agreement PDU includes an 8-byte header identifying type, zone, session and sender, plus a 16-byte tag. PDUs travel in unprotected CAN XL frames with a dedicated SDT value, outside the protected data plane, as MKA packets do in MACsec.

- **M1 (KS  $\rightarrow$  all).** Only the KS generates the ephemeral ML-KEM key pair  $(ek, dk)$ , using ML-KEM.KeyGen before the trigger. It broadcasts  $ek$  with a 32-byte session nonce  $n_{KS}$ . For ML-KEM-768, the 1240-byte PDU fits into one CAN XL frame.
- **M2 (members  $\rightarrow$  KS).** Each member  $i$  encapsulates to the broadcast key, computing  $(ss_i, ct_i) = \text{ML-KEM.Encaps}(ek)$ . It returns  $ct_i$  in PDU M2 <sub>$i$</sub> , authenticated by a transcript tag under  $PSK_i$ . The ML-KEM-768 PDU occupies 1112 bytes.
- **KS processing.** The server verifies each ciphertext's tag, decapsulates to obtain  $ss_i = \text{ML-KEM.Decaps}(dk, ct_i)$ , and derives the member's key-encryption key using Eq. (1).
- **M3 (KS  $\rightarrow$  all).** The KS broadcasts the SAK wrapped separately for every member using AES key wrap [41]. Each entry requires 40 bytes plus a 2-byte member index; for  $N = 16$ , the 696-byte PDU fits into one frame.
- **M4 (members  $\rightarrow$  KS) and M5 (KS  $\rightarrow$  all).** Each member  $i$  unwraps the SAK, installs it for reception and acknowledges in M4 <sub>$i$</sub>  with a tag under  $KEK_i$ . Once the final acknowledgement arrives, the KS broadcasts M5 to activate the new association.

$$KEK_i = \text{KDF}(ss_i || PSK_i, n_{KS} || H(ek) || H(ct_i) || i) \quad (1)$$

The key-derivation function may be HKDF [42] or a KDF recommended in NIST SP 800-56C [43]. A member's  $KEK_i$  depends on both its ephemeral KEM secret and  $PSK_i$ ; an attacker therefore needs both to unwrap the SAK. If  $PSK_i$  is later compromised, prior-session SAKs remain protected because  $dk$  is erased after the session. Recorded traffic gives a quantum adversary no advantage as long as ML-KEM and the symmetric primitives remain secure. Tags on M2 <sub>$i$</sub>  and M4 <sub>$i$</sub>  authenticate members and confirm keys, while the M3 tag authenticates the KS under a key derived from all  $KEK_i$ . Installation time  $TTK_{\text{inst}}$  is reached when the last member has installed the SAK after M3; zone time-to-key TTK ends when M5 is received.

### D. Variants

Five variants serve as comparisons:

- **VX (classical baseline).** The broadcast structure of VB is retained, with X25519 replacing ML-KEM. M1 carries a 32-byte public key. Each member performs two scalar multiplications for key generation and shared-secret computation, while the KS performs one per member.
- **VH (hybrid).** M1 broadcasts both an ML-KEM-768 key and an X25519 key. Members return a ciphertext and their X25519 key; the KDF combines both secrets in the spirit of X-Wing [26].
- **VA (per-ECU key pairs).** After a short server trigger, each member generates its own ML-KEM key pair and sends  $ek_i$ . The KS encapsulates separately for each member and sends a unicast reply containing  $ct_i$  and the wrapped SAK. The member then decapsulates.
- **VS1 (signed broadcast).** This extends VB with an ML-DSA-44 signature on M1 from the KS. Members verify the signature before encapsulation; the 3.6-kbyte PDU requires two fragments.
- **VS2 (mutual signatures).** This extends VS1 by having each member sign its M2 <sub>$i$</sub> , which the KS verifies.

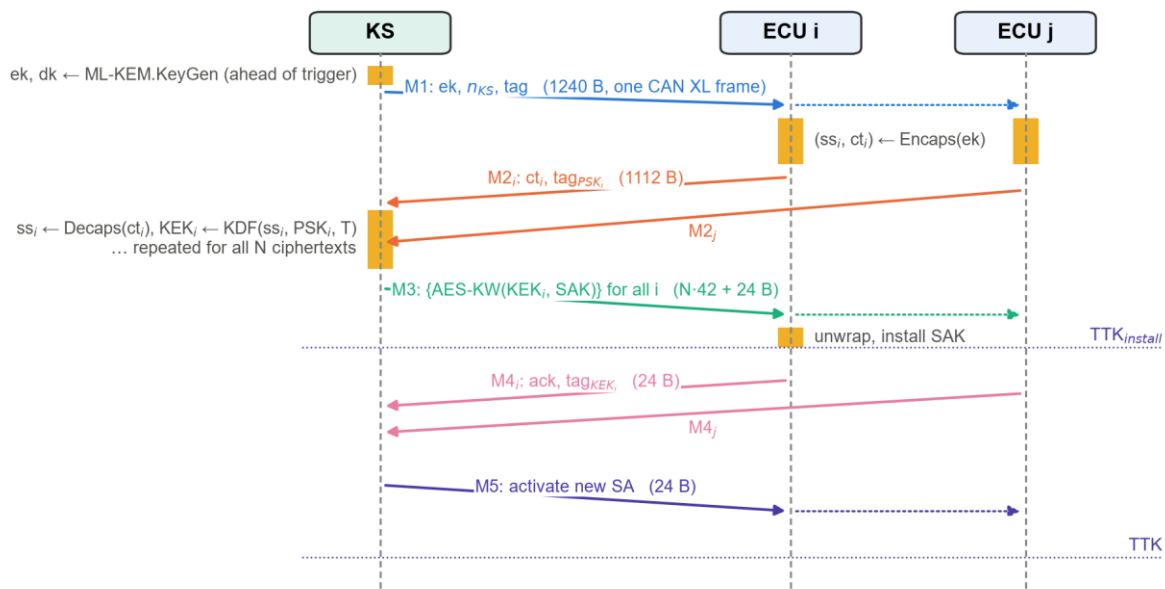


**Fig. 2.** Message sequence of PQ-CKA (variant VB) with the sizes for ML-KEM-768 and  $N = 16$ .

Table 3 lists PDU sizes and frame counts for each variant. All variants except VS1 and VS2 fit each PDU into a single CAN XL frame. On CAN FD with ISO 15765-2 transport, ML-KEM PDUs require 18–26 frames, and the signed PDUs of VS2 require up to 57 [44].

**Table 3.** Key-agreement PDU sizes (bytes) and number of frames on CAN XL and CAN FD (64-byte frames, ISO-TP), for  $N = 16$ .

| Variant                 | M1    | M2 <sub>i</sub> | M3              | CAN XL frames (M1/M2 <sub>i</sub> ) | CAN FD frames (M1/M2 <sub>i</sub> ) |
|-------------------------|-------|-----------------|-----------------|-------------------------------------|-------------------------------------|
| VX                      | 88    | 56              | 696             | 1 / 1                               | 2 / 1                               |
| VB, ML-KEM-512          | 856   | 792             | 696             | 1 / 1                               | 14 / 13                             |
| VB, ML-KEM-768          | 1,240 | 1,112           | 696             | 1 / 1                               | 20 / 18                             |
| VB, ML-KEM-1024         | 1,624 | 1,592           | 696             | 1 / 1                               | 26 / 26                             |
| VH                      | 1,272 | 1,144           | 696             | 1 / 1                               | 21 / 19                             |
| VA (M2 carries $ek_i$ ) | 56    | 1,208           | 1,152 (unicast) | 1 / 1                               | 1 / 20                              |
| VS1                     | 3,660 | 1,112           | 696             | 2 / 1                               | 59 / 18                             |
| VS2                     | 3,660 | 3,532           | 696             | 2 / 2                               | 59 / 57                             |

M4<sub>i</sub> and M5 carry 24 bytes each in all variants.

#### IV. WORST-CASE TIME-TO-KEY ANALYSIS

The timing analysis bounds the interval between a key-agreement trigger and receipt of M5 by all members. It composes frame transmission times (Section IV-A), PDU response times under interference, blocking and errors (IV-B and IV-C), cryptographic execution times (IV-D), and protocol sequencing (IV-E).

##### A. Frame transmission times

For a CAN XL data frame carrying  $n$  payload bytes at nominal bit rate  $R_N$  and data-phase bit rate  $R_D$ , the worst-case transmission duration is

$$C_{XL}(n) = \frac{19 + 4 + 16}{R_N} + \frac{116 + 8n + [(109 + 8n)/10]}{R_D} \quad (2)$$

The nominal-rate term includes 19 bits from frame start to the ADH switching bit, at most 4 dynamic stuff bits across the 17 bits through XLF, and 16 bits for the data-to-arbitration switch, acknowledgement, end of frame and intermission. The data-phase term includes the three switching bits DH1, DH2 and DL1; the header fields (SDT 8, SEC 1, DLC 11, stuff-bit count 4, preface CRC 13, VCID 8 and acceptance field 32); the data field; the 32-bit frame CRC; the 4-bit format check pattern; and one fixed stuff bit after every ten bits from SDT through the frame CRC [1], [2]. For comparison, the maximum duration of a CAN FD frame with an 11-bit identifier and bit-rate switching is

$$C_{FD}(n) = \frac{17 + 4 + 12}{R_N} + \frac{d + \lfloor d/4 \rfloor + 4 + c + \lfloor (4 + c)/4 \rfloor + 1}{R_D}, \quad d = 5 + 8n \quad (3)$$

where  $c$  equals 17 for  $n \leq 16$  and 21 otherwise. The term  $\lfloor d/4 \rfloor$  bounds dynamic stuff bits in the data field; fixed stuffing in the CRC field follows ISO 11898-1. Table 4 reports the resulting durations. At 0.5/10 Mbit/s, a 1240-byte ML-KEM-768 broadcast takes 1.182 ms and a maximum-length frame takes 1.893 ms. The latter also sets the longest blocking time that a frame can encounter on a CAN XL bus carrying maximum-length frames. Fig. 3 plots transmission duration against PDU size.

**Table 4.** Worst-case frame durations.

| Frame                               | Payload (bytes) | Bit rates (Mbit/s) | Duration (ms) |
|-------------------------------------|-----------------|--------------------|---------------|
| CAN XL control frame                | 8               | 0.5 / 10           | 0.098         |
| CAN XL, M2i (ML-KEM-768 ciphertext) | 1112            | 0.5 / 10           | 1.069         |
| CAN XL, M1 (ML-KEM-768 key)         | 1240            | 0.5 / 10           | 1.182         |
| CAN XL, M1 (ML-KEM-1024 key)        | 1624            | 0.5 / 10           | 1.520         |
| CAN XL, maximum length              | 2048            | 0.5 / 10           | 1.893         |
| CAN FD, full frame                  | 64              | 0.5 / 2            | 0.406         |

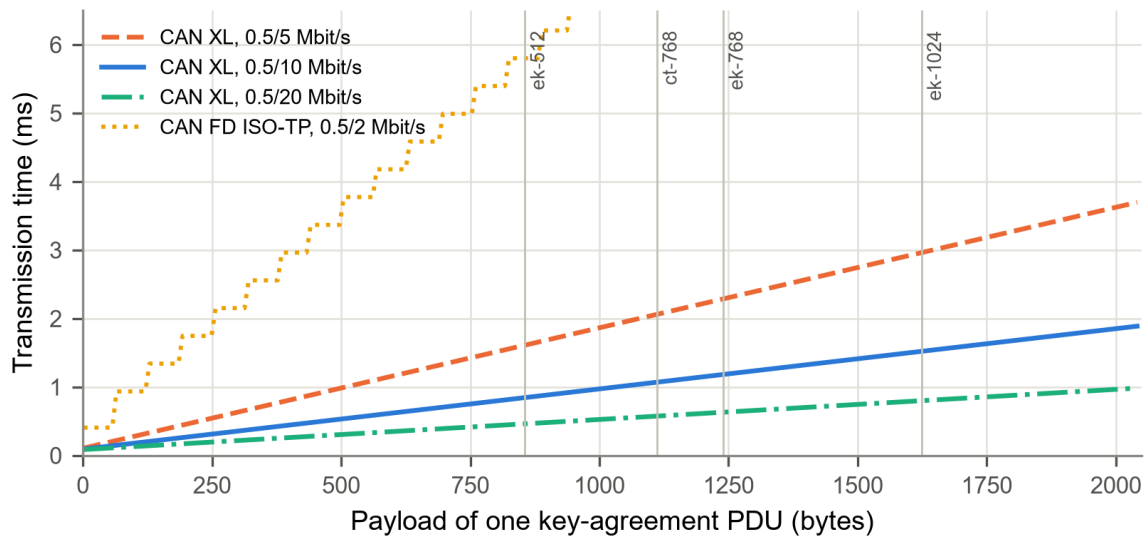


**Fig. 3.** Transmission time of a key-agreement PDU on CAN XL and on CAN FD with ISO-TP segmentation. Vertical lines mark the ML-KEM PDUs of Table 3.

### B. Response-time analysis

CAN arbitration is based on fixed priorities and is non-preemptive. Following Davis et al. [35], the queuing delay  $w_m$  of key-agreement frame  $m$ , with transmission duration  $C_m$ , satisfies the fixed-point equation

$$w_m = B_m + E_m(w_m + C_m) + \sum_{k \in hp(m)} \left\lceil \frac{w_m + J_k + \tau}{T_k} \right\rceil C_k + \sum_{j \in O(m)} C_j \quad (4)$$

and the worst-case response time is  $R_m = w_m + C_m$ . The blocking term  $B_m$  is the duration of the longest lower-priority frame, which may have begun transmission just before  $m$  was queued. The set  $hp(m)$  contains higher-priority periodic frames with period  $T_k$ , release jitter  $J_k$  and duration  $C_k$ ;  $\tau$  is one nominal bit time. Pending higher-priority one-shot protocol frames form  $O(m)$ , such as other members' ciphertexts, and each can delay  $m$  only once. The error term uses the sporadic fault model of Tindell et al. and Punnekkat et al. [34], [36]:

$$E_m(t) = (n_e + \lceil \frac{t}{T_e} \rceil)(O_{err} + \max_{k \in hp(m)} C_k) \quad (5)$$

with  $n_e$  errors in an initial burst, an additional error every  $T_e$ , signalling overhead  $O_{err}$  of 31 nominal bits, and retransmission of the longest frame of equal or higher priority.

### C. Multi-frame PDUs

A PDU exceeding one frame—VS1, VS2 and all post-quantum variants on CAN FD—is queued as a burst of  $q$  equal-priority frames. They transmit consecutively unless higher-priority traffic wins arbitration between frames. The last frame's queuing delay satisfies

$$w_m^{(q)} = B_m + (q - 1)C_m + E_m(w_m^{(q)} + C_m) + \sum_{k \in hp(m)} \lceil \frac{w_m^{(q)} + J_k + \tau}{T_k} \rceil C_k + \sum_{j \in O(m)} C_j \quad (6)$$

$$R_m^{(q)} = w_m^{(q)} + C_m \quad (7)$$

and  $R_m^{(q)}$  bounds completion of the burst. On CAN FD, ISO 15765-2 transport sends a first frame, waits for the receiver's flow-control frame and then sends consecutive frames [44]. The overall response time is the sum of the bounds for these three stages.

### D. Execution times

For a cryptographic operation with maximum measured cycle count  $c_{\max}$  on a processor clocked at  $f$ , execution time is bounded by

$$e = \frac{c_{\max}(1 + \mu)\kappa}{f\alpha} \quad (8)$$

where  $\mu = 0.10$  supplies a safety margin above the measured maximum,  $\kappa$  accounts for memory wait states ( $\kappa = 1$  for zero-wait-state memory), and  $\alpha$  is the CPU share allocated to key agreement. Inputs are the Cortex-M4 maxima reported by pqm4 [28] and the published X25519 count of 625 358 cycles [29], listed in Table 5. ML-KEM runs in constant time, and the margin covers its measured spread of less than 2% (Table 5). In contrast, ML-DSA signing uses rejection sampling: its main loop repeats a random number of times, with an expected count of 4.25 for ML-DSA-44 [12]. Signing therefore has no deterministic upper execution-time bound; Section V-H quantifies this distinction. Symmetric tag computation, key derivation and key wrapping are assumed to use the hardware security module's AES and hash accelerators, with a fixed allowance  $t_{\text{sym}} = 0.25$  ms per protocol step.

**Table 5.** Cortex-M4 cycle counts used in the analysis (pqm4, m4fspeed/m4f implementations) and execution-time bounds from Eq. (8).

| Operation                    | Average (cycles) | Maximum (cycles) | Spread (max/avg - 1) | Bound at 160 MHz (ms) | Bound at 300 MHz (ms) |
|------------------------------|------------------|------------------|----------------------|-----------------------|-----------------------|
| ML-KEM-512 Encaps            | 390 881          | 391 073          | 0.05%                | 2.69                  | 1.43                  |
| ML-KEM-512 Decaps            | 428 167          | 428 357          | 0.04%                | 2.94                  | 1.57                  |
| ML-KEM-768 KeyGen            | 642 096          | 651 103          | 1.40%                | 4.48                  | 2.39                  |
| ML-KEM-768 Encaps            | 658 754          | 667 769          | 1.37%                | 4.59                  | 2.45                  |
| ML-KEM-768 Decaps            | 707 827          | 716 842          | 1.27%                | 4.93                  | 2.63                  |
| ML-KEM-1024 Encaps           | 1 031 565        | 1 039 544        | 0.77%                | 7.15                  | 3.81                  |
| ML-KEM-1024 Decaps           | 1 094 008        | 1 101 987        | 0.73%                | 7.58                  | 4.04                  |
| ML-DSA-44 Sign               | 3 943 121        | 17 009 165       | 331.36%              | 116.94                | 62.37                 |
| ML-DSA-44 Verify             | 1 421 623        | 1 422 362        | 0.05%                | 9.78                  | 5.22                  |
| X25519 scalar multiplication | 625 358          | 625 358          | —                    | 4.30                  | 2.29                  |

### E. Composition of the time-to-key

For VB,  $e_{\text{Enc}}^{\text{ECU}}$  and  $e_{\text{Dec}}^{\text{KS}}$  denote the execution-time bounds from Eq. (8) on member and key-server processors. Ciphertext PDUs  $M_{2,1} \dots M_{2,N}$  are released together after every member completes encapsulation. Each PDU experiences one-shot interference from higher-priority ciphertexts, giving the following upper bounds on arrival times at the KS:

$$A_i = t_1 + e_{\text{Enc}}^{\text{ECU}} + t_{\text{sym}} + R_{M_{2,i}}, \quad t_1 = t_{\text{sym}} + R_{M_1} \quad (9)$$

The server decapsulates ciphertexts in arrival order. Sorting arrivals as  $A_{(1)} \leq \dots \leq A_{(N)}$ , the last decapsulation completes at

$$F_N = \max_{1 \leq k \leq N} (A_{(k)} + (N - k + 1)(e_{\text{Dec}}^{\text{KS}} + t_{\text{sym}})) \quad (10)$$

Delaying an arrival cannot reduce the completion time of a work-conserving first-in-first-out server. Substituting the arrival bounds  $A_i$  into Eq. (10) therefore gives an upper bound on actual completion time. The remaining protocol stages determine the two instants of interest:

$$\text{TTK}_{\text{inst}} = F_N + R_{M_3} + t_{\text{sym}}, \quad \text{TTK} = \text{TTK}_{\text{inst}} + \max_i R_{M_{4,i}} + t_{\text{sym}} + R_{M_5} \quad (11)$$

Other variants use the same composition with their respective PDUs and operations. In VA, the KS encapsulates for members in arrival order and immediately sends each unicast reply, with member-side decapsulation following the last reply. Each response time is bounded independently at a critical instant, making the composition safe but pessimistic. Section V-I measures the resulting gap.

### F. Simulation

A discrete-event bus simulator complements the analytical bound. Frames from each periodic source are released with random phases, and the protocol reacts to frame completions. ML-KEM execution times are sampled uniformly between measured minima and maxima; ML-DSA signing times follow the geometric iteration model in Section V-H. Each variant is simulated over 10 000 random trigger instants. A separate series of 2000 runs releases all periodic frames synchronously with the trigger and starts a maximum-length frame immediately beforehand, approximating the first protocol step's critical instant.

## V. EVALUATION

### A. Setup

The control traffic is based on the ambient capture "drive basic short" from the Real ORNL Automotive Dynamometer (ROAD) dataset, a recording of a production vehicle on a dynamometer [45], [46]. The 444.8 s capture contains 1,063,624 frames across 106 identifiers, each with an 8-byte payload. Each identifier is assigned the standard period nearest its median inter-arrival time: among others, 9 identifiers use 10 ms, 18 use 20 ms, 31 use 100 ms and 26 use 1 s. Release jitter equals the period minus the 5th percentile of observed inter-arrival times, capped at one period. Identifiers retain their original order when mapped to CAN XL priorities and occupy 23.3% of the bus at 0.5/10 Mbit/s. Four 1500-byte bulk streams raise total utilisation to 50%, while a lowest-priority stream of 2048-byte frames creates maximum blocking. Unless stated otherwise, key-agreement frames have priorities between control and bulk traffic. Table 6 gives the remaining parameters.

**Table 6.** Evaluation parameters (base case).

| Parameter                                                       | Value                                         |
|-----------------------------------------------------------------|-----------------------------------------------|
| Zone size $N$                                                   | 16 member ECUs                                |
| Bit rates (nominal / data phase)                                | 0.5 / 10 Mbit/s                               |
| Total bus load                                                  | 50% (control 23.3%, bulk and blocking frames) |
| Member ECU clock (Cortex-M4 equivalent)                         | 160 MHz                                       |
| Key-server clock (Cortex-M4 equivalent)                         | 300 MHz                                       |
| KEM parameter set / implementation                              | ML-KEM-768 / pqm4 m4fspeed                    |
| WCET margin $\mu$ , memory factor $\kappa$ , CPU share $\alpha$ | 0.10, 1.0, 1.0                                |
| Symmetric processing per step $t_{\text{sym}}$                  | 0.25 ms                                       |
| KS ephemeral key pair                                           | generated before the trigger                  |
| Error model                                                     | fault-free (Section V-G varies it)            |

### B. Base case

Table 7 compares WC-TTK, installation bounds and simulated distributions for the eight variants; Fig. 4 separates each bound by protocol phase. For  $N = 16$ , VB with ML-KEM-768 reaches zone time-to-key within 122.7 ms, with every member holding the SAK within 94.1 ms. The X25519 baseline VX needs 119.4 ms, making the post-quantum overhead only 2.8%. Member encapsulation is less expensive than the classical scalar multiplications: on Cortex-M4, ML-KEM-768 encapsulation takes 667 769 cycles, about half the cost of the two X25519 multiplications needed by a member. This shortens the member phase from 8.8 ms to 4.8 ms. Larger ciphertexts extend uplink time from 15.5 ms to 34.3 ms, but server decapsulation overlaps with transmission of later ciphertexts.

**Table 7.** Worst-case bounds and simulated time-to-key (ms) for  $N = 16$  at 50% load.

| Variant                 | WC-TTK             | TTK <sub>inst</sub> bound | Simulated mean | Simulated 99.9th pct. | Simulated max. | Synchronous max. |
|-------------------------|--------------------|---------------------------|----------------|-----------------------|----------------|------------------|
| VX (X25519)             | 119.4              | 90.8                      | 52.5           | 57.6                  | 58.8           | 69.9             |
| VB, ML-KEM-512          | 103.2              | 74.6                      | 38.8           | 44.2                  | 45.7           | 52.7             |
| VB, ML-KEM-768          | 122.7              | 94.1                      | 55.3           | 60.9                  | 62.1           | 71.7             |
| VB, ML-KEM-1024         | 148.6              | 120.0                     | 78.8           | 84.1                  | 85.8           | 94.6             |
| VH (hybrid)             | 168.0              | 139.4                     | 96.4           | 101.6                 | 102.4          | 114.5            |
| VA (per-ECU keys)       | 124.1              | 95.4                      | 63.9           | 68.6                  | 69.5           | 74.1             |
| VS1 (signed M1)         | 134.7              | 106.1                     | 67.0           | 72.8                  | 73.6           | 83.1             |
| VS2 (mutual signatures) | 337.3 <sup>a</sup> | 308.7 <sup>a</sup>        | 158.8          | 192.3                 | 216.8          | 223.2            |

<sup>a</sup> Computed with the largest signing time observed in 1000 pqm4 runs; ML-DSA signing has no deterministic bound (Section V-H).

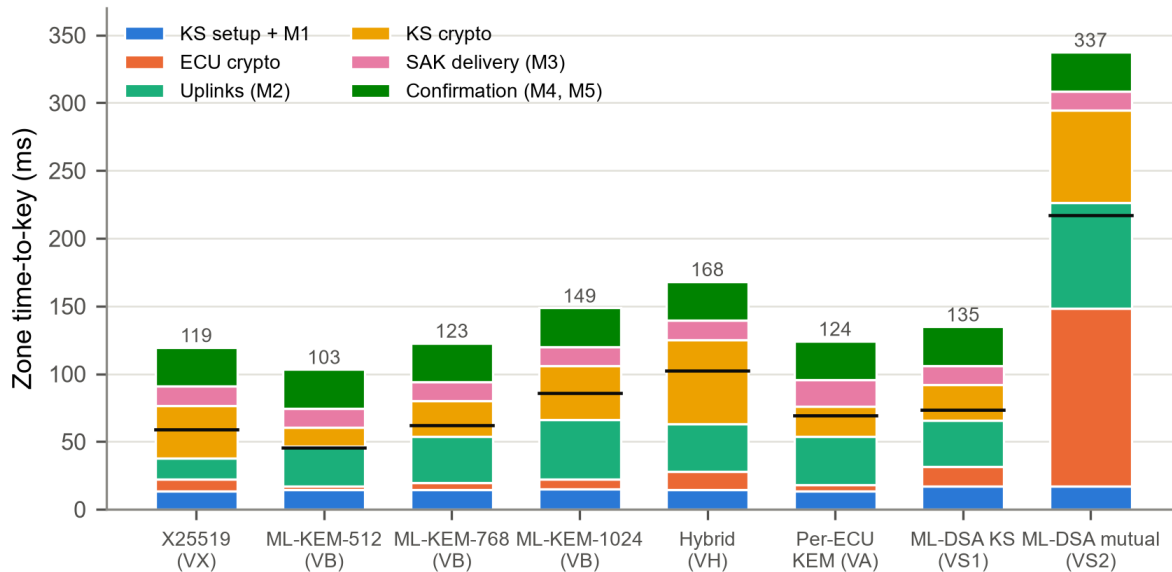


**Fig. 4.** Composition of the WC-TTK bound by protocol phase for  $N = 16$ . The black tick on each bar marks the maximum of 10 000 simulated runs.

Broadcast and acknowledgement rounds account for much of the bound. Together, M1, M3, M4 and M5 require 57.2 ms, or 47% of VB's bound. Each round faces a separate critical instant with interference from up to 106 control frames, whose arbitration phases run at the nominal bit rate. Consequently, each round costs about 13–15 ms in the worst case, largely independent of payload: the 1240-byte M1 takes only 0.8 ms more than the 24-byte M5. Server decapsulations remaining after the final ciphertext arrives add 26.1 ms. Moving from ML-KEM-512 to ML-KEM-1024 adds 45.4 ms, mainly through increased KS work. Hybrid VH adds 45.3 ms to VB because each decapsulation is accompanied by an X25519 operation. Per-ECU key pairs in VA add 1.4 ms: sequential server encapsulations replace decapsulations, and  $N$  large unicast replies replace a small multicast. Signed broadcast VS1 adds 12.0 ms for member verification of the ML-DSA-44 signature and the two-fragment M1. Mutual signatures in VS2 multiply the bound by 2.7.

### C. Zone size

Fig. 5 plots the bound against zone membership. Beyond a few members, all variants grow approximately linearly. Between  $N = 16$  and 64, the slopes are 3.08 ms per member for VB with ML-KEM-768, 2.75 ms for VX and 5.38 ms for VH. The larger of the server's per-member processing time and the ciphertext transmission time determines the slope: ML-KEM-768 decapsulation at 300 MHz takes 2.63 ms plus  $t_{\text{sym}}$ , versus 1.069 ms for a ciphertext frame. At 64 members, VB reaches 270.7 ms and the hybrid reaches 426.1 ms.

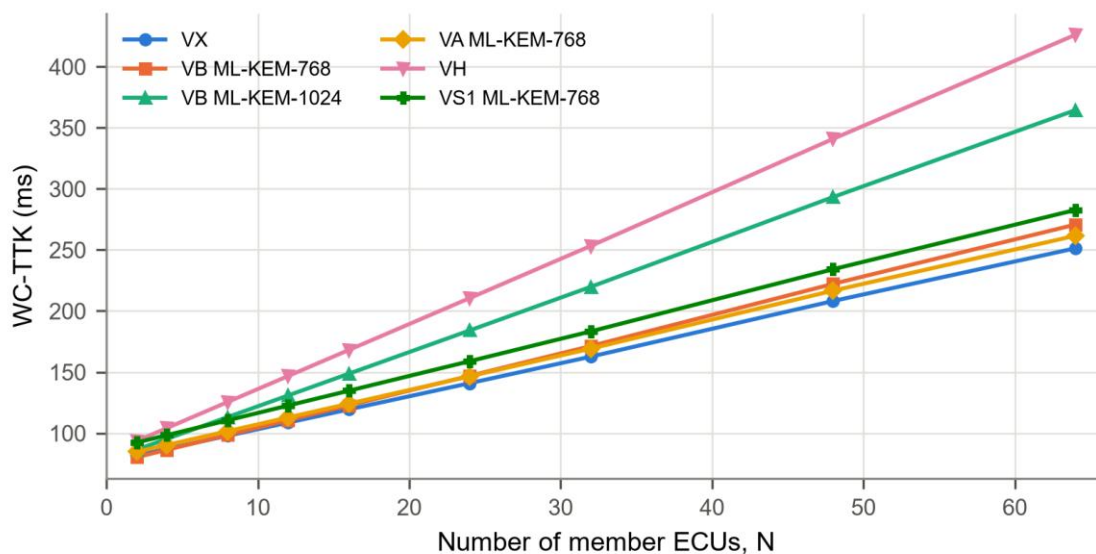


**Fig. 5.** WC-TTK versus zone size (0.5/10 Mbit/s, 50% load, KS at 300 MHz).

#### D. Processor performance

The server is the limiting factor in larger zones (Fig. 6a). With  $N = 32$ , increasing its clock from 100 MHz to 300 MHz and 1 GHz lowers the bound from 339.5 ms to 171.3 ms and 121.5 ms. Above about 962 MHz in Cortex-M4-equivalent performance, decapsulation is faster than ciphertext arrival; further processor speed gives little benefit because the bus limits uplink throughput. Member processors contribute much less (Fig. 6b), since encapsulations execute in parallel: the VB bound changes from 133.4 ms at 48 MHz to 120.4 ms at 320 MHz. X25519 places more work on members through two scalar multiplications and is slower than VB on 48 MHz ECUs, at 139.5 ms. Using reference C code instead of optimised assembly raises VB to 165.9 ms. A 30% memory wait-state penalty ( $\kappa = 1.3$ ) raises it to 136.7 ms, and halving the key-agreement CPU share ( $\alpha = 0.5$ ) raises it to 169.3 ms. Generating the KS key pair after the trigger adds only 2.4 ms, the key-generation duration on the 300 MHz server.

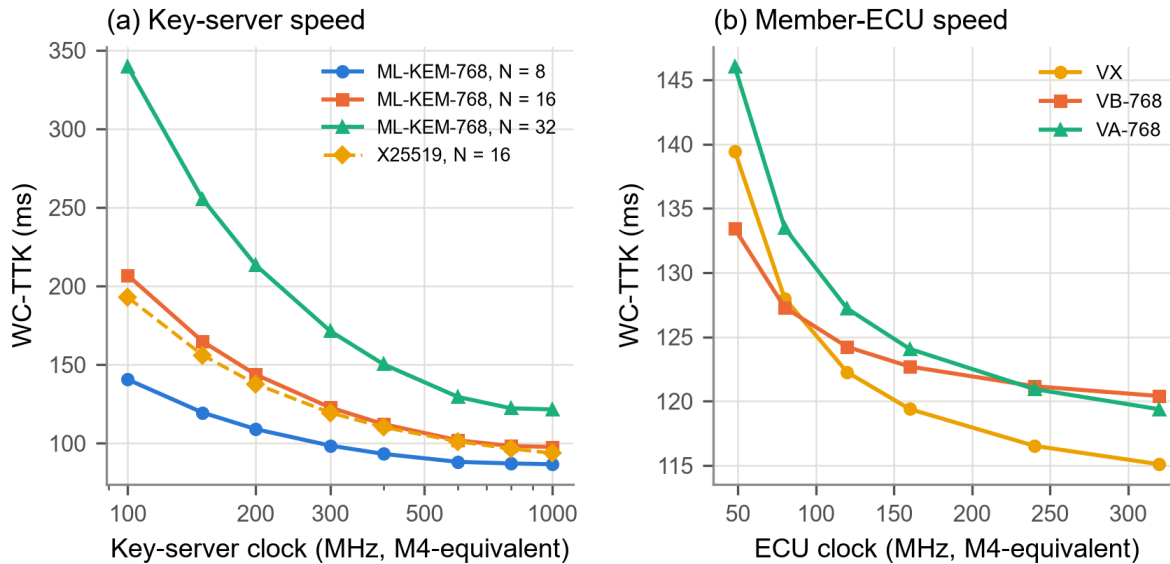


Fig. 6. WC-TTK versus (a) key-server and (b) member-ECU clock frequency.

#### E. Bit rates and CAN FD

Because arbitration of interfering frames dominates protocol rounds, the nominal bit rate matters more than the data-phase rate. At 0.5 Mbit/s nominal, increasing the data phase from 5 to 10 and 20 Mbit/s lowers VB's bound from 153.1 to 122.7 and 110.8 ms. Raising arbitration to 1 Mbit/s with a 10 Mbit/s data phase gives 97.9 ms. CAN FD (Fig. 7) segments each ML-KEM PDU into 18–20 ISO-TP frames, each competing independently for bus access. For  $N = 16$ , the bound reaches 203.1 ms at 0.5/5 Mbit/s and 363.5 ms at 0.5/2 Mbit/s; CAN XL therefore shortens it by 40% and 66%. The penalty increases with zone size: 32 members require 541.6 ms on CAN FD at 2 Mbit/s, versus 171.3 ms on CAN XL. X25519 remains substantially faster on CAN FD at 202.3 ms. Single-frame transport is thus what makes post-quantum agreement practical on CAN XL.

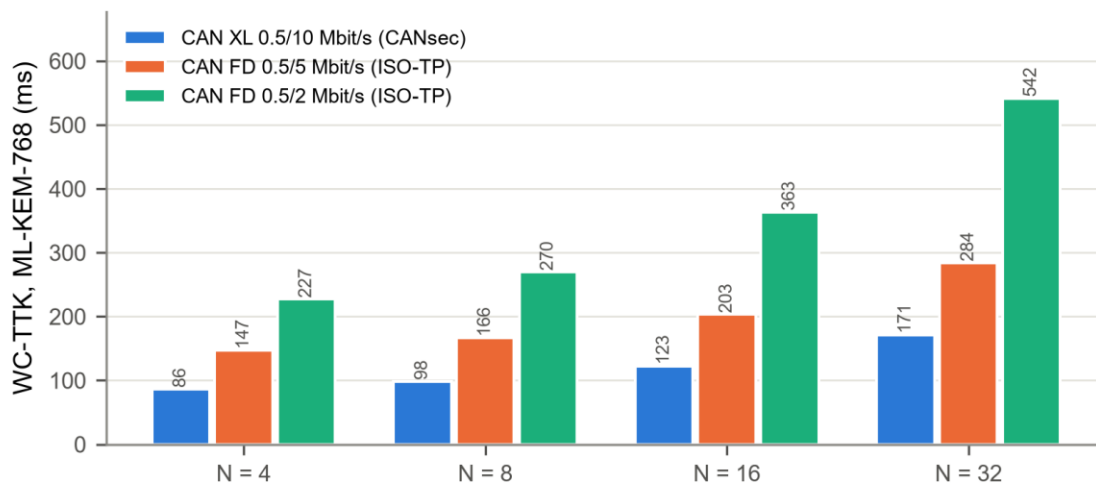


Fig. 7. WC-TTK of VB with ML-KEM-768 on CAN XL and on CAN FD with ISO-TP segmentation.

### F. Bus load and priority assignment

Priority assignment also determines key-agreement latency (Fig. 8). Between control and bulk traffic, the bound is independent of bulk load, remaining 122.7 ms from 30% to 80% utilisation. Bulk traffic can block a key-agreement frame only once per step, already covered by the 2048-byte blocking frame. Below the bulk band, the bound grows from 164.1 ms at 30% load to 345.5 ms at 80%. Above control traffic, it falls to 66.0 ms, but ciphertext bursts delay time-critical messages: 16 frames occupy the bus for 17.1 ms, increasing the worst-case response of 10 ms control messages by up to 20.83 ms, twice their period. Placement between control and bulk traffic avoids this cost because key-agreement frames are shorter than the 2048-byte frames already included in the control traffic's blocking term.

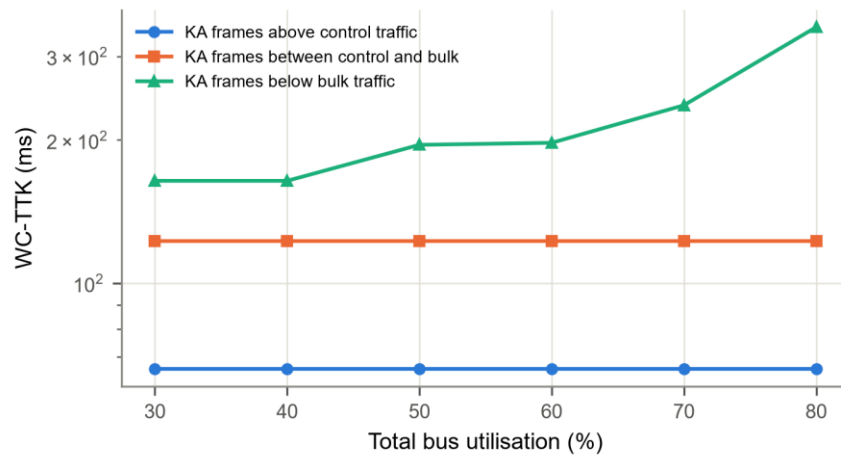


Fig. 8. WC-TTK versus total bus utilisation for three priority bands of the key-agreement frames (log scale).

### G. Transmission errors

Transmission errors have a modest effect on the bound. An initial burst of two errors followed by one error every 100, 50 or 20 ms raises VB from 122.7 ms to 133.5 ms; an error every 10 ms raises it to 142.6 ms. A retransmitted key-agreement frame lasts about 1 ms, short compared with the control-traffic interference that dominates each protocol round.

### H. Signature-based authentication

ML-DSA signing repeats its main loop until a candidate signature passes the rejection tests. Using the expected repetition count in FIPS 204—4.25 for ML-DSA-44 [12]—the iteration count  $k$  follows a geometric distribution with success probability  $p = 1/4.25$ , and signing time is  $c_{\text{fix}} + kc_{\text{it}}$ . Fitting the pqm4 average and minimum yields  $c_{\text{fix}} \approx 1\,156\,999$  and  $c_{\text{it}} \approx 655\,558$  cycles for ML-DSA-44. The largest of 1000 measured runs, 17 009 165 cycles, corresponds to about 24.2 iterations. Its exceedance probability is  $2.0\text{e-}03$  for a signing and  $3.1\text{e-}02$  for at least one of 16 members in a session. Fig. 9 shows the tail at 160 MHz: signing exceeds 220.3 ms with probability  $10^{-6}$  and 326.8 ms with probability  $10^{-9}$ . Inserting these probabilistic limits into the analysis gives VS2 time-to-key values of 440.6 ms and 547.2 ms, three to four times the KEM-only design. Member authentication using signatures therefore replaces a hard real-time bound with a probabilistic one. The KEM-and-PSK design remains deterministic under the execution-time assumptions because ML-KEM runs in constant time.

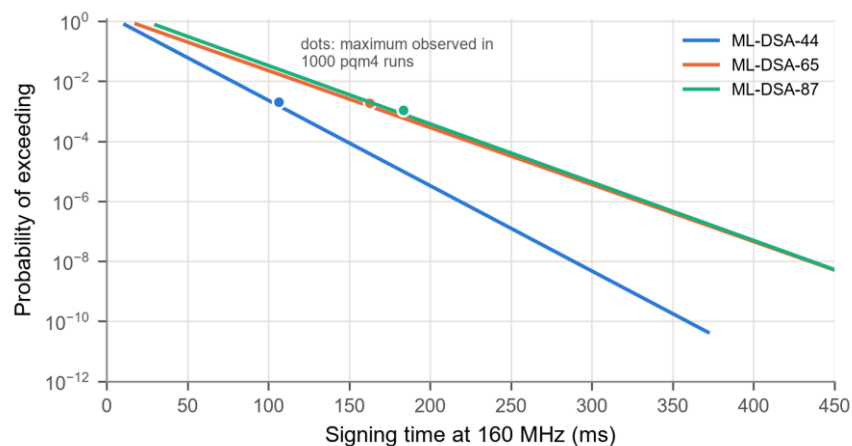


Fig. 9. Exceedance probability of the ML-DSA signing time at 160 MHz under the geometric iteration model; dots mark the largest time observed in 1000 pqm4 runs.

### 1. Tightness of the bound

Fig. 10 compares the simulated distributions with the analytical bounds; no simulated run exceeds its bound. With random phasing, VB's largest simulated time-to-key is 62.1 ms, giving a bound-to-maximum ratio of 1.97. Synchronous control-traffic release raises the largest value to 71.7 ms, with a ratio of 1.71. The difference reflects the compositional analysis, which assumes a new critical instant for every protocol round. Periodic traffic cannot realise all these instants within the same session. KEM-based variants produce narrow timing distributions, while VS2 has a long right tail reflecting the variability of signature generation.

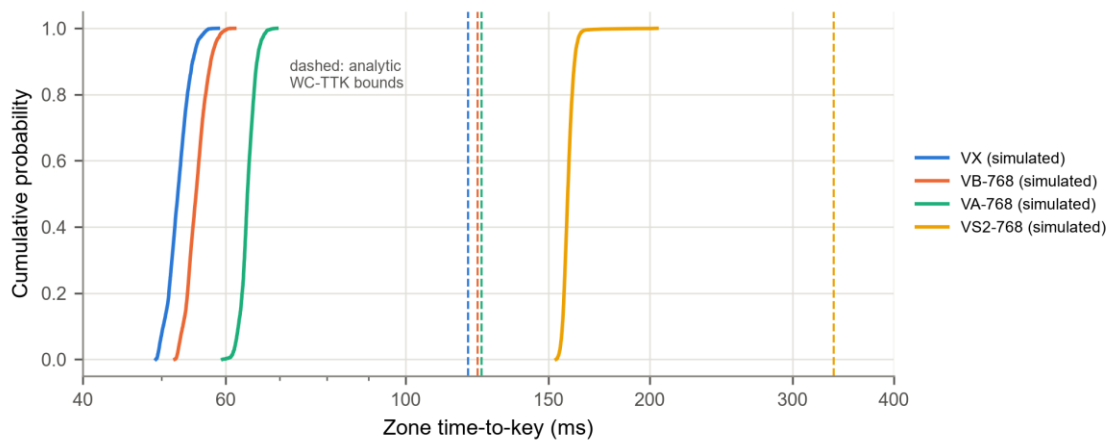


Fig. 10. Cumulative distributions of the simulated zone time-to-key (10 000 runs each) and the analytic bounds (dashed).

## VI. DISCUSSION

### A. Design guidelines

The analysis leads to six guidelines for CANsec key agreement on CAN XL:

- **Prefer KEM-plus-PSK authentication.** ML-KEM with pre-shared-key tags retains deterministic timing within a few percent of X25519. Member signatures with ML-DSA add hundreds of milliseconds and a probabilistic tail.
- **Broadcast one ephemeral key.** Broadcasting the KS encapsulation key and multicasting the SAK distribution minimise large frames and enable parallel member encapsulation.
- **Budget key-server performance.** At 300 MHz, WC-TTK grows by about 3.08 ms per member. A server that decapsulates faster than ciphertext frames arrive—about 962 MHz Cortex-M4-equivalent performance, or a hardware accelerator—ceases to be the bottleneck.
- **Reduce sequential rounds.** Each round costs 13–15 ms in the worst case at 0.5 Mbit/s arbitration. Combining acknowledgement and activation, or accepting both associations during change-over as MACsec does through association numbers, can shorten the bound more than changing the KEM.
- **Place key-agreement frames between control and bulk traffic.** This makes the bound independent of bulk load without affecting control-traffic response times.
- **Prefer CAN XL to CAN FD for post-quantum agreement.** Single-frame transport shortens the bound by 40–66% for 16 members and retains a small linear growth rate with zone size.

The results also permit direct checks against a start-up budget. A 150 ms budget, for example, accommodates about 24 members with VB and ML-KEM-768 at 300 MHz, compared with about 12 for hybrid VH. Mutual ML-DSA authentication cannot guarantee this budget.

### B. Security considerations

PQ-CKA protects the SAK with two independent secrets: an ephemeral ML-KEM secret supporting forward secrecy and quantum resistance, and a pre-shared key authenticating members and the KS. Recording traffic and later breaking X25519 does not expose the SAK, since the base protocol uses no classical key exchange. Later theft of  $PSK_i$  cannot recover past SAKs once  $dk$  and the shared secrets have been erased. Hybrid VH adds protection against an unexpected ML-KEM weakness at a measured latency cost of 45.3 ms for 16 members. A formal protocol proof, following approaches used for KEMTLS [24] or post-quantum WireGuard [25], remains future work.

### C. Limitations

Execution times use Cortex-M4 measurements scaled by clock frequency. Automotive processors, including multi-core controllers, differ in pipelines, caches and memory systems, so their behaviour will differ; operating-system interference is represented only by the CPU share  $\alpha$ . The frame model follows ISO 11898-1:2024 as described in public literature. CiA 613-2 remains a draft standard proposal and CiA 613-3 a working draft, so PDU formats may change. Control messages come from a single vehicle whose identifiers

were obfuscated in the published dataset; the priority order therefore reflects those obfuscated identifiers. The analysis includes random transmission errors but excludes denial-of-service attacks, against which CAN arbitration provides no protection.

## VII. CONCLUSION

CANsec provides a MACsec-like data plane for CAN XL while leaving key agreement open. This paper proposes PQ-CKA, based on an ephemeral ML-KEM encapsulation key broadcast by the KS and pre-shared-key authentication. Its compositional WC-TTK bound accounts for frame timing, arbitration, transmission errors and cryptographic execution. With traffic from a real vehicle, 16 members and a half-loaded 0.5/10 Mbit/s bus, zone time-to-key is bounded by 122.7 ms, with every member holding the SAK after 94.1 ms. This is only 2.8% above X25519 and 66% below the same protocol on CAN FD at 2 Mbit/s. Ten thousand simulated runs per variant remained below the bound. Sequential protocol rounds and server decapsulation rate are the main contributors to latency; ML-DSA member signatures introduce a probabilistic tail and make time-to-key three to four times longer. Under the model assumptions, CAN XL can support post-quantum key agreement with hard guarantees of 100–150 ms for zones of up to about two dozen members, provided the design uses KEMs, a capable key server and few sequential rounds.

## REFERENCES

- [1] *Road vehicles—Controller area network (CAN)—Part 1: Data link layer and physical coding sublayer*, ISO 11898-1:2024, 2024.
- [2] CAN in Automation, “CAN XL,” CiA, 2026. [Online]. Available: <https://www.can-cia.org/can-knowledge/can-xl> (accessed Sep. 27, 2026)
- [3] CAN in Automation, *CAN XL add-on services—Part 2: CANsec data plane*, CiA 613-2, DSP 1.0.0, 2026.
- [4] *IEEE Standard for Local and Metropolitan Area Networks—Media Access Control (MAC) Security*, IEEE Std 802.1AE-2018, 2018, doi: 10.1109/ieeestd.2018.8585421.
- [5] CAN in Automation, “CiA 613 series: CAN XL add-on services,” CiA, 2026. [Online]. Available: <https://www.can-cia.org/can-knowledge/cia-613-series-can-xl-add-on-services> (accessed Sep. 27, 2026)
- [6] F. Wiemer, A. Mutter, J. Ndop, J. Göppert, A. Sikora, and T. Walrant, “Beyond Ethernet: Reusing MACsec for CANsec,” *IACR Cryptology ePrint Archive*, Paper 2025/2224, 2025. [Online]. Available: <https://eprint.iacr.org/2025/2224>
- [7] P. W. Shor, “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer,” *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, 1997, doi: 10.1137/s0097539795293172.
- [8] C. Gidney and M. Ekerå, “How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits,” *Quantum*, vol. 5, Art. no. 433, 2021, doi: 10.22331/q-2021-04-15-433.
- [9] C. Gidney, “How to factor 2048 bit RSA integers with less than a million noisy qubits,” *arXiv*, arXiv:2505.15917, 2025. [Online]. Available: <https://arxiv.org/abs/2505.15917>
- [10] M. Mosca, “Cybersecurity in an Era with Quantum Computers: Will We Be Ready?,” *IEEE Security & Privacy*, vol. 16, no. 5, pp. 38–41, 2018, doi: 10.1109/msp.2018.3761723.
- [11] National Institute of Standards and Technology, “Module-lattice-based key-encapsulation mechanism standard,” NIST, Gaithersburg, MD, USA, FIPS 203, 2024, doi: 10.6028/nist.fips.203.
- [12] National Institute of Standards and Technology, “Module-lattice-based digital signature standard,” NIST, Gaithersburg, MD, USA, FIPS 204, 2024, doi: 10.6028/nist.fips.204.
- [13] G. Alagic et al., “Recommendations for key-encapsulation mechanisms,” NIST, Gaithersburg, MD, USA, NIST SP 800-227, 2025, doi: 10.6028/nist.sp.800-227.
- [14] A. Regenscheid, “Transition to Post-Quantum Cryptography Standards,” NIST, Gaithersburg, MD, USA, NIST IR 8547 (initial public draft), 2024, doi: 10.6028/nist.ir.8547.ipd.
- [15] UNECE, *UN Regulation No. 155—Uniform provisions concerning the approval of vehicles with regards to cyber security and cyber security management system*, E/ECE/TRANS/505/Rev.3/Add.154, 2021.
- [16] *Road vehicles—Cybersecurity engineering*, ISO/SAE 21434:2021, 2021.
- [17] R. Yasmin, H. J. Hadi, M. K. Khan, N. Ahmad, and M. A. Alshara, “Evaluation of Hybrid CAN XL–Ethernet Architectures for Next-Generation Automotive Networks,” *IEEE Open Journal of Vehicular Technology*, vol. 7, pp. 2055–2073, 2026, doi: 10.1109/ojvt.2026.3709660.
- [18] AUTOSAR, *Specification of Secure Onboard Communication Protocol*, AUTOSAR FO R20-11, 2020.
- [19] CAN in Automation, *CAN XL add-on services—Part 3: LLC frame fragmentation*, CiA 613-3, WD 0.0.11, 2025.
- [20] *IEEE Standard for Local and Metropolitan Area Networks—Port-Based Network Access Control*, IEEE Std 802.1X-2020, 2020, doi: 10.1109/ieeestd.2020.9018454.
- [21] J. S. Buruaga, A. Bugler, J. P. Brito, V. Martin, and C. Striecks, “Versatile quantum-safe hybrid key exchange and its application to MACsec,” *EPJ Quantum Technology*, vol. 12, no. 1, Art. no. 84, 2025, doi: 10.1140/epjqt/s40507-025-00382-x.
- [22] J. Bos et al., “CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM,” in *Proc. IEEE European Symposium on Security and Privacy (EuroS&P)*, 2018, pp. 353–367, doi: 10.1109/eurosp.2018.00032.
- [23] A. Langle, M. Hamburg, and S. Turner, “Elliptic Curves for Security,” Internet Engineering Task Force, RFC 7748, 2016, doi: 10.17487/rfc7748.
- [24] P. Schwabe, D. Stebila, and T. Wiggers, “Post-Quantum TLS Without Handshake Signatures,” in *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2020, pp. 1461–1480, doi: 10.1145/3372297.3423350.
- [25] A. Hülsing, K.-C. Ning, P. Schwabe, F. J. Weber, and P. R. Zimmermann, “Post-quantum WireGuard,” in *Proc. IEEE Symposium on Security and Privacy (SP)*, 2021, pp. 304–321, doi: 10.1109/sp40001.2021.00030.
- [26] M. Barbosa et al., “X-Wing: The hybrid KEM you’ve been looking for,” *IACR Communications in Cryptology*, vol. 1, no. 1, pp. cc1–1–52, 2024, doi: 10.62056/a3qj89n4e.
- [27] M. J. Kannwischer, J. Rijneveld, P. Schwabe, and K. Stoffelen, “pqm4: Testing and benchmarking NIST PQC on ARM Cortex-M4,” *IACR Cryptology ePrint Archive*, Paper 2019/844, 2019. [Online]. Available: <https://eprint.iacr.org/2019/844>
- [28] pqm4 contributors, “pqm4: Post-quantum crypto library for the ARM Cortex-M4 — benchmarks,” GitHub, 2025. [Online]. Available: <https://github.com/mupq/pqm4/blob/master/benchmarks.md> (accessed Sep. 27, 2026)

- [29] B. Haase and B. Labrique, "AuCPace: Efficient verifier-based PAKE protocol tailored for the IIoT," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 1–48, 2019, doi: 10.46586/tches.v2019.i2.1-48.
- [30] T. Fritzmann, J. Vith, D. Flórez, and J. Sepúlveda, "Post-quantum cryptography for automotive systems," *Microprocessors and Microsystems*, vol. 87, Art. no. 104379, 2021, doi: 10.1016/j.micpro.2021.104379.
- [31] A. Castiglione and T. Elia, "Securing in-vehicle communications through post-quantum cryptography and network segmentation," *Computers and Electrical Engineering*, vol. 126, Art. no. 110488, 2025, doi: 10.1016/j.compeleceng.2025.110488.
- [32] A. Lotto, F. Marchiori, A. Brighente, and M. Conti, "A Survey and Comparative Analysis of Security Properties of CAN Authentication Protocols," *IEEE Communications Surveys & Tutorials*, vol. 27, no. 4, pp. 2470–2504, 2025, doi: 10.1109/comst.2024.3486367.
- [33] G. Cena, L. Seno, and S. Scanzio, "Robust multicast origin authentication in MACsec and CANsec for automotive scenarios," *arXiv*, arXiv:2502.20555, 2025. [Online]. Available: <https://arxiv.org/abs/2502.20555>
- [34] K. Tindell, A. Burns, and A. Wellings, "Calculating controller area network (CAN) message response times," *Control Engineering Practice*, vol. 3, no. 8, pp. 1163–1169, 1995, doi: 10.1016/0967-0661(95)00112-8.
- [35] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, "Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised," *Real-Time Systems*, vol. 35, no. 3, pp. 239–272, 2007, doi: 10.1007/s11241-007-9012-7.
- [36] S. Punnekkat, H. Hansson, and C. Norstrom, "Response time analysis under errors for CAN," in *Proc. 6th IEEE Real-Time Technology and Applications Symposium (RTAS)*, 2000, pp. 258–265, doi: 10.1109/rtas.2000.852470.
- [37] I. Broster, A. Burns, and G. Rodríguez-Navas, "Timing Analysis of Real-Time Communication Under Electromagnetic Interference," *Real-Time Systems*, vol. 30, no. 1-2, pp. 55–81, 2005, doi: 10.1007/s11241-005-0504-z.
- [38] O. Ikumapayi, H. Olufowobi, J. Daily, T. Hu, I. C. Bertolotti, and G. Bloom, "CANASTA: Controller Area Network Authentication Schedulability Timing Analysis," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 8, pp. 10024–10036, 2023, doi: 10.1109/tvt.2023.3258746.
- [39] A. Berisa et al., "Comparative Evaluation of Various Generations of Controller Area Network Based on Timing Analysis," in *Proc. IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2023, pp. 1–8, doi: 10.1109/etfa54631.2023.10275549.
- [40] J. Laufenberg, T. Kropf, and O. Bringmann, "CAN Simulation Framework - From Classic CAN to CAN XL," in *Proc. IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, 2023, pp. 3343–3348, doi: 10.1109/itsc57777.2023.10422342.
- [41] J. Schaad and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm," Internet Engineering Task Force, RFC 3394, 2002, doi: 10.17487/rfc3394.
- [42] H. Krawczyk and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)," Internet Engineering Task Force, RFC 5869, 2010, doi: 10.17487/rfc5869.
- [43] E. Barker, L. Chen, and R. Davis, "Recommendation for Key-Derivation Methods in Key-Establishment Schemes," NIST, Gaithersburg, MD, USA, NIST SP 800-56C Rev. 2, 2020, doi: 10.6028/nist.sp.800-56c2.
- [44] *Road vehicles—Diagnostic communication over Controller Area Network (DoCAN)—Part 2: Transport protocol and network layer services*, ISO 15765-2:2016, 2016.
- [45] M. E. Verma et al., "A comprehensive guide to CAN IDS data and introduction of the ROAD dataset," *PLOS ONE*, vol. 19, no. 1, Art. no. e0296879, 2024, doi: 10.1371/journal.pone.0296879.
- [46] Oak Ridge National Laboratory, "Real ORNL Automotive Dynamometer (ROAD) CAN Intrusion Dataset," Zenodo, 2020. [Online]. Available: <https://zenodo.org/records/10462796> (accessed Sep. 27, 2026)

#### AUTHORS

**First Author** – Omar Al-Daihani, B.Sc. in Computer Engineering, Computer Engineer, Kuwait, [aldaihaniali@gmail.com](mailto:aldaihaniali@gmail.com).

**Correspondence Author** – Omar Al-Daihani, [aldaihaniali@gmail.com](mailto:aldaihaniali@gmail.com), +965 99220884.