

Identify Post-Quantum Cryptographic Algorithms for Automotive Security: Performance-Driven Guidance for Secure Boot, OTA, and V2X Communication

Dileep Singh Kushwaha
dileep-singh.kushwaha@thalesgroup.com
Cybersecurity Managing Consultant, CPS
Thales DIS Technology India Private Limited
Noida, India

Parneet Kaur
parneet.kaur@thalesgroup.com
Cybersecurity Analyst, CPS
Thales DIS Technology India Private Limited
Noida, India

Abstract

The rapid expansion of quantum computing threatens the security foundations of classical public-key cryptography, including RSA and ECC, both vulnerable to Shor's algorithm. For the automotive industry, where secure communication underpins critical functions such as over-the-air (OTA) updates, secure boot, firmware signing, PKI validation, and vehicle-to-everything (V2X) connectivity, this threat demands early and informed migration planning. This paper presents a cross-platform performance evaluation of NIST-selected and candidate Post-Quantum Cryptography (PQC) algorithms integrated into the Transport Layer Security (TLS) protocol, in both standalone and hybrid (classical + PQC) configurations. Signature schemes CROSS, MAYO, SPHINCS+ (SHA and SHAKE variants), ML-DSA, and FALCON are evaluated alongside key encapsulation mechanisms including FrodoKEM, BIKE, and ML-KEM. Testing spans two representative platforms: a resource-constrained Raspberry Pi 4B (Broadcom BCM2711, 4GB RAM) representing embedded automotive ECUs, and an Ubuntu-based Intel Core i5 system representing backend or gateway infrastructure. Execution time, memory consumption, and CPU core utilization are measured across algorithm-KEM combinations to characterize real-world feasibility. Results indicate lattice-based schemes offer a strong balance of performance, interoperability, and long-term assurance suited to V2X and PKI-heavy communication, while compact signature schemes such as MAYO and FALCON better serve resource-constrained operations like secure boot and firmware signing, where verification speed and low memory footprint are prioritized over frequent key exchange. The study further identifies a critical dependency of PQC feasibility on 64-bit architecture, highlighting the limitations of legacy 32-bit automotive chipsets that lack dedicated cryptographic accelerators. These findings offer practical, use-case-specific guidance for automotive OEMs and suppliers navigating the transition from classical to quantum-resistant cryptography, supporting risk-informed decisions across onboard and offboard vehicle communication architectures.

I. Introduction

The gradual evolution of quantum computing brings revolutionary potential along with enormous risks, especially regarding data security. Conventional cryptographic algorithms, like RSA, ECC, which attain security from the computational intractability of mathematical problems (prime factorization and discrete logarithm), insoluble for classical computers, subsequently taking exponential time to break, hence providing confidentiality and integrity of data in digital systems. Yet, quantum algorithms i.e., Shor's algorithm in particular can do these calculations in polynomial time, which makes

public-key cryptosystems widely in use insecure. The possibility of quantum attacks necessitates a change in paradigm in cryptographic culture to guarantee long-term data security. This has challenged worldwide standardization organizations such as NIST, to start the process of developing and testing quantum-proof cryptographic algorithms under its PQC standardization efforts. Algorithms like Crystals-Kyber, Crystals-Dilithium, SPHINCS+, Falcon are the leading contenders for replacing traditional public-key schemes. These algorithms are secure against both classical and quantum attacks. Automotive cybersecurity engineering is governed by ISO/SAE 21434 [21], with regulatory enforcement of connected-vehicle cybersecurity and software update integrity under UNECE WP.29 Regulations No. 155 [22] and No. 156 [23], underscoring the urgency of migrating vehicle communication security to quantum-resistant schemes.

This paper offers a wide-ranging assessment of the identified PQC algorithms on cross-platforms, with emphasis on their deployment in client-server communication through the Transport Layer Security (TLS) protocol. The research seeks to determine the most appropriate PQC algorithm in performance, interoperability, and implementation limitations, and providing useful guidance for migration to quantum-resistant communications. Since computational efficiency is such a key factor in the practical deployment of PQC in the real world, our analysis sheds light upon their feasibility for deployment.

II. Methodology

A. Background

With transition to quantum computers, the need to migrate from traditional algorithms to PQC algorithms is imminent. PQC algorithms have been under study since the mid-1990s, ever since Shor's algorithm was introduced, which broke the underlying mathematical hardness of traditional algorithms like RSA and ECC compromising the security for TLS, as problems like large prime factorization and discrete logarithms were solved in polynomial time. The need to find a new mathematical hardness base for cryptosystems arose, concerned stakeholders started actively pursuing and supporting research for algorithms that would prove to be resistant to quantum computing. The Open Quantum Safe (OQS) project initiated open-source liboqs platform providing C library for PQC algorithms offering API for key encapsulation mechanisms and digital signatures for the integration of quantum-resistant algorithms in various applications and platforms.

B. Testing Environment

To evaluate the proposed integration of quantum-resistant algorithms, the testing environment consists of following hardware and software specifications. On Hardware, the PQC algorithms were integrated and were deployed on following platforms:

Platform	Virtual Machine	Raspberry Pi 4B
CPU	Intel Core i5	Broadcom BCM2711
RAM	4 GB	4 GB
Operating System	Ubuntu 64-bit	Raspberry Pi OS 64-bit

Table 1: Testing Environment details.

III. Chosen Signing Schemes & KEMs Literature review

A. Cross

CROSS is a signature scheme that relies on the fact that Restricted Syndrome Decoding Problem (RSDP), being an NP-Complete problem, cannot be efficiently solved. RSDP is a variant of classical SDP where the solutions are constrained to a predefined subset of the finite field [1]. Formally, given a parity-check matrix

$\mathbb{H} \in \mathbb{F}_2^{n \times m}$, a syndrome $\mathbb{S} \in \mathbb{F}_2^n$, a weight bound $w \in \mathbb{N}$, and an index subset $\mathbb{I} \subseteq \{1, \dots, m\}$, the problem asks whether there exists an error vector $\mathbb{e} \in \mathbb{F}_2^m$ such that

$$\mathbb{H} \cdot \mathbb{e}^T = \mathbb{S}$$

where $wt(\mathbb{e}) \leq w$, and $supp(\mathbb{e}) \subseteq \mathbb{I}$. RSDP is used to form interactive Zero Knowledge (ZK) Identification Protocols that require active interaction between a *Prover* (P) and a *Verifier* (V) using a Commitment-Challenge-Response mechanism. Most Zero Knowledge proofs tend to follow the same three step structure:

Commitment: The Prover creates a commitment to a secret value (witness). The commitment is shared with Verifier, this commitment alone cannot be used to deduce the secret

Challenge: The Verifier after receiving the commitment generates a random challenge that dictates the form of information the prover must disclose in the next step.

Response: The Prover accepts the question, calculates the answer, and returns it to verifier. This allows Verifier to check if the Prover has access to witness. To verify that the prover has access to witness, the verifier asks multiple questions. Repeating this interaction reduces the possibility of the prover faking knowledge drops significantly [2].

But interactive ZK identification protocols require multiple interactions between prover and verifier and thus cannot be used for digital signatures. For that purpose, we apply Fiat-Shamir Transform over ZK protocols making them non-interactive and suitable for usage in digital signature, not to mention the increased efficiency. The Fiat-Shamir Transform [1] removes interaction in classical challenge-response commitment by substituting verifier's challenge with the result of cryptographic hash function:

Commitment: This step remains the same, a commitment is generated over a secret value (witness).

Challenge: Rather than receiving a random challenge from the verifier, the prover calculates the challenge deterministically as $c = \text{HASH}(m, com)$

where m is the message to be proven, and com is the commitment.

Response: The prover calculates the response from the challenge and the initial randomness. The non-interactive proof resulting from it is the commitment and response, which the verifier can check independently by recomputing the challenge from the hash function.

The security of CROSS is based on the hardness of the Restricted Syndrome Decoding Problem (RSDP), an NP-complete problem that is thought to be resistant to quantum attacks. It utilizes a zero-knowledge identification protocol that functions in a commitment–challenge–response mechanism and is based on code-based commitments and syndrome equations. The interactive identification scheme is then transformed into a digital signature scheme by applying the Fiat–Shamir transform, which makes the protocol non-interactive. CROSS provides post-quantum security with effective verification and small signature sizes by eschewing number-theoretic assumptions and depending only on problems that are challenging quantum algorithms.

B. FrodoKEM

FrodoKEM is a Key Encapsulation Mechanism that works on the principle of *Learning With Errors* (LWE) problem. In 2005 Regev proved the hardness of the LWE assuming the hardness of various worst-case lattice-based problems against quantum-algorithms and established a public-key encryption scheme whose IND-CPA security is related to the hardness of LWE. Much more research has been conducted following Regev's work on LWE that provided additional theoretical support for hardness of LWE problem along with extensive analysis of LWE. This led to the formation of a public-key encryption (PKE) scheme FrodoPKE that served as building block for FrodoKEM.

The LWE-based PKE scheme, proposed by Lindner and Peikert, uses an efficient square public matrix A that is pseudo randomly generated from a small seed [3].

$$A \in \mathbb{Z}_q^{n \times n}$$

This matrix is used to generate a key pair using the following equation [4].

$$B = A \cdot s + e(\text{mod } q)$$

Where A and B are publicly known, and s is a secret matrix with entries drawn from a small-error distribution. The matrix e is an error matrix also drawn from a small-error distribution. The hardness of FRODOPKE comes from the difficulty of recovering s given A and B .

To make use of FRODOPKE as a KEM algorithm, it needs to be IND-CCA secure. This is where Fujisaki-Okamoto (FO) Transform comes into picture. FO transform is used to convert an IND-CPA secure PKE into IND-CCA secure KEM by binding encryption key and ciphertext together using hash functions. After the public keys (A , B), secret key (s) and a random seed (r) are generated we can use them to encapsulate the session key (K) by:

$$(s', e', e'') = \text{GenMatrices}(\mathbb{H}_1(r, pk))$$

$$B' = A \cdot s' + e'(\text{mod } q) \quad V = B \cdot s' + e''(\text{mod } q)$$

Then we encode the random seed r into a matrix M and compute ciphertext [4]:

$$C = V + M(\text{mod } q) \quad ct = (B', C)$$

And to derive the session key we use: $\mathbb{K} = \mathbb{H}_2(r, ct)$

To decapsulate the session key

$$V' = B' \cdot s(\text{mod } q) \quad M' = C - V'(\text{mod } q)$$

$$r' = \text{Decode}(M')$$

$$(s', e', e'') = \text{GenMatrices}(\mathbb{H}_1(r', pk))$$

We use the above computed matrices to recompute B and C

$$B'' = A \cdot s' + e'(\text{mod } q) \quad C'' = B \cdot s' + e''(\text{mod } q)$$

We compare the recomputed values of B'' and C'' with received B' and C and if they do not match, we return an error due to integrity breach [5].

To derive the session key (K) we use: $\mathbb{K} = \mathbb{H}_2(r', ct)$

The FrodoKEM algorithm is designed for IND-CCA security at three levels; FrodoKEM-640 for NIST level 1; FrodoKEM-976 for NIST level 3; and FrodoKEM-1344 for NIST level 5. Furthermore, the FrodoKEM algorithm can be distinguished into 2 more categories based on the pseudorandom number generator used for generation of matrix A :

- AES-128 is used in devices that support AES acceleration.
- SHAKE-128 is used in devices that do not support AES acceleration.

In conclusion, FrodoKEM stands as a strong candidate for post-quantum cryptographic applications, offering robust security grounded in well-studied lattice problems. By integrating the Fujisaki–Okamoto transform, it ensures resistance against adaptive attacks, while its flexible design—supporting multiple security levels and hardware-friendly variants—makes it a practical and forward-looking solution for securing data in the quantum era.

C. BIKE

Bit Flipping Key Encapsulation (BIKE) is a key encapsulation mechanism (KEM) that, much like classic McEliece relies on the hardness of the Syndrome Decoding Problem (SDP) but on a different linear code family. McEliece makes use of Goppa polynomial, randomly generated over a finite field, to create a generator matrix (G) that is in turn used to transform the plaintext message into a codeword (C). The matrix C is then scrambled with a non-singular matrix $S^{k \times k}$ and a permutation matrix $P^{n \times n}$ in order to disguise G . This results in a very large public key $pk = (G_{pub}, t)$ where $G_{pub} = S \cdot G \cdot P$ and t is degree of the Goppa polynomial. BIKE improves on this by making use of quasi-cyclic [6] moderate-density parity-check (QC-MDPC) codes instead of the Goppa polynomial reducing the public key size dramatically. By using a moderate-density parity-check (MDPC) matrix instead of a low-density parity-check (LDPC) matrix BIKE ensures security alongside

of performance and by making use of quasi-cyclic version of the said MDPC the entire parity check matrix can be described only by its first row, allowing for significant reduction in key sizes. To generate keys [7], we randomly choose two binary circulant matrices $H_0, H_1 \in \mathbb{F}_2^{r \times r}$ where r is the block size, and each row has hamming weight ω , these two matrices serve as the secret key. We then define a parity-check matrix and public key like so

$$H = [H_0 | H_1] \in \mathbb{F}_2^{r \times 2r} \quad pk = h = H_1 H_0^{-1} \in \mathbb{F}_2^r$$

To encapsulate a shared secret using BIKE we generate a uniformly random bitstring m and derive an error vector

$$e = (e_0, e_1) = G(m) \in \mathbb{F}_2^{2r} \quad \text{with } wt(e) = \omega$$

The above generated public key structure is used to generate ciphertext and derive the shared key using the same.

$$c = H_0 e_0^T + H_1 e_1^T \in \mathbb{F}_2^r \quad K = \text{KDF}(m || c)$$

The output of the encapsulation is the pair (c, K) . To decapsulate the shared key from the ciphertext we make use of the already known secret key to calculate the error vector, this is made possible with bit-flipping decoding algorithm that efficiently decodes the error vector if the structure is known. We use the generated error vector to recover the message m by using the inverse of the hashing function $m = G^{-1}(e)$. Now that we have m and c we can simply compute the shared key $K = \text{KDF}(m || c)$. If decoding fails it implies an invalid ciphertext, in which case BIKE uses a FO transform and outputs a random key making the cryptosystem IND-CCA secure.

There are, currently, three different variants of BIKE algorithm, they are described in Table 2 below.

BIKE	NIST Level	Public Key Size	Secret Key Size	Ciphertext Size	Shared Secret Size
BIKE1	1	1541	5223	1573	32
BIKE3	3	3083	10105	3115	32
BIKE5	5	16494	16494	5154	32

Table 2: Different variants of BIKE Algorithm

D. MAYO

MAYO is a signing algorithm based on multivariate quadratic (MQ) polynomials, which is an NP-Hard problem, even for quantum computers, making it suitable as a post quantum crypto signature scheme. MAYO builds upon Unstructured Oil and Vinegar (UOV) [8] signing algorithm by introducing algebraic and structural modifications over a finite field converting it into a non-field ring structure allowing for smaller keys and faster signing [9]. MAYO uses a more structured algebraic system known as overring instead of finite fields, it can be represented as

$$R = \frac{\mathbb{F}_q[\theta]}{\theta^d}, \text{ where } d > 1$$

All the operations that were performed on a finite field i.e. $\mathbb{F}_q$ is replaced by an overring i.e. $\mathbf{R}$. Now we define a system of m quadratic equations each with n variables.

$$P_i(x) = \sum_{j \leq k}^n a_{i,j,k} x_j x_k + \sum_j^n b_{i,j} x_j + c_i \in \mathbf{R}, \text{ where } i = 1 \dots m$$

The above quadratic system represents a nonlinear map from $\mathbf{R}^n \rightarrow \mathbf{R}$ [10]. To generate a key from this system we use the oil-vinegar approach and define a private polynomial map $\mathbf{F}: \mathbf{R}^n \rightarrow \mathbf{R}^o$, where o is the number of oil variables and define two linear transformations over the ring $\mathbf{R}$

$$\mathbf{S} \in \mathbf{GL}_n(\mathbf{R}); \mathbf{T} \in \mathbf{GL}_o(\mathbf{R})$$

$\mathbf{S}$ & $\mathbf{T}$ are two invertible linear maps used to define the public key $pk(x) = \mathbf{T} \cdot \mathbf{F} \cdot \mathbf{S}(x)$ and the expanded version of the same acts as the private key i.e. $sk(x) = (\mathbf{F}, \mathbf{T}, \mathbf{S})$.

To generate a signature, we must first hash the message $y = H(m) \in \mathbb{F}_q^o$; $m \in \{0,1\}^q$ and then compute an invertible public map over the same

$$x' = \mathbf{F}(\mathbf{S}(x)) = \mathbf{T}^{-1}(y)$$

This entire system is then used to compute an invertible linear map in n dimension by $\sigma = \mathbf{S}^{-1}(x')$

The signature can then be verified by simply decoding it with the public map i.e.

$$y' = pk(\sigma)$$

and comparing the calculated value with hash of message $y' = H(m)$. Currently there are three variants of MAYO, they are described in table 3 below.

MAYO	NIST Level	Public Key Size	Private Key Size	Signature Size
MAYO-1	1	1168	24	321
MAYO-3	3	2656	32	577
MAYO-5	5	5008	40	838

Table 3: Different Variants of MAYO

E. SPHINCS+

Stateless Practical Hash-based Incredibly Nice Cryptographic Signature, otherwise known as SPHINCS, is a PQC signature scheme built solely on hash functions, making it quantum-resistant under Grover's algorithm [11]. Hash-based signature schemes have been around since Lamport signatures in 1979 and Merkle Trees in 1989, but all these signature schemes were stateful meaning that the signer had to maintain the state securely and only allowed for a limited number of signatures before the secret key had to be replaced. This makes them impractical for usage as the signer will need to maintain a

record of which part of the private key has been used to sign and limited number of signatures make it harder to scale these stateful signature schemes.

This is where SPHINCS come in with its statelessness and hierarchical structure that allows us to sign an arbitrarily large number of signatures without maintaining the secret state. SPHINCS uses a layered, tree-based structure combining multiple cryptographic approaches like:

- One-Time Signatures (OTS): It uses Winternitz One-Time Signature (WOTS+), an approach that is efficient and hash based.
- Merkle Trees: used to aggregate many OTS public keys and compress them in one single root, the public key.
- Hypertree: A stack of Merkle Trees used to reduce signature size.
- Hash Functions: it only uses hash functions like SHA-2, SHA-3 or Haraka, making it a minimal assumption construction.

To sign a message $\mathbf{M}$ SPHINCS require a public key $\mathbf{pk} = (\mathbf{root}, \mathbf{pk}_{seed})$ and a secret key $\mathbf{sk} = (\mathbf{sk}_{seed}, \mathbf{sk}_{prf}, \mathbf{pk})$ [12]. To prevent multi-target attacks, SPHINCS uses a randomness function to derive a random value for each signature $\mathbf{R} = \mathbf{PRF}(\mathbf{sk}_{prf}, \mathbf{M})$, then to compute the message digest and derive the input to Hash to Obtain Random Subset Tree (HORST) and the index of subtree and leaf in hypertree, used to pick WOTS+ keys, we hash the message $\mathbf{M}$, randomizer $\mathbf{R}$, and public key $\mathbf{pk}$.

$$\mathbf{m}_{hash} = H_{msg}(\mathbf{R}, \mathbf{pk}, \mathbf{M}) \rightarrow (\mathbf{m}_{HORST}, \mathbf{idx}_{tree}, \mathbf{idx}_{leaf})$$

Then to sign the message digest using HORST we first generate a list of t secret values from $\mathbf{sk}_{seed}$ deterministically using a pseudorandom generator and hash each of these values to build a Merkle Tree; $\mathbf{pk}_i = H(\mathbf{s}_i)$. The signing procedure uses the bits of $\mathbf{m}_{HORST}$ to select k indices from $[0, t-1]$ and includes the authentication path in the HORST Merkle tree up to the root. This constitutes HORST signature $\mathbf{sig}_{HORST}$ and the root is used as the message for WOTS+ signature at the bottom of the hypertree. To traverse the hypertree of height $\mathbf{h}' = \frac{\mathbf{h}}{\mathbf{d}}$, for each layer from 0 to $\mathbf{d}-1$ we sign the root from previous level using WOTS+ then add the Merkle authentication path from WOTS+ public key to the root of current subtree and compute and update the new tree and leaf indices for the next level. The final signature structure will be like

$$\sigma = (\mathbf{R}, \mathbf{sig}_{HORST}, \{\mathbf{sig}_{WOTS+}^{(i)}, \mathbf{auth}_{path}^{(i)}\}_{i=0}^{\mathbf{d}-1})$$

To verify the signature, we again compute $\mathbf{m}_{hash} = H_{msg}(\mathbf{R}, \mathbf{pk}, \mathbf{M})$, use $\mathbf{sig}_{HORST}$ to compute HORST root and then recursively verify WOTS+ signatures and Merkle paths layer by layer if the final Merkle root matches the public key root the verification is successful.

SPHINCS creates large signatures of size approx. 41kB and is space intensive, this problem is resolved in SPHINCS+ by making use of Forest of Random Subsets (FORS) instead of HORST which is also a few-time signature scheme but much faster and space efficient making the signatures much smaller. SPHINCS+ was subsequently standardized by NIST as SLH-DSA under FIPS 205 [19].

F. ML-DSA

ML-DSA (Module lattice based Digital Signature algorithm) is based on the Module learning with Errors (MLWE) problem and employs the Fiat-Shamir with aborts construction. It is

Schnorr-like signature with several optimizations, the Schnorr signature scheme applies the Fiat-Shamir heuristic to an interactive protocol between a verifier who knows g (the generator of a group in which discrete logs are believed to be difficult) and the value $y = gx$ and a prover who knows g and x [13]. The interactive protocol, where the prover demonstrates knowledge of x to the verifier, consists of three steps:

- **Commitment:** The prover generates a random positive integer r that is less than the order of g and commits to its value by sending gr to the verifier.
- **Challenge:** The verifier sends a random positive integer c that is less than the order of g to the prover.
- **Response:** The prover returns $s = r - cx$ reduced modulo the order of g , and the verifier checks whether $g^s \cdot y^c = g^r$.

This protocol is rendered noninteractive and converted into a signature scheme by replacing the verifier's random choice of c in step 2 with a deterministic process that pseudo randomly derives c from a hash of the commitment gr concatenated with the message to be signed. For this signature scheme, y is the public key, and x is the private key [13].

- **Commitment:** The prover generates $y \in \mathbb{Z}_q^l$ with small coefficients and commits to its value by sending $w_{\text{Approx}} = Ay + y_2$ to the verifier, where $y_2 \in \mathbb{Z}_q^k$ is a vector with small coefficients.
- **Challenge:** The verifier sends a vector $c \in \mathbb{Z}_q^n$ with small coefficients to the prover [13].
- **Response:** The prover returns $z = y + S_1 c$, and the verifier checks that z has small coefficients and that $Az - Tc \approx w_{\text{Approx}}$.

Using Schnorr signatures, the signer derives the challenge by a pseudorandom process from a hash of the commitment concatenated with the message. To correct the bias, the signer applies rejection sampling to z ; if coefficients of z fall outside of a specified range, the signing process is aborted, and the signer starts over from a new value of y [13]. For more efficiency of ML-DSA scheme, some modifications are added to the security framework, like:

- To reduce key and signature size and to use fast NTT-based polynomial multiplication, ML-DSA uses module-structured matrices.
- To obtain beyond unforgeability (BUFF) properties, ML-DSA does not directly sign the message M but rather signs a message representative μ that is obtained by hashing the concatenation of a hash of the public key and M .

The algorithm is believed to be strongly unforgeable, implies that scheme holds the capability to detect the unauthorized modifications to data and to authenticate the identity of the signatory. The algorithm is applicable with matrices of different sizes:

- ML-DSA-44
- ML-DSA-65
- ML-DSA-87

G. ML-KEM

ML-KEM is a key-encapsulation mechanism based on CRYSTALS-KYBER, standardized by NIST as FIPS 203 [18]. The security of ML-KEM is based on the presumed hardness of the Module Learning with Errors (MLWE) which is generalization of LWE and the hardness of the MLWE problem is based on the complexity of

certain computational problems in module lattices. In the LWE problem, the input is a set of random “noisy” linear equations in some secret variables $x \in \mathbb{Z}_q^n$, and the task is to recover x . The noise in the equations is such that standard algorithms (e.g., Gaussian elimination) are intractable. The LWE problem naturally lends itself to cryptographic applications. Though, in MLWE, $\mathbb{Z}_q^n$ is replaced by a certain module R_q^k , [14] which is constructed by taking the k -fold Cartesian product of a certain polynomial ring R_q . In particular, the secret in the MLWE problem is an element x of the module R_q^k .

The construction of the ML-KEM scheme proceeds in two steps.

1. Construct a public-key encryption (PKE) scheme from the MLWE problem.
2. Convert PKE scheme into a key-encapsulation mechanism using Fujisaki-Okamoto (FO) transform.

To improve the efficiency of multiplication in the ring R_q , as an integral part and not only for optimization, ML-KEM uses Number Theoretic Transform (NTT) where R_q is the ring $\mathbb{Z}_q[X]/(X^n + 1)$ of polynomials of the form $f = f_0 + f_1X + \dots + f_{255}X^{255}$ (where $f_j \in \mathbb{Z}_q$ for all j), with the ring operations defined by arithmetic modulo $X^n + 1$ [15].

The ring R_q is isomorphic to another ring T_q , which is a direct sum of 128 quadratic extensions of $\mathbb{Z}_q$. The NTT is a computationally efficient isomorphism between these two rings. When a polynomial $f \in R_q$ is input, the NTT outputs an element $\hat{f} := \text{NTT}(f)$ of the ring T_q , where $\hat{f}$ is called the “NTT representation” of f . The isomorphism property implies that $f \times_{R_q} g = \text{NTT}^{-1}(\hat{f} \times_{T_q} \hat{g})$ where $\times_{R_q}$ and $\times_{T_q}$ denote multiplication in R_q and T_q , respectively. Moreover, since T_q is a product of 128 rings that each consist of polynomials of degree at most one, the operation $\times_{T_q}$ is much more efficient than the operation $\times_{R_q}$ [15].

ML-KEM comes equipped with three parameter sets:

- ML-KEM-512 (security category 1)
- ML-KEM-768 (security category 3)
- ML-KEM-1024 (security category 5)

For Parameters set of ML-KEM, each parameter set assigns a particular numerical value to five integer variables: k , η_1 , η_2 , d_u , and d_v , in addition, with two constants: $n = 256$ and $q = 3329$. When all inputs are well-formed and randomness generation is successful, the key establishment procedure of ML-KEM will never explicitly fail, meaning that both ML-KEM. Encaps (Encapsulation) and ML-KEM. Decaps (Decapsulation) will each output a 256-bit value, if the generated outputs match, then it can be used directly as a shared secret key for symmetric cryptography.

- The parameter k determines the dimensions of the matrix A that appears while key generation and Encryption. It also determines the dimensions of vectors s and e while key generation and the dimensions of vectors y and e_1 during encryption.
- Parameter 1 is required to specify the distribution for generating the vectors s and e while key generation and the vector y during encryption.
- The parameter η_2 is required to specify the distribution for generating the vectors e_1 and e_2 while encryption.
- The parameters d_u and d_v serve as parameters and inputs while encryption and decryption.

	n	q	k	η_1	η_2	d_u	d_v	required RBG strength (bits)
ML-KEM-512	256	3329	2	3	2	10	4	128
ML-KEM-768	256	3329	3	2	2	10	4	192
ML-KEM-1024	256	3329	4	2	2	11	5	256

Figure 1: Standard Parameters for ML-KEM Algorithm

Each ML-KEM parameter set is claimed to be at least as secure as a generic block cipher with a prescribed key size or a generic hash function with a prescribed output length. More precisely, it is claimed that the computational resources needed to break ML-KEM are greater than or equal to the computational resources needed to break the block cipher or hash function when those computational resources are estimated using any realistic model of computation

H. FALCON

FALCON (Fast Fourier Lattice-based Compact signatures over NTRU) is a lattice-based signature scheme. The framework works on two important factors:

- NTRU – A class of cryptographic lattices.
- Fast Fourier Sampling - a technique for trapdoor sampler.

In a nutshell, the Falcon signature scheme may therefore be described as follows:

Falcon = GPV framework + NTRU lattices + Fast Fourier sampling

The use of NTRU lattices by NTRUSign allows it to be very compact, however it had a flaw which led to devastating key-recovery attacks. To overcome the flaws GPV framework is implemented providing a provably secure way for building secure hash-and-sign lattice-based signature schemes. The combination of GPV framework with NTRU lattices, results in provably secure NTRU Sign [16].

To avoid hindrance during transition from pre-quantum to post-quantum schemes, falcon's design principle using lattice-based signatures which keep both public key $|pk|$ and $|sig|$ rather small. The Main elements in Falcon are polynomials of degree n with integer coefficients. The degree n is normally a power of two (typically 512 or 1024). Computations are done modulo a monic polynomial of degree n denoted ϕ (which is always of the form $\phi = x^n + 1$).

When using a trapdoor sampler, an important parameter to set is the standard deviation σ . If it is too low, it could lead to disclosure of the secret basis (and indeed, for all known samplers, a value $\sigma = 0$ opens the door to learning attacks, if it is too high, the sampler does not return optimally short vectors, and the scheme is not as secure as it could be. So, Fast Fourier sampler shares many similarities with Klein's sampler, including shares the optimal value for σ , mitigating the identified issues.

$$\sigma = \eta_c(Z^{2n}) \cdot \|B\|_{GS}$$

The main advantage of Falcon is its compactness. The signature generation and verification procedures are very fast. GPV framework is proven secure in the Quantum (random oracle model), assuming the hardness of SIS for some parameters.

IV. Performance Results of Algorithms on Embedded device & Linux system

A. Cross Signature

The integration of the proposed hybrid (classical-post quantum) signature scheme with various chosen KEMs over the TLS channel results in clear and quantifiable performance results. The size of the cross-signature is relatively small, making it easier to execute quickly and integrate with lower complexity. The integration of the signature scheme with various KEMs results in different performance results, mainly because of the computational dependencies of the KEM algorithms.

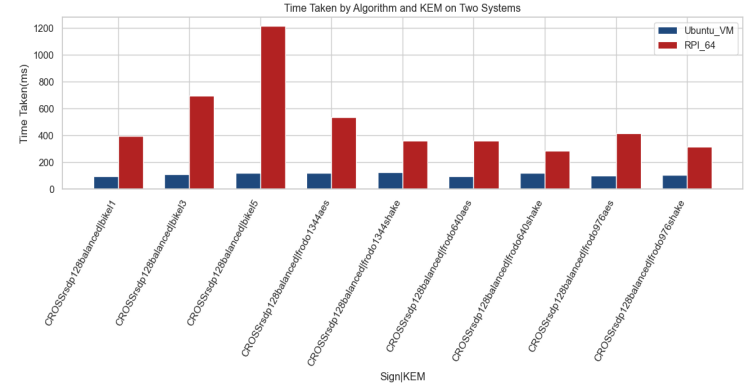


Figure2: Time taken by cross signature with KEMs on platforms

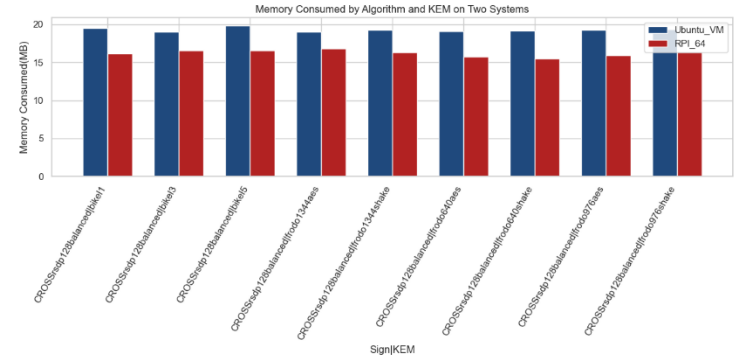


Figure3: Memory Consumption by cross signature with KEMs on platforms

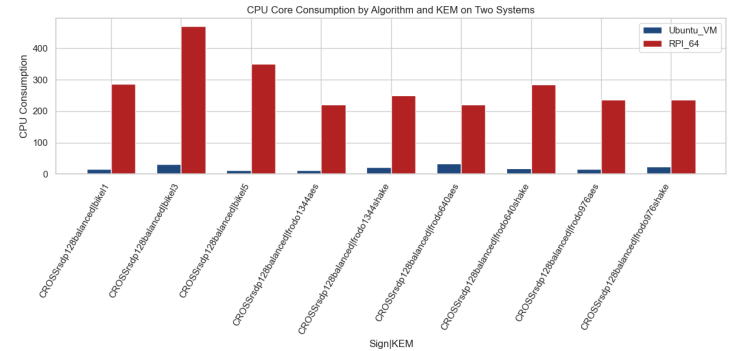


Figure 4: CPU Core consumption by cross signature with KEMs on platforms

As shown in the figures, platform-specific differences can be noticed. Specifically, the integration with BIKE L5 has the highest execution

time on the Raspberry Pi platform, while BIKE L3 has higher CPU core usage. Based on the performance characteristics of the signature scheme, it can be appropriately used for various automotive and IoT-related applications, such as OTA update paths, vehicle-to-backend communication, and supplier ECU delivery.

B. MAYO

The proposed signature scheme is especially suited for applications where a small key size and a low-latency signing process are required. The signature scheme provides three different variants, which improves its flexibility and allows it to be used in a wide variety of applications. During the implementation of the signature scheme in the TLS communication with the chosen KEMs, the performance characteristics were determined and are shown in the respective figures.

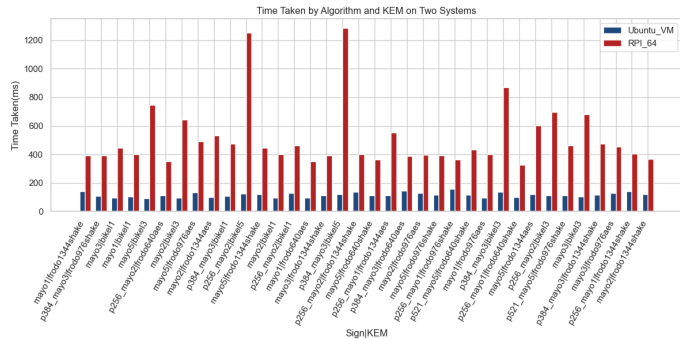


Figure 5: Time taken by Mayo signature with KEMs on platforms

Because of the relatively higher computational complexity of some KEMs, such as BIKE L3, BIKE L5, and FrodoKEM-1344-AES, there was an increase in communication latency observed on both Raspberry Pi and Linux platforms.

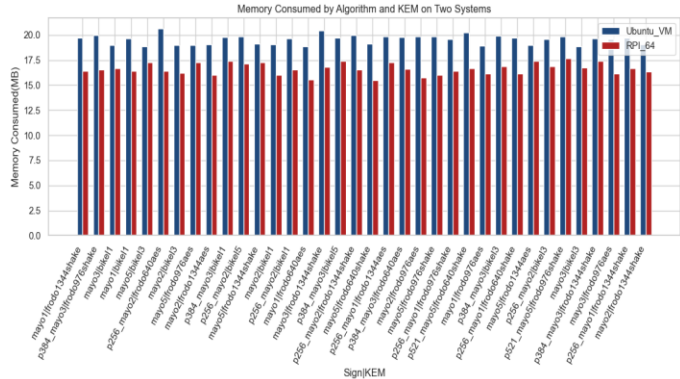


Figure 6: Memory consumption by Mayo signature with KEMs on platforms.

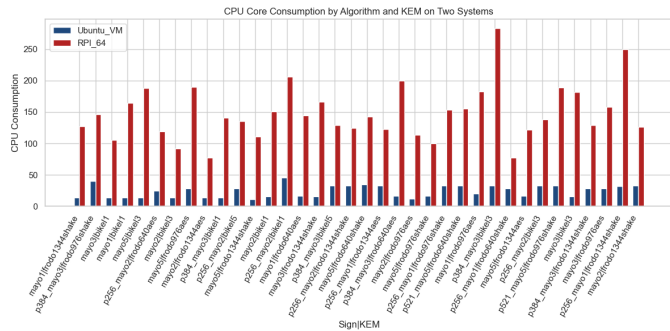


Figure 7: CPU Core consumption by Mayo signature with KEMs on platforms

The figures presented above show the cross-platform results of memory usage, CPU core usage, and execution time, which were analyzed to conclude the most suitable deployment environment for each setup. In the automotive and IoT context, the signature scheme shows high appropriateness for use cases such as secure boot, due to its small signature size and efficient verification time, and vehicle-to-cloud API communication, where it can be employed for efficient message-level authentication.

C. SPHINCS

The advantage of SPHINCS signature is its stateless and have hierarchical tree-like structure. We have integrated two different variants of SPHINCS+ signature with KEMs, to identify optimal results for distinct use cases.

i. SPHINCS-SHA

The signature uses SHA-256/SHA-512 hash function, providing flexibility for implementation in embedded devices, sharing wide hardware support. In the identified performance of signature in standalone and hybrid nature with KEMs, depicts the relevant results.

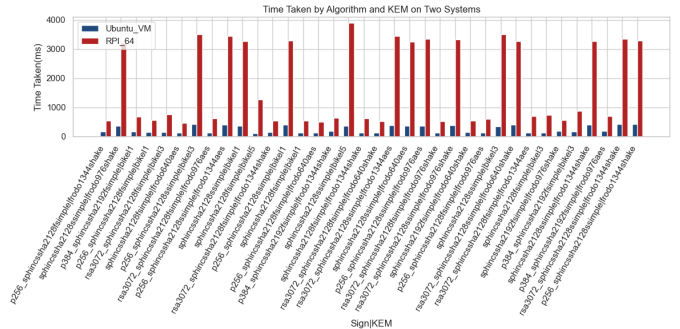


Figure 8: Time taken by SPHINCS-SHA with KEMs on platforms

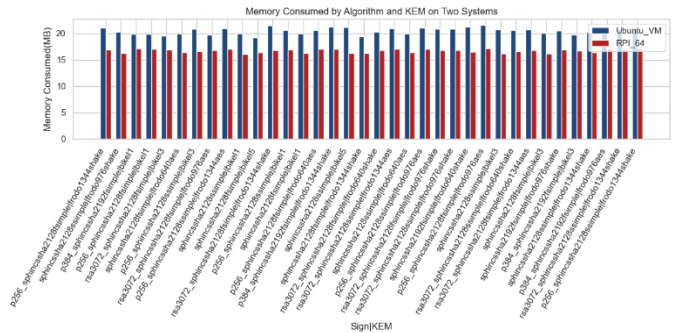


Figure 9: Memory Consumption of SPHINCS-SHA with KEMs on platforms

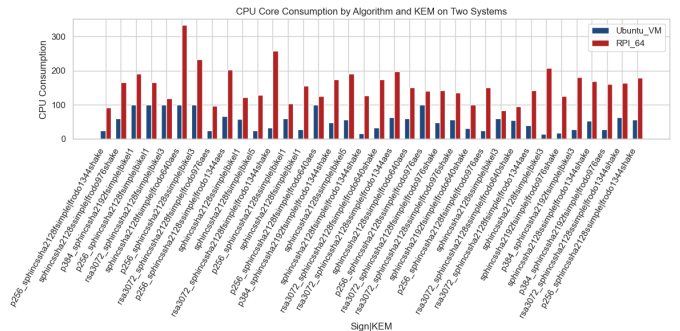


Figure 10: CPU Core Consumption of SPHINCS-SHA with KEMs on platforms

As evident from the performance measurements, SHA-based schemes are better supported by embedded hardware and software technology platforms, but with relatively slower speed performance for signing and verification, and a slightly higher memory usage. But due to the popularity of SHA hardware accelerators, the overall implementation is quite feasible and efficient. Therefore, the scheme will be more suited to sensitive applications such as secure boot, OTA update verification, and gateway ECU authorization.

ii. SPHINCS-SHAKE

The signature uses SHAKE-256 (XOF) hash function. The implementation for the signature is simple though it shares limited hardware support. The availability of the hardware accelerator for the signature process for embedded devices is limited, making it possible for limited set of use cases.

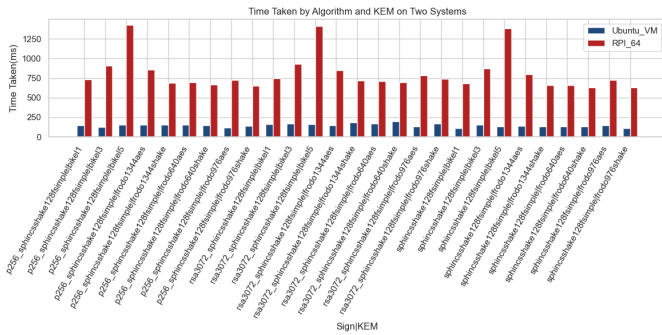


Figure 11: Time Taken by SPHINCS-SHAKE with KEMs on platforms

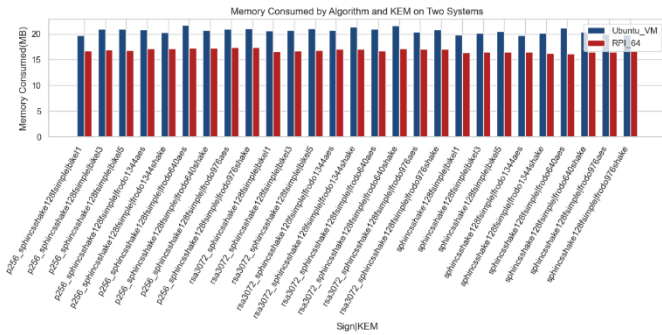


Figure 12: Memory Consumption of SPHINCS-SHAKE with KEMs on platforms

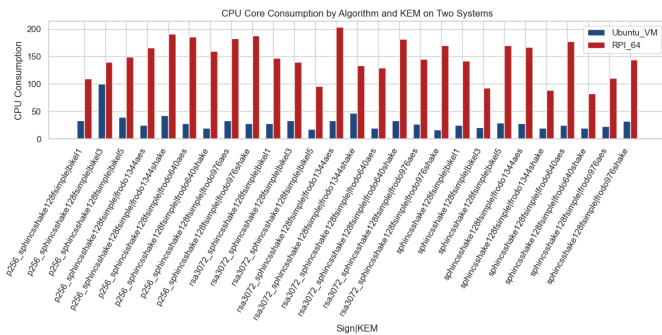


Figure 13: CPU Core Consumption of SPHINCS-SHAKE with KEMs on platforms

The graphs illustrate the performance achievements for the standalone signature scheme as well as the combination of the signature scheme and various KEMs. According to the performance indicators, the signature scheme has a relatively fast processing speed and thus can be applied in cases where fast large-scale cryptographic computations are required. These include car event log signature verification, software verification, and PKI chain validation.

D. ML-DSA

The signature scheme has moderate signature size and, most importantly, does not require any kind of hardware support, which is a major advantage for a wide range of IoT/automotive applications. Some cases of use can be OTA signing, firmware signing, Secure Boot, or V2X communication, the security of which is governed by standards such as IEEE 1609.2 [24].

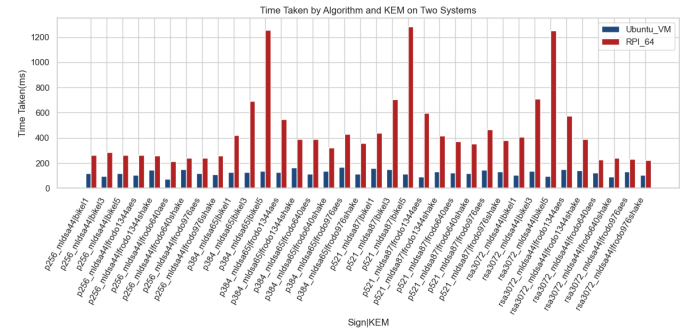


Figure 14: Time taken by ML-DSA with KEMs on platforms

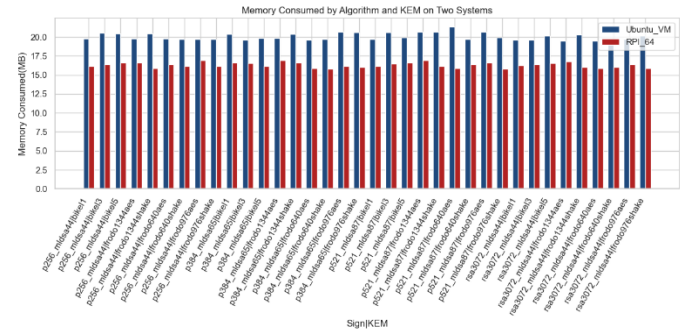


Figure 15: Memory Consumption of ML-DSA with KEMs on platforms

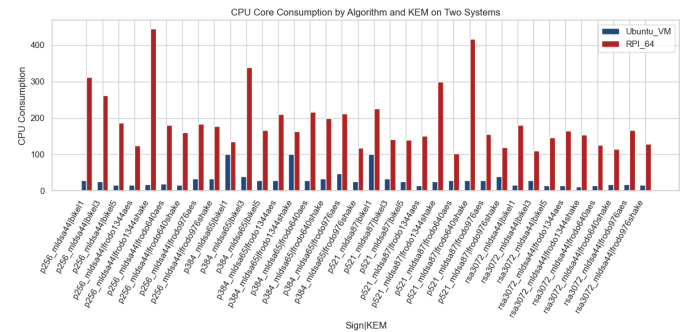


Figure 16: CPU Core Consumption of ML-DSA with KEMs on platforms

The performance achieved on the embedded platforms as well as on Linux platforms with a variety of KEMs is given in the figures. These highlight the performance properties of the signature scheme in both

standalone modes as well as in hybrid modes, which allows a fair comparison in every aspect.

E. FALCON

The signature is known for its compactness and small signature size, making it relevant for practical use cases. Due to small size, the verification speed is fast for the signature. Due to complexity (NTRU lattices), the implementation of signature is difficult specifically for constrained devices, though it shares excellent TLS suitability. The results for the performance metrics for the standalone and hybrid signatures with KEMs are represented.

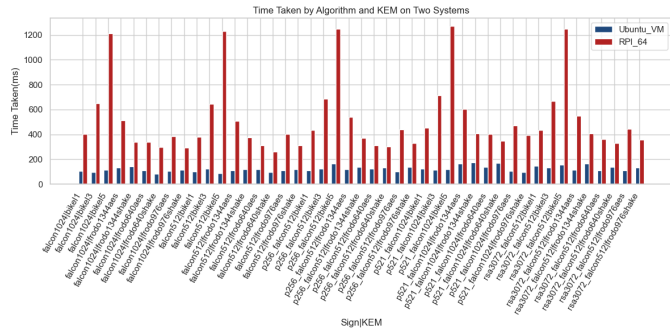


Figure 17: Time taken by Falcon with KEMs on platforms

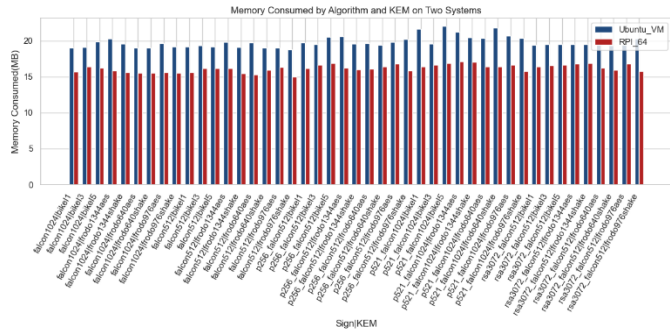


Figure 18: Memory Consumption by Falcon with KEMs on platforms

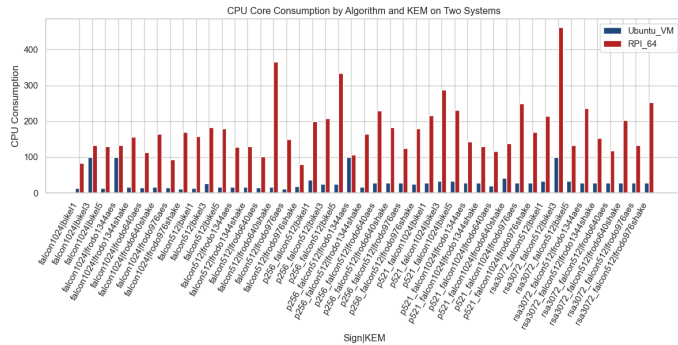


Figure 19: CPU core consumption by Falcon with KEMs on platforms

As per the nature and performance of the signature, the signature is relevant for the use cases like ECU to backend communication due to minimal handshake size, Telematics TLS authentication, PKI validation – faster chain verification etc.

V. Conclusions

The paper examines the implementation performance of post quantum cryptography standalone and hybrid (classical + PQC) algorithms over

TLS protocol with integration with KEMs on embedded systems constrained by computational power and memory and Linux platforms (Ubuntu). It categorizes the performance of PQC algorithms on different platforms for important parameters: time taken, memory consumption and CPU core consumption. Based on the quantitative results obtained in this study, we identified the most relevant deployment scenarios for the evaluated post-quantum cryptographic algorithms, with a particular focus on automotive and IoT systems, in line with the practice-based migration guidance set out by NIST SP 1800-38A [20]. Lattice-based algorithms demonstrate strong suitability for communication- and infrastructure-heavy use cases such as TLS, PKI, and V2X, where interoperability, scalability, and long-term cryptographic assurance are critical. Their balanced trade-off between security, performance, and ecosystem maturity makes them well aligned with emerging connected-vehicle architectures.

In contrast, signature schemes such as MAYO, characterized by compact signature sizes and fast verification times, are better suited for resource-constrained operations, most notably firmware authentication and ECU software signing. These use cases prioritize verification speed and memory efficiency over frequent key exchange, especially during manufacturing, secure boot, and over-the-air update workflows. The results indicate that a one-size-fits-all approach to PQC adoption is neither practical nor desirable; instead, algorithm selection must be guided by the operational constraints and security objectives of each automotive subsystem.

During our research on the implementation of various PQC algorithms across multiple operating systems and CPU architectures, we observed a clear trend: current software ecosystems and cryptographic libraries are primarily optimized for modern 64-bit platforms, while support for legacy 32-bit architecture remains limited [17]. Although extending PQC support to 32-bit systems is technically feasible, the restricted computational capability and lower data throughput of the 32-bit bus introduce significant performance bottlenecks for these algorithms. Furthermore, the current lack of automotive-grade chipsets equipped with dedicated PQC hardware accelerators further amplifies this challenge. Taking together, these factors may compel the automotive industry to transition to 64-bit architecture sooner rather than later, a shift that has long been deferred but is becoming increasingly difficult to avoid. Future research should explore new techniques to implement these algorithms also compatible with constrained devices.

References

- [1] M. Baldi et al., “CROSS: Codes and Restricted Objects Signature Scheme,” NIST PQC Submission, 2023
- [2] M. Baldi et al., “Zero Knowledge Protocols and Signatures from the Restricted Syndrome Decoding Problem,” 2023
- [3] J. W. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila, "Frodo: Take Off the Ring! Practical, Quantum-Secure Key Exchange from LWE," in Proc. 2016 ACM SIGSAC Conf. on Computer and Communications Security (CCS), Vienna, Austria, Oct. 2016, pp. 1006–1018.
- [4] J.W. Bos et al, “Enabling FrodoKEM on Embedded Devices; NXP Semiconductors, 2023
- [5] E. Alkim et al., “FrodoKEM: Learning with Errors Key Encapsulation – Algorithm Specifications and Supporting Documentation,” Mar.25, 2020
- [6] N. Aragon et al, ”BIKE: Bit-flipping Key Encapsulation”, NIST PQC Seminars, September 27, 2024.
- [7] M. R. Nosouhi, S. W. A. Shah, L. Pan and R. Doss, “Bit Flipping Key Encapsulation for the Post-Quantum Era,” IEEE Access, vol. 11, 2023, doi:10.1109/ACCESS.2023.3282928
- [8] W.Beullens, “MAYO: Practical Post-Quantum Signatures from Oil-and-Vinegar Maps,” 2022.

- [9] W. Beullens, F. Campos, S. Celi, B. Hess, and M. J. Kannwischer, "MAYO Signature Scheme – Algorithm Specifications," NIST PQC Submission, 2023.
- [10] W. Beullens, F. Campos, S. Celi, B. Hess, and M. J. Kannwischer, "MAYO Specification Document," NIST Post-Quantum Cryptography: Additional Digital Signature Schemes, Round 2 Submission, Feb. 2025
- [11] D. J. Bernstein et al., "SPHINCS: Practical Stateless Hash-Based Signatures," 2015
- [12] J.-P. Aumasson, D. J. Bernstein, W. Beullens, C. Dobraunig, M. Eichlseder, S. Fluhrer, S.-L. Gazdag, A. Hülsing, P. Kampanakis, S. Kölbl, T. Lange, M. M. Lauridsen, F. Mendel, R. Niederhagen, C. Rechberger, J. Rijneveld, P. Schwabe, and B. Westerbaan, "SPHINCS+: Submission to the NIST Post-Quantum Cryptography Project," Version 3.1, June 2022.
- [13] National Institute of Standards and Technology, "FIPS 204: Module-Lattice-Based Digital Signature Standard (ML-DSA)," Aug. 2024.
- [14] 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
- [15] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Kyber Algorithm Specifications and Supporting Documentation," Version 3.02, NIST Post-Quantum Cryptography Standardization Project, Aug. 2021.
- [16] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU, Specification v1.2, Oct. 2020
- [17] O. Alnaseri, Y. Himeur, S. Atalla, and W. Mansoor, "Complexity of Post-Quantum Cryptography in Embedded Systems and Its Optimization Strategies," arXiv:2504.13537, 2025.
- [18] National Institute of Standards and Technology, "FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard," Aug. 2024.
- [19] National Institute of Standards and Technology, "FIPS 205: Stateless Hash-Based Digital Signature Standard (SLH-DSA)," Aug. 2024.
- [20] W. Newhouse, M. Souppaya, W. Barker, and C. Brown, "Migration to Post-Quantum Cryptography: Preparation for Considering the Implementation and Adoption of Quantum Safe Cryptography," NIST SP 1800-38A, National Cybersecurity Center of Excellence (NCCoE), 2023.
- [21] International Organization for Standardization / SAE International, "ISO/SAE 21434:2021 — Road Vehicles — Cybersecurity Engineering," 2021.
- [22] United Nations Economic Commission for Europe, "UN Regulation No. 155 — Uniform Provisions Concerning the Approval of Vehicles with Regard to Cyber Security and Cyber Security Management System," WP.29, 2021.
- [23] United Nations Economic Commission for Europe, "UN Regulation No. 156 — Uniform Provisions Concerning the Approval of Vehicles with Regard to Software Update and Software Update Management System," WP.29, 2021.
- [24] IEEE Vehicular Technology Society, "IEEE Std 1609.2-2022 — IEEE Standard for Wireless Access in Vehicular Environments — Security Services for Applications and Management Messages," 2022.

