Post-Quantum Secure Architectures for Automotive Hardware Secure Modules

Wen Wang
Yale University
New Haven, CT, USA
wen.wang.ww349@yale.edu

Marc Stöttinger
Continental AG
Frankfurt, Germany
marc.stoettinger@continental-corporation.com

***Abstract*—The rapid development of IT in the automotive industry has driven increasing requirements on incorporating security functionalities in the in-vehicle architecture, which is usually realized by adding a Hardware Secure Module (HSM) in the Electronic Central Unit (ECU). Therefore, secure communications can be enforced by carrying out secret cryptographic computations within the HSM by use of the embedded hardware accelerators. However, there is no common standard for designing the architecture for an HSM. A future design of a common automotive HSM is desired by the automotive industry which not only fits to the increasing performance demand, but also further defends against future attacks by attackers exploiting large-scale quantum computers. The arrival of future quantum computers motivates the investigation into post-quantum cryptography (PQC), which will retain the security of an HSM in the future. We analyzed the candidates in NIST's PQC standardization process, and proposed new sets of hardware accelerators for the future generation of the automotive HSMs. Our evaluation results show that building a post-quantum secure automotive HSM is feasible and can meet the hard requirements imposed by a modern vehicle ECU.**

***Index Terms*—Post-Quantum Cryptography, Hardware Security Module, ECU, Automotive, FPGA**

I. Introduction

The development of information technology in automotive industry in the last decade has driven the vehicles getting more and more connected nowadays, both in terms of communications between different vehicles and between different embedded subsystems within a vehicle. In a vehicle, a large network of interactive embedded systems built upon Electronic Central Units (ECU) are deployed, each maintains a specific set of functionalities. However, the integration of connectivity services in the in-vehicle architectures not only increases convenience to the users, but also a growing attacking surface for malicious attackers in the automotive world. Nowadays, integrating a Hardware Security Module (HSM), which is a secure enclave encapsulating security-oriented functionalities within an ECU, is a basic component to protect the vehicles from being compromised by the malicious attackers.

Several leading chip manufactures have offered a few HSM solutions with automotive-grade architectures, which usually include a microcontroller processor, different purposed memory blocks (e.g., RAM, ROM, flash), hardware accelerators for hash functions, symmetric and asymmetric cryptographic operations, as well as secure interfaces between the HSM and the host processor within the ECU [1]. Within an ECU, secret-dependent cryptographic operations are usually carried out by use of the hardware accelerators within the HSM. An HSM not only keeps secret information (e.g., secret keys) inside a trusted hardware enclave, but also provides big speedups benefiting from the dedicated hardware accelerations.

However, there has been no official standard for the architecture or features of an HSM in the automotive industry. Hence, to build the future HSMs, more understandings about the current trends in modern automotive industry as well as explorations of the future challenges are needed. On one hand, more powerful HSMs are needed with the increasing performance requirements on modern vehicles. On the other hand, more radical changes may be needed in designing the architecture of the HSMs given the potential arrival of a "quantum era". Commonly used asymmetric cryptographic algorithms, e.g., RSA and ECC, are vulnerable to attacks using quantum computers: Shor's algorithm [2] is able to solve the underlying hard problems of RSA and ECC in polynomial time. Grover's algorithm [3] can provide a quadratic speedup for brute-force search, thus symmetric cryptographic accelerators and hash functions in modern HSMs, i.e., the AES core, are still usable in the future by doubling the key size; however, the asymmetric cryptographic accelerators within HSMs will be fully broken. Given the rapid advances in the construction of the quantum computers, a new field of post-quantum cryptography (PQC) has been developed [4] over the last decade providing cryptographic algorithms that will remain secure against attacks using a quantum computer.

Automotive HSMs will need to use post-quantum secure algorithms to ensure their security in a "quantum era". These PQC algorithms usually have much bigger key sizes compared to today's cryptographic algorithms, and take more time to do the key operations. Moreover, the arithmetic required in PQC algorithms are different from those underlying cryptographic algorithms used today. Therefore, new designs of the hardware architectures for future HSMs are needed.

There exist several general-purpose software-hardware co-designs for PQC schemes (e.g., [5], [6]), however, all of these designs focus on one specific scheme and none of them target the use cases and requirements of automotive HSMs. This work provides a detailed analysis of the applicability of different PQC candidates made to the 2nd round in the NIST

PQC standardization process[1] in the automotive IT architectures. Based on the analysis, post-quantum secure autmotive HSM architectures are proposed with an emphasis on the hardware accelerators for accelerating the recommended PQC algorithms. The contributions of this paper include:

- Four PQC candidates are identified and recommended to secure general automotive use cases. A detailed software profiling results on those candidates are provided to choose and design hardware accelerators.
- Post-quantum secure architecture for automotive HSMs are provided with detailed performance results when the designs are synthesized on an Artix-7 FPGA.
- Hardware accelerators designed for the chosen PQC algorithms will be made available under open-source license.

The reminder of the paper is structured as follows: In the next section we provide some preliminaries on PQC and automotive security use cases. In Section III we first give a recommendation on the signature and KEM schemes to be used in an HSM targeting post-quantum security, then follows the software profiling results of these chosen schemes. In Section IV, hardware architecture designs of the proposed HSMs are proposed and detailed performance results are later presented in Section V.

II. Preliminaries

In this section, we first provide an introduction to the different classes of PQC algorithms. Second, we provide a detailed case study on a typical automotive scenario which covers the typical use cases of modern HSMs.

A. Post-Quantum Cryptography

There are five popular classes of PQC algorithms: hash-based, code-based, lattice-based, multivariate, and isogeny-based cryptography. Each of the classes is based on a different mathematical problem that is hard to be solved by both modern computers and quantum computers. These schemes differ in the size of the keys and messages, the efficiency, as well as the trust in their security analysis, etc.

1) Code-Based Cryptography: Most code-based encryption schemes are based on the McEliece cryptosystem which was first proposed in 1978 [7] and its instantiation using binary Goppa codes remains secure nowadays. However, one main issue in a McEliece cryptosystem is the large size of the public key. Some research efforts have been focused on reducing the size of the keys by incorporating structures into the code, e.g., QC-MDPC codes [8]. However, these structured codes can potentially lead to attacks [9].

2) Hash-Based Cryptography: Hash-based signature schemes are considered very mature as its security fully relies on the properties of the underlying hash function which is well-understood. XMSS [10] and LMS [11] are two such *stateful* hash-based signature schemes that are under NIST's consideration to be standardized early as part of the post-quantum cryptography development[2]. Therefore, hash-based signatures are promising candidates as post-quantum secure signature schemes.

3) Lattice-Based Cryptography: Lattice-based cryptography is arguably the most popular among different PQC families. Its security is based on different types of hard problems defined on a high-dimension lattice, e.g., "learning with errors" (LWE), "shortest vector problem" (SVP), etc. Lattice-based schemes based on generic lattices [12] are generally more confidence-inspiring while those based on ideal lattices [13] have much smaller keys and better performance. However, choosing security parameters for lattice-based schemes has always been challenging as their security against quantum-computer attacks is not yet well-understood nowadays [14].

4) Multivariate Cryptography: Multivariate cryptography is based on the hardness of solving a multivariate quadratic system of equations over a finite field, which is an NP-hard problem. Although the security of multivariate cryptosystems is well analyzed, it is not an easy task to construct such a scheme both securely and efficiently. Many schemes have been proven insecure over the last decade. Few of them focused on signature schemes (e.g., [15], [16]) remain secure nowadays.

5) Isogeny-Based Cryptography: Isogeny-based cryptography is the youngest family among all the PQC candidates which was initially proposed as an encryption scheme in 2006 [17]. Its underlying arithmetics are similar to those in classical ECC schemes since the scheme is constructed based on the isogenies between elliptic curves. However, due to the novelty of isogeny-based problems, there is not yet enough confidence in these schemes.

B. Case Study on SOTA

Software-over-the-air (SOTA) update is a technology for automotive manufactures to maintain and improve vehicles through downloading remote software updates. It is also an excellent example of interaction of various basic security use cases on ECU level. Therefore, we use SOTA to identify the required security features of an HSM.

Algorithm 1 depicts the work flow of a SOTA SW update on one "target" ECU. The update process is supported by a dedicated ECU with connectivity features.

As we can see from the above process, a complete SOTA update includes most of the use cases of an HSM, i.e., *Secure Boot*, *Secure Software Update* and *Secure Diagnosis*. The only use case not directly covered by this SOTA update is the *Secure On-board Communication (SecOC)* which usually relies on symmetric cryptography (e.g., AES) to maintain low latencies. Therefore, SOTA is a good case study for analyzing the required features in an automotive HSM.

The SOTA update scenario comes with certain requirements with respect to run-time and memory storage: The authenticity verification and integrity verification processes are expected to finish within 1s, the public verification key, the signature (used

[1] https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-2-Submissions

[2] https://csrc.nist.gov/Projects/Stateful-Hash-Based-Signatures

Algorithm 1 Software over-the-air update

1: Backend notifies the vehicle about available SW update.
2: A dedicated ECU inside the vehicle *verifies* the *authenticity* of the notification from the backend.
3: **if** verification is successful **then**
4: SW package is downloaded by the dedicated ECU.
5: **end if**
6: **if** SW packages is *authentic* and *integrity* is ensured **then**
7: Start flashing process via a *secure diagnosis* access protocol.
8: Target ECU checks the *integrity* of the flashed SW.
9: **if** flashed SW is correct **then**
10: Target ECU updates its *secure boot* reference value.
11: Reboot target ECU.
12: **end if**
13: **end if**

for checking *authenticity*) and the token (used for checking *integrity*) should all be smaller than 4kB. In addition, it is considered that SecOC and SOTA can happen concurrently. Due to the AUTOSAR standard[3], SecOC leverages AES-CMAC for message authentication.

Both SOTA update and SecOC are security-dependent services that are carried out inside an HSM. Hence, an HSM requires the use of the following primitives: Digital signature, hash function, and symmetric cryptography. Modern HSMs usually have no dedicated use cases for public key encryption or key encapsulation (KEM) on ECU level. However, future generations of HSMs may require the support of KEM schemes, e.g., to support possible key exchanges between ECUs during runtime on vehicle level. Thus we take KEM schemes into account in designing futures HSMs as well.

III. PQC for Automotive HSMs

A. Recommended PQC Schemes

We choose four PQC schemes from the candidates in the 2nd round of the NIST PQC standardization process targeting NIST security level 3 (*medium*) and 5 (*high*) in order to provide a trade-off between performance and security for different automotive applications. All of the four chosen schemes satisfies the hard requirements imposed in Section II-B.

1) Schemes for Medium Security Level: For applications requring a *medium* security level, ideal-lattice-based signature scheme *CRYSTALS-Dilithium* and code-based KEM scheme *ROLLO* are chosen due to their efficiencies and small key sizes. *Dilithium* [18] is based on a variant of a well-studied problem, which is the Module Learning with Errors (MLWE) problem. Apart from having good performance, it is quite easy to implement the operations in *Dilithium* both in software and in hardware. *ROLLO* [19] is a merger combining three submissions from the first round of the NIST PQC standardization process. All of the merged algorithms are based on the ideal Low Rank Parity Check (LRPC) codes. Compared with the other KEM candidates, *ROLLO* has the smallest key and the highest overall bandwidth for the same security against known attacks, thus it is most suitable for automotive use cases. Therefore, *Dilithium* and *ROLLO* can be combined and used for applications having strict requirements on both performance and key sizes, but without very high security requirements.

2) Schemes for High Security Level: Stateful hash-based signature scheme *XMSS* and ideal-lattice-based KEM scheme *CRYSTALS-Kyber*, both with high security levels, are chosen given their underlying well-studied mathematical problems. *XMSS* has been standardized by the IETF [10], and is currently one of the two *stateful* hash-based signature schemes that are being considered as early standardizations by NIST. The security of *XMSS* fully relies on the underlying hash function which can be efficiently constructed even in the presence of large quantum computers [20]. The security of *Kyber* [21] relies on the same problem as *Dilithium*, and therefore its security is well-studied as well. Moreover, both *XMSS* and *Kyber* have relatively small keys, and their key operations are quite fast. Therefore, *XMSS* and *Kyber* can be combined and used for applications targeting a high security level given their confidence-inspiring security reductions.

B. Software Profiling Results

For designing a post-quantum secure HSM, we first need to identify a new set of hardware accelerators since the mathematical problems underlying PQC schemes are usually radically different from modern cryptography. As a software-hardware co-design, hardware accelerators in HSMs are designed to accelerate the most compute-intensive operations. To understand what are the most compute-intensive operations in the recommended schemes (in Section III-A), we profiled the reference software implementations included in their submissions to the 2nd round of the NIST PQC competition by use of the performance analysis tool *Gprof*.

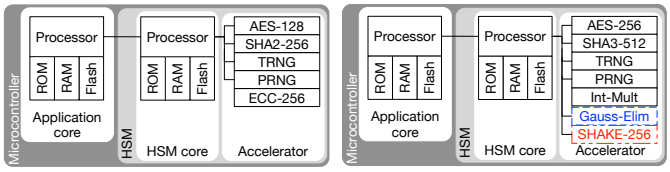
For signature schemes *XMSS* and *Dilithium*, only the signature verification operation is included in the profiling while for KEM schemes *ROLLO* and *Kyber*, all the operations (e.g., key generation, key encapsulation and key decapsulation) are included. The results are shown as follows:

1) XMSS: In *XMSS*, most of the time in the verification operation is consumed by the underlying hash function. For *high* security level, SHA2-512 or SHAKE-256 is used as the hash function [10].

2) ROLLO: ROLLO-I is chosen for the profiling purpose. Given the similaries in ROLLO-I/II/III, its profiling result reflects well the common patterns in all of the algorithm variants. The result shows that apart from the big overhead in conversions between strings and bytes, most of the computation time in *ROLLO* is taken by Gaussian elimination operations on big matrices and AES-256 to expand the randomness.

3) Dilithium and Kyber: For *Dilithium* and *Kyber*, the profiling results are quite similar as they are based on very similar lattice-based problems. Both of them have two implementation variants: One uses SHAKE to sample the random

[3]https://www.autosar.org/standards/



(a) EVITA HSM - oriented on EVITA level full

(b) PQC HSM - security specific accelerators: blue for security level medium and red for security level high

Fig. 1: Basic accelerator architectures of HSMs

numbers and expand the randomness and the other one uses AES-256 instead. For both of these two implementations, the SHAKE/AES computation takes the most of the time, which is followed by modular multiplications: Integer multiplications followed by Barrett [22] or Montgomery [23] reductions which is further computed through several multiplications.

IV. Hardware Architectures of PQC HSM

In this section, we focus on the discussions of designing an HSM for applications in the automotive domain. As a design reference we focus on the HSM definition given in the EVITA project[4]. Figure 1a depicts a typical architecture of an EVITA like HSM. Since EVITA *medium* and EVITA *light* are subsets of a EVITA *full* HSM, the analysis on these smaller HSMs can be done similarly. From now on we will refer to EVITA *full* HSMs as modern HSMs.

A. Modern HSMs

A modern *full* HSM typically includes the following cryptographic building blocks [1], as summarized in Table I:

- **SHA2-256**, used as a general-purpose hash function as an alternative to the originally proposed *WHIRLPOOL*.
- **(AES-128, all)**, used for symmetric cryptographic operations including: *Key generation*, *encryption* and *decryption*, as discussed in Section II-B, this block is required for SecOC.
- **TRNG**, used as a true random number generator (TRNG). The digital entropy is collected from the variances in the hardware, e.g., jitter between digital ring oscillators.
- **(AES-128, enc)**, used as a pseudo random number generator (PRNG). This PRNG is usually seeded by a **TRNG** and expands the randomness by use of an AES engine which only supports the *encryption* operation.
- **ECC-256**, used for 256-bit elliptic curve arithmetics in an asymmetric cryptosystem.

Once big quantum computers are available, such an HSM is no longer secure: The security of **ECC-256** is fully compromised due to Shor's algorithm [2] while the securities of **SHA2-256**, **(AES-128, all)** and **(AES-128, enc)** are halved due to Grover's algorithm [3]. To ensure that an automotive HSM stays safe even against quantum computers, post-quantum secure asymmetric primitives should be used to replace **ECC-256** and at the same time, symmetric primitives and hash function with doubled security should be used.

[4]https://www.evita-project.org

Cryptographic Block	Modern HSM like *EVITA Full*	PQC HSM *medium* SL	PQC HSM *high* SL
Hash Function	SHA2-256	SHA3-512	SHA3-512
Symmetric	(AES-128, all)	(AES-256, all)	(AES-256, all)
PRNG	(AES-128, enc)	(AES-256, enc)	(AES-256, enc)
TRNG	Hardware-Based	Hardware-Based	Hardware-Based
Asymmetric	ECC-256	(AES-256, enc) Int-Mult Gauss-Elim	SHAKE-256 Int-Mult

TABLE I: Comparisons of Cryptographic Blocks Required in Modern HSMs and Post-Quantum Secure HSMs (of medium and high security levels respectively); SL = Security Level.

B. PQC HSMs

In a post-quantum secure HSM (PQC HSM), hash function and symmetric cryptographic primitives are all chosen such that they can maintain a 128-bit security level against quantum computers for both *medium* and *high* security levels in order to be more conservative. For asymmetric cryptographic primitives, much more radical changes are needed: Fully different cryptographic blocks are needed to build a PQC HSM compared to those needed in modern HSMs. Table I and Figure 1b summarizes the HSM solutions we proposed targeting *medium* and *high* security levels respectively. The following cryptographic blocks are recommended for a PQC HSM:

- **SHA3-512**, used as a general-purpose hash function. SHA3-512 is part of the SHA-3 standard released by NIST in 2015 [24] and its structure is internally different from the SHA-2 standard.
- **(AES-256, all)**, used for symmetric cryptographic communications. It has the same purpose as **(AES-128, all)**, but with doubled security level.
- **(AES-256, enc)**, used as a PRNG.

1) PQC HSMs with Medium Security Level: For a PQC HSM with *medium* security level, as discussed in Section III-A1, *Dilithium* is recommended as the signature scheme replacing *ECC-256*, and in addition, *ROLLO* is recommended to support KEM operations which might be needed for some use cases in future automotive HSMs. As discussed in Section III-B, randomness expansion and multiplications are two of the most compute-intensive operations in *Dilithium*. Therefore, the following cryptographic blocks are recommended to accelerate *Dilithium*:

- **Int-Mult**, a 32-bit signed integer multiplier, is needed to support the multiplication operations. This integer multiplier can also be a general purpose component of the HSM core.

HW Accelerator	Functionality	Feature	LUT	FF	RAMB36	DSP	Fmax (MHz)
Modern HSM Specific Hardware Crytographic Accelerators.							
(AES-128, enc)	PRNG	Iterative	653	141	0	0	382
(AES-128, all)	Symmetric Crypto	Iterative	2932	2471	0	0	168
SHA2-256	Hash Function	Iterative	2069	1831	0	0	113
ECC-256	Asymmetric Crypto	NIST Curve P-256	3602	3443	24	35	123
Common Blocks in Modern HSMs and PQC HSMs.							
TRNG	TRNG	Ring Oscillator	419	105	0	0	419
Internal CPU	Microprocessor	Murax SoC	1387	1506	33	0	218
PQC HSM Specific Hardware Crytographic Accelerators.							
(AES-256, enc)	PRNG	Iterative	647	301	0	0	237
(AES-256, all)	Symmetric Crypto	Iterative	2958	2474	0	0	165
SHA3-512	Hash Function	Balanced	3589	2244	0	0	232
Int-Mult	Asymmetric Crypto	32-bit Signed	1064	64	0	0	232
SHAKE-256	Asymmetric Crypto	Parallel Slices = 16	2778	223	0	0	152
Gauss-Elim	Asymmetric Crypto	Array Size = 32	1641	2466	3.5	0	357
Total Logic Utilization for Modern HSMs and PQC HSMs.							
Total	Modern HSM	Similar to EVITA Full	11062	9497	57	35	113
Total	PQC HSM	Medium Security Level	11705	9160	36.5	0	165
Total	PQC HSM	High Security Level	12842	6917	33	0	152

TABLE II: Performance results of hardware accelerators synthesized for Aritx-7 FPGA with Vivado 2018.3.

- **(AES-256, enc)**, an AES-256 module supporting the encryption functionality is needed for expanding randomness in the AES-based implementation of *Dilithium*. In this case, *Dilithium* and the PRNG can share the same (**AES-256, enc**) core.

For *ROLLO*, since Gaussian elimination is the most expensive operation (as shown in Section III-B), the following module is designed:

- **Gauss-Elim**, composed of a systolic array which is able to compute on multiple matrix columns in parallel.

2) PQC HSMs with High Security Level: For a PQC HSM with *high* security level, as discussed in Section III-A2, *XMSS* is recommended as the signature scheme replacing *ECC-256*, and *Kyber* is recommended as the KEM scheme. Similarly, **Int-Mult** is needed for accelerating *Kyber*. Moreover, the following cryptographic core is needed:

- **SHAKE-256**, it is chosen as both *XMSS* and *Kyber* targeting *high* security level can be constructed with SHAKE-256 and this enables hardware resource sharing.

V. Evaluation and Comparison

In this section, detailed performance and synthesis results of the two recommended PQC HSMs with *medium* and *high* security levels are discussed. The designs are synthesized for the Artix-7 FPGA platform (model XC7A200TFFG1156-3). Moreover, we compare the PQC HSM results with a typical modern HSM design.

A. Evaluation

Among all the hardware accelerators, **AES-128/256**[5], **ECC-256**[6], **SHA2-256**[7], **SHA3-512**[8], **TRNG**[9] and **Int-Mult**[10] are taken directly from open-source. **AES-128/256**, **SHA2-256** are all based on computations on blocks iteratively. The selected **TRNG** collects its digital entropy source based on jitter between multiple digital ring oscillators. This true randomness can be further fed into PRNG to expand the randomness. Murax SoC is used as a representative of the **Internal CPU** within an HSM. It is an open-source RISC-V based processor with a very small resource usage but maintains performance comparable to an ARM Coretex-M3 processor [6]. All of these cores are chosen with target of achieving a good time-area efficiency.

SHAKE-256 is constructed based on an area-efficient SHAKE module proposed in [25]. Within the hash core, the number of parallel slices can be flexibly chosen at synthesis time in order to achieve a trade-off between area and performance. In our work, in total 16 slices are used in parallel to achieve an area-performance balanced design.

Gauss-Elim is a variant of the design proposed by Wang, Szefer and Niederhagen in 2016 [26]. Similar to [26], **Gauss-Elim** is constructed by use of a square processor array where the diagonal processors are responsible for finding the pivoting element and generating commands, and the rest of the processors simply carry out those commands and deliver the same commands to the processors on the same row. In design [26], a matrix is transformed into its reduced row

[5]https://github.com/secworks/aes
[6]https://trac.cryptech.is/browser/user/shatov/ecdsa256
[7]https://github.com/secworks/sha256
[8]https://opencores.org/projects/sha3
[9]https://trac.cryptech.is/browser/core/rng/rosc_entropy
[10]https://github.com/suoglu/Integer-Multiplier-Hardware-Parameterized

echelon form after one pass. However, in our design, **Gauss-Elim** is able to transform a matrix to its row echelon form when used once (one-pass), and when used twice, a reduced row echelon form of the matrix is achieved (two-pass). Since both transformations are needed in the *ROLLO* scheme, our design can be used to accelerate both transformations. The size of the array can be flexibly tuned by setting a user-defined parameter at synthesis time. The array size is chosen as 32 for PQC HSMs as shown in Table II in order to fit to the word size of the microprocessor and at the same time maintain a time-area balanced design.

B. Modern HSM vs PQC HSMs

As depicted in Table II, we summarized the total area usage for three HSM configurations: Modern HSM, PQC HSM with *medium* security level and PQC HSM with *high* security level. Compared with modern HSM, PQC HSMs introduce a small overhead in terms of area, e.g., in terms of LUT usage, a 5% and a 16% overhead are introduced in PQC HSMs targeting *medium* and *high* security levels respectively.

However, it is worth mentioning that, since the performance of these cryptographic cores are implementation-specific, different optimizations can be applied to tune the performance results. Our work targets at exploring the feasibility of designing a time-area efficient HSM for typical automotive use cases which can defend against large scale quantum computers. Based on the analysis results, we can conclude that it is feasible to design PQC HSMs with 128-bit quantum security level that are able to maintain good performances without bearing big overhead.

VI. Conclusion

In this paper, we presented the first analysis on designing a post-quantum secure HSM for automotive use cases. We first analyzed the typical use cases in an automotive HSM, and then recommended signature and KEM schemes based on the memory constraints and performance requirements of these use cases. The reference software implementations of these schemes are further profiled, and based on the profiling result, new sets of hardware accelerators are recommended to be integrated in PQC HSMs targeting *medium* and *high* security levels respectively. The evaluation results of the PQC HSMs compared with modern HSMs show that automotive HSMs can remain future-proof by using post-quantum algorithms to ensure their security without much overhead in terms of designing cryptographic hardware accelerators.

References

[1] B. Weyl, M. Wolf, F. Zweers, T. Gendrullis, M. S. Idrees, Y. Roudier, H. Schweppe, H. Platzdasch, R. El Khayari, O. Henniger, *et al.*, "Secure on-board architecture specification," *Evita Deliverable D*, vol. 3, p. 2, 2010.

[2] 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.

[3] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Symposium on the Theory of Computing (STOC)*, pp. 212–219, ACM, 1996.

[4] J. Buchmann, E. Dahmen, and M. Schneider, "Merkle tree traversal revisited," in *Post-Quantum Cryptography (PQCrypto)* (J. Buchmann and J. Ding, eds.), vol. 5299 of *LNCS*, pp. 63–78, Springer, 2008.

[5] T. Fritzmann, U. Sharif, D. Müller-Gritschneder, C. Reinbrecht, U. Schlichtmann, and J. Sepulveda, "Towards reliable and secure post-quantum co-processors based on risc-v," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1148–1153, IEEE, 2019.

[6] W. Wang, B. Jungk, J. Wälde, S. Deng, N. Gupta, J. Szefer, and R. Niederhagen, "Xmss and embedded systems-xmss hardware accelerators for risc-v.," *IACR Cryptology ePrint Archive*, vol. 2018, p. 1225, 2018.

[7] R. J. McEliece, "A public-key cryptosystem based on algebraic," *Coding Thv*, vol. 4244, pp. 114–116, 1978.

[8] N. Aragon, P. Barreto, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, S. Gueron, T. Guneysu, C. A. Melchor, *et al.*, "Bike: bit flipping key encapsulation," 2017.

[9] Q. Guo, T. Johansson, and P. Stankovski, "A key recovery attack on mdpc with cca security using decoding errors," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 789–815, Springer, 2016.

[10] A. Hülsing, D. Butin, S. Gazdag, J. Rijneveld, and A. Mohaisen, "XMSS: eXtended Merkle Signature Scheme," *RFC*, vol. 8391, pp. 1–74, 2018.

[11] D. McGrew, M. Curcio, and S. Fluhrer, "Leighton-micali hash-based signatures," *RFC*, vol. 8554, pp. 1–61, 2019.

[12] M. Naehrig, E. Alkim, J. Bos, L. Ducas, K. Easterbrook, B. LaMacchia, P. Longa, I. Mironov, V. Nikolaenko, C. Peikert, A. Raghunathan, and D. Stebila, "FrodoKEM," tech. rep., National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-2/submissions/FrodoKEM-Round2.zip.

[13] N. Bindel, S. Akleylek, E. Alkim, P. S. L. M. Barreto, J. Buchmann, E. Eaton, G. Gutoski, J. Kramer, P. Longa, H. Polat, J. E. Ricardini, and G. Zanon, "qTESLA," tech. rep., National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-2/submissions/qTESLA-Round2.zip.

[14] D. Coppersmith and A. Shamir, "Lattice attacks on ntru," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 52–61, Springer, 1997.

[15] J. Ding, M.-S. Chen, A. Petzoldt, D. Schmidt, and B.-Y. Yang, "Rainbow," tech. rep., National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-2/submissions/Rainbow-Round2.zip.

[16] W. Beullens, B. Preneel, A. Szepieniec, and F. Vercauteren, "LUOV," tech. rep., National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-2/submissions/LUOV-Round2.zip.

[17] A. Rostovtsev and A. Stolbunov, "Public-key cryptosystem based on isogenies.," *IACR Cryptology ePrint Archive*, vol. 2006, p. 145, 2006.

[18] V. Lyubashevsky, L. Ducas, E. Kiltz, T. Lepoint, P. Schwabe, G. Seiler, and D. Stehle, "CRYSTALS-DILITHIUM," tech. rep., National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-2/submissions/CRYSTALS-Dilithium-Round2.zip.

[19] N. Aragon, O. Blazy, J.-C. Deneuville, P. Gaborit, A. Hauteville, O. Ruatta, J.-P. Tillich, G. Zemor, C. A. Melchor, S. Bettaieb, L. Bidoux, M. Bardet, and A. Otmani, "ROLLO," tech. rep., National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-2/submissions/ROLLO-Round2.zip.

[20] J. Buchmann, E. Dahmen, and A. Hülsing, "Xmss-a practical forward secure signature scheme based on minimal security assumptions," in *International Workshop on Post-Quantum Cryptography*, pp. 117–129, Springer, 2011.

[21] P. Schwabe, R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. L. andVadim Lyubashevsky, J. M. Schanck, G. Seiler, and D. Stehle, "CRYSTALS-KYBER," tech. rep., National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-2/submissions/CRYSTALS-Kyber-Round2.zip.

[22] P. Barrett, "Implementing the rivest shamir and adleman public key encryption algorithm on a standard digital signal processor," in *Conference on the Theory and Application of Cryptographic Techniques*, pp. 311–323, Springer, 1986.
[23] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of computation*, vol. 44, no. 170, pp. 519–521, 1985.
[24] M. J. Dworkin, "Sha-3 standard: Permutation-based hash and extendable-output functions," tech. rep., 2015.
[25] B. Jungk, *FPGA-based evaluation of cryptographic algorithms.* PhD thesis, Goethe University Frankfurt am Main, 2016.
[26] W. Wang, J. Szefer, and R. Niederhagen, "Solving large systems of linear equations over gf (2) on fpgas," in *2016 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pp. 1–7, IEEE, 2016.