Post-Quantum Secure Over-the-Air Update of Automotive Systems

Joppe W. Bos[1][0000−0003−1010−8157], Alexander Dima[2], Alexander Kiening[3][0000−0003−2161−8662], and Joost Renes[4][0000−0003−1884−6330]

[1] NXP Semiconductors, Leuven, Belgium
joppe.bos@nxp.com
[2] PiNTeam GmbH, Germany
alexander.dima@pinteam.eu
[3] DENSO AUTOMOTIVE Deutschland GmbH, Germany
a.kiening@eu.denso.com
[4] NXP Semiconductors, Eindhoven, Netherlands
joost.renes@nxp.com

Abstract. With the announcement of the first winners of the NIST Post-Quantum Cryptography (PQC) competition in 2022, the industry has now a confirmed foundation to revisit established cryptographic algorithms applied in automotive use cases and replace them with quantum-safe alternatives. In this paper, we investigate the application of the NIST competition winner CRYSTALS-Dilithium to protect the integrity and authenticity of over-the-air update packages. We show how this post-quantum secure digital signature algorithm can be integrated in AUTOSAR Adaptive Platform Update and Configuration Management framework and evaluate our approach practically using the NXP S32G vehicle network processor. We discuss two implementation variants with respect to performance and resilience against relevant attacks, and conclude that PQC has little impact on the update process as a whole.

Keywords: Post-Quantum Cryptography · Over-the-Air Update · Migration.

1 Introduction

Digital signatures are one of the core cryptographic building blocks in modern digital security. The goal of this concept, invented by Diffie and Hellman [13], is to provide message authentication against the public key of a sender. Applications of digital signatures are countless and diverse. In this work we look at the application of digital signatures in automotive applications where secure boot and secure (over-the-air) update are two main applications to protect against modifications of the software or firmware with potentially malicious intentions.

In the last decades multiple digital signature designs have been standardized. These approaches are either based on Elliptic Curve Cryptography (ECC) [31,39] or the Rivest–Shamir–Adleman (RSA) [48] algorithm. However, assuming the

availability of a large-scale quantum computer the security of these "classical" approaches is threatened by Shor's *quantum* algorithm [50] which is able to recover such ECC/RSA private keys in polynomial time. To prepare for this quantum threat, alternative public-key algorithms are necessary. These are typically referred to as *post-quantum* or *quantum-safe* algorithms. The new algorithms are intended to run on classical hardware yet provide sufficient protection even against adversaries that are in possession of a quantum computer. They are not to be confused with *quantum* cryptography such as Quantum Key Distribution (QKD), where the cryptographic algorithms also run on infrastructure (partially) consisting of quantum computers.

Recognizing the threat of quantum computer, the US National Institute of Standards and Technology (NIST) put out a call for proposals [42] in 2016 to submit candidates for a new standards consisting of post-quantum secure algorithms. In July 2022, NIST has recommended one primary algorithm for digital signatures: CRYSTALS-Dilithium [14,35] (denoted simply as Dilithium in the remainder of this paper) In addition, two other signature schemes will also be standardized: Falcon [45] and SPHINCS+ [5,23]. The final standard for Dilithium is expected to be published in 2024, which will deviate slightly from the original Dilithium proposal due to comments from industry and academia that were received as part of the standardization process. The NIST standardized algorithms are also receiving interest from European standardization and norms bodies. For example, the German Federal Office for Information Security (BSI) will consider including Dilithium into their Technical Guidelines for Cryptographic Mechanisms TR-02102-1 [15, Section 4] after the final NIST standard has been published.

In this paper, we investigate the practical impact of Dilithium in one of the key automotive use-cases: Over-The-Air (OTA) update. This is done from a Tier-1 perspective where all the communication cost from host processor to the secure processor is taken into account. For our experiment, we use the S32G vehicle network processor and compare running the post-quantum secure signature verification in two settings. Firstly, the verification is performed on the microprocessor which is based on the Arm Cortex-A53 and secondly, on the microcontroller which is part of the Hardware Security Engine (HSE) and based on the Arm Cortex-M7. The A53 is larger and faster compared to the M7 but does not offer security features to protect key-material or against advanced attacks such as fault injection [8,7].

2 Background and Related Work

2.1 Post-Quantum Digital Signatures

In this work we follow the recommendation from NIST and focus on *lattice-based* digital signatures. This area in cryptography has a rich history and a wide variety of options relying on different hardness assumptions. One approach in lattice-based cryptography is based on Regev's work introducing the Learning With Errors (LWE) problem [47], which relates to solving a "noisy" linear system

Table 1. Public key and signature sizes for each security level of Dilithium in bytes.

Security level	Public key	Signature
Dilithium-2	1 312	2 420
Dilithium-3	1 952	3 293
Dilithium-5	2 592	4 595

modulo a known integer. This problem can be used as the basis for a signature scheme, as shown by Lyubashevsky [34], by improving on his idea to apply Fiat-Shamir with aborts [33] to lattices. A more specialized version is based on the Ring Learning With Errors (R-LWE) problem [36,43], which works in a special ring (more specifically the ring of integers of a cyclotomic number field) that offers significant storage and efficiency improvements compared to LWE. Although R-LWE has additional algebraic structure and relies on the (worst-case) hardness of problems in *ideal* lattices, no significant concrete improvements in cryptanalysis are known. Finally, a combination of many of these ideas (plus various improvements) resulted in CRYSTALS–Dilithium [14] based on something which is known as *Module*-LWE. Table 1 gives an overview of the public-key and signature sizes for the three Dilithium parameter sets.

An alternative direction to realize post-quantum secure digital signatures is that of *hash-based* signatures. Two algorithms from this realm are known as eXtended Merkle Signature Scheme (XMSS) and Leighton-Micali Signatures [38] (LMS) and have been established as Requests For Comments (RFCs) by the Internet Engineering Task Force (IETF) [24,38]. Recently, they have been published as a NIST Special Publication 800-208 [12] as well. These approaches are based on well-established cryptographic primitives (i.e., hash functions) and have efficient signature verification times. The main downside is that both signature schemes are *stateful* on the end of the signer, which can seriously complicate key management. Their *stateless* counterpart SPHINCS+ [22] solves this issue, but at the cost of significantly increasing the signature size. Other alternatives explore digital signatures based on the hardness of multivariate quadratic equations, error-correcting codes and isogenies. Unfortunately either their keys or signatures are extremely large, or they are relatively inefficient, making them difficult to apply in embedded scenarios.

In June 2023 a new standardization process was started by NIST with the focus on small digital signature schemes. At the time of writing the submission deadline had passed, but the proposals had not been made public yet. This could lead to new, smaller signature schemes that could outperform Dilithium or hash-based signatures. Having that said, it will take a number of years for the proposals to be analyzed and finally published into a standard.

2.2 PQClean

The NIST standardization effort required inclusion of reference implementations in pure C, to demonstrate the correctness and efficiency of the proposed algo-

rithms. As there were no requirements with respect to software engineering standards, the code quality varied wildly and was often not directly fit for integration into higher level protocols. This observation led Kannwischer, Schwabe, Stebila and Wiggers [30] to develop *PQClean*, a collection of *clean* implementations of the NIST PQC proposals in pure C.[5] For example, any implementation included in PQClean should check that code is valid in C99, passes functional tests, does not write outside provided buffers, etc. For a full list of requirements we refer to the Github repository.

Besides the clean implementations, PQClean also includes optimized implementations for targeted architectures depending on the algorithm. It includes optimized Dilithium implementations for modern processors with support for AVX2, and for the 64-bit Arm architecture family AArch64. The latter is part of the ARMv8-A instruction set that is implemented in the Cortex-A53. We use this optimized implementation to run Dilithium on the S32G host processors.

2.3 AUTOSAR

The AUTomotive Open System ARchitecture (AUTOSAR)[6] is a consortium of automotive companies and other interested parties to specify a harmonized architecture and API for automotive middleware vendors to implement. These middlewares offer typical services to automotive applications such as communication, logging, and diagnostics. AUTOSAR specifies two architectures for different kind of Electronic Control Unit (ECUs): the Classic Platform to support applications with real-time requirements, and the Adaptive Platform for high-performance applications running on a POSIX-compabtible operating system such as Linux or QNX [1]. Both Classic and Adaptive Platforms are designed to interact in an in-vehicle network.

For the Adaptive Platform, the AUTOSAR partners are developing a common Adaptive AUTOSAR Demonstrator (APD) to validate the specifications. This demonstrator can be used by partners for their own proof of concept developments. We used this APD as basis for our practical evaluation later.

2.4 S32G Vehicle Network Processors

We target an *S32G vehicle network processor* as the platform of choice for the impact assessment of integrating post-quantum cryptography in the over-the-air update protocol. This high-end automotive processor is developed by NXP Semiconductors and part of a larger S32 automotive platform which includes the S32K, S32R and S32V and is designed to meet the safety and security requirements in the automotive and industrial domains (i.e., compliance with IEC 61508 [25] and ASIL-D classification in ISO 26262 [26]). Typical uses include service-oriented gateways, domain controllers, vehicle computers and safety processors. The S32G consists of a combination of microcontrollers (MCUs) based

[5] https://github.com/PQClean/PQClean
[6] https://www.autosar.org

on the Arm Cortex-M7, and microprocessors (MPUs) based on Arm Cortex-A53. These application CPUs are combined with several types of memory (SRAM, DRAM, NOR/NAND Flash) and various hardware accelerators.

The precise configuration depends on the choice of model: we deploy the S32G274A which contains 3 Arm Cortex-M7 cores, 4 Arm Cortex-A53 cores, and 8 MB of system RAM. Each of the MCUs runs in a delayed lockstep configuration at a maximum frequency of 400 MHz and has 32 KB instruction and data caches. The MPUs are configured as 2 clusters of 2 cores each running at a maximum frequency of 1 GHz. Every core has access to 32 KB L1 instruction and data caches, while each cluster shares another 512 KB of L2 cache. Optionally, the A53 clusters can be configured to also run in a delayed lockstep setting, effectively removing one of the clusters from an application's point of view but increasing the fault tolerance.

Most notably, the processor contains a Hardware Security Engine (HSE) which supports both symmetric and (classical) asymmetric cryptography accelerators, a random number generator, and dedicated secure memory. The HSE is responsible for the boot flow if secure boot is enabled as well as serving as the Root of Trust (RoT) for host applications: in our setting computing the post-quantum secure signature verification. The HSE Firmware (HSE-FW) has been extended by Bos et al. [9, Section 4.1] to include Dilithium into the signature verification service. This is a high-secure low-memory software implementation which has protection against fault attacks, and supports all parameter sets of Dilithium v3.1.

2.5 Related work

A generic approach to reduce the memory consumption of Dilithium on constrained devices is presented by Bos, Renes, and Sprenkels in [10]. In the automotive domain there have been investigations on applying post-quantum cryptography to the setting of secure boot. An impact assessment of hash-based post-quantum secure schemes on secure boot is studied by Kampanakis, Panburana, Curcio and Shroff [27]. Hermelink, Pöppelmann, Stöttinger, Wang and Wan perform an investigation into Authenticated Key Exchange (AKE) combining different post-quantum cryptographic schemes [20]. In [16], Feritzmann, Vith, Flórez and Sepúlveda analyze lattice-based Key Encapsulation Mechanisms (KEMs) for automotive systems. An investigation of the practical impact of migrating the secure boot flow on a Vehicle Network Processor using Dilithium is done by Bos, Carlson, Renes, Rotaru, Sprenkels, and Waters in [9].

The other use-case is secure over-the-air software update. A survey on this topic in connected vehicles is performed by Halder, Ghosal, and Conti [18]. Some benchmark results using the post-quantum secure schemes Dilithium and Falcon in the setting of OTA update on a Arm Cortex-A53 are presented by Manna, Perazzo, Treccozzi, and Dini in [37]. In [3], Banegas, Zandberg, Baccelli, Herrmann, and Smith investigate low-power software update with the Software Updates for Internet of Things (SUIT) [40,41] specification focussing on low-power IoT devices.

3 Over-the-Air Update

Over-the-air updates include services like software over-the-air (SOTA), firmware over-the-air (FOTA), and over-the-air provisioning (OTAP). To implement this, the AUTOSAR Adaptive Platform offers the specification of the functional cluster "Update and Configuration Management" (UCM) which is responsible for handling the vehicle side of the update process. This includes the following steps of the components shown in Figure 1:

1. Download the update package by the UCM client;
2. Check the package's authentification tag (e.g., digital signature);
3. Evaluate the package's manifest;
4. Interact with state management;
5. Apply the update.

In this section we describe the considerations we have taken for the implementation.

3.1 OTA Update Security

The verification of the authentication tag (i.e., step 2) is the most important step for our work. There are several ways in which a malicious entity can subvert the integrity of the package. Firstly, they can attempt to obtain a digital signature over a malicious package, which would pass the authentication check in step 2. This could be done by comprimising the signing infrastructure and having direct access to the (secret) signing key, but for the purpose of this work such attacks are considered out of scope as they cannot be thwarted by countermeasures on the processor itself. Alternatively, the signing key can be retrieved by breaking the public-key cryptographic scheme that is deployed. This attack is a realistic scenario for a quantum adversary, as it allows them to efficiently obtain the signing key for classical systems such as RSA or ECDSA. Indeed, for this reason timely migration to post-quantum alternatives is critical. In this work we deal with this attack vector by relying on the quantum security of Dilithium.

Assuming that no signature can be created over a malicious package, a malicious entity could instead try to compromise the integrity of the public key. In that case the verification in step 2 would pass, but would verify authenticity with respect to an attacker controlled public key for which they can sign arbitrary packages. To trust the authenticity of update packages, strong protection on the public key is therefore required. For example, this can be achieved by storing the (hash of the) public key in Read-Only Memory (ROM) such as fuses.

Finally, the runtime execution of the verification of the authentication tag could be compromised. It is well known that such advanced security threats should be considered in the automotive domain [32]. One category of such a relevant security threat for signature verification are *active attacks*. An example of such an attack is fault analysis [8] and more specifically differential fault analysis (DFA) [7]. Such types of attacks work by utilizing under-powering and power

Table 2. Threats and assets for signature verification.

#	Threat	Targeted Asset
1	Compromise the signing infrastructure	Secrecy of the signing key
2	Change the key via manipulated software on the ECU	Integrity of the public key
3	Change the key via physical access to the ECU	
4	Change the execution of the cryptographic algorithms via manipulated software on the ECU	Integrity of the algorithm execution
5	Change the execution of the cryptographic algorithm via physical access to the ECU	

spikes, clock gitches, temperature attacks, optical attacks or electromagnetic injection to introduce faults during the execution. If some operations are skipped or performed incorrectly during the execution process, then this should lead to an incorrect calculation of the output which could be used to learn something about the secret key material used. When using signature verification no secret key material is used. However, a simple attack would be to fault the control if-statement if a signature is correct to always yield true. This would lead to unauthorized updates with invalid signatures to be installed on the target device. In the setting of Dilithium a survey of both passive and active attacks is given in [46].

We summarize an overview of the threats in Table 2. Although there are different ways to achieve security against these attack vectors, ultimately it boils down to a well-defined *Root of Trust* (RoT). The expected capabilities of RoTs are outlined for example by TCG [11, Part 1 §9.5.5] or GlobalPlatform [17]. Its implementation has to conform with the latest security standards and certifications (e.g., ISO 26262). Typically, the RoT comes in the form of a Hardware Security Module (HSM) or Hardware Security Engine (HSE).

3.2 OTA Update Implementation

As shown in Figure 1, cryptographic algorithms can be implemented in ECUs in two ways: in an HSM or in a software library executed directly on a host application core. While the HSM approach generally offers better protection mechanisms especially against physical attacks (see Section 3.1), it is also typically less flexible than executing the software on the host processor. Which approach to choose depends on performance requirements and the risk treatment decisions based on the Threat Analysis and Risk Assessment (TARA).

To implement support for digital signature verification, UCM can use the services offered by the CryptoAPI, which abstracts interaction with concrete cryptographic implementations by means of Crypto Providers. Crypto Providers are secure wrappers of cryptographic software libraries or secure element host drivers. They serve two purposes: standardizing user interaction and securing

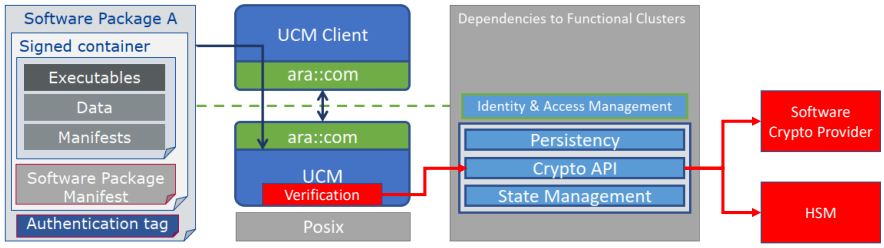

Fig. 1. Update package and UCM dependencies to other Functional Clusters [2].

crypto objects such as secret key material, secret seed material or special public keys. Ideally, to comply with the AUTOSAR architecture, we would add a new Crypto Provider for the HSE Linux host driver. Unfortunately, the Adaptive Platform Demonstrator does not feature a sufficiently up-to-date reference implementation of the CryptoAPI. Therefore, we decided to integrate the PQC-enhanced HSE Linux host driver with UCM and use low-level driver interfaces directly to perform the signature verification.

3.3 OTA Update Performance Requirements

It is clear that performance is critical for Over-the-Air updates. However, the performance issues do not come from executing the cryptographic operations but arise when scheduling OTA software updates at a massive scale over a cellular network (like 4G LTE to 5G). This becomes clear from the standardization survey for over-the-air updating in vehicles [19] where no standards mention specific latency requirements for the signature verification but the challenges of deploying large scale updates over the cellular network is mentioned over-and-over again. Security, on the other hand, is highlighted at many places. This is of course not a surprise since safety and security are key requirements in an update protocol: this reduces, for example, the risk to remotely install malware in the vehicle. A survey on security attacks and defense techniques for connected and autonomous vehicles can be found in [44].

In short, a (small) performance loss when migrating towards post-quantum cryptography is not expected to have an impact on the end user experience as it will be negligible in the overall update time. As we shall see in Section 4, this holds true for both the implementation in the HSM/HSE as well as on the host processor. From a security point of view, it therefore makes the HSE the preferred way of executing the signature verification. The host processor would be the preferred option only if the additional flexibility is more important than the loss in security guarantees.

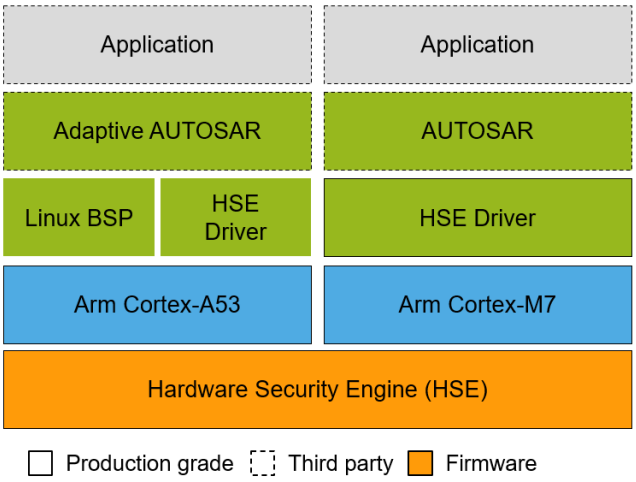

Fig. 2. The S32G2 processors for vehicle networking software ecosystem.

4 Benchmark results and Discussion

In this section we elaborate on the practical details of the PQC secure OTA demonstrator, and discuss the pros and cons of the different options.

4.1 Development Setup

For development we used the S32G Vehicle Networking Evaluation Board (VNP-EVB) and Vehicle Networking Reference Design (VNP-RDB2). Both function virtually identically for our purposes, the latter being the more recent and recommended development board (though now already superseded by the RDB3 board). These boards come equipped with a Linux Board Support Package (BSP) for the Cortex-A53 MPUs that we used as a basis for integrating Dilithium into our software. The BSP provides Linux User IO (UIO) drivers and a messaging module (`libhse`) that provides direct low-level access to the HSE. See Figure 2 for an overview of the S32G2 software ecosystem. The use of Dilithium signature verification is almost completely transparent for a user: as is the case for ECC or RSA, a message, public key and signature are provided to a signature service in which the signing scheme can be specified. As part of the integration of Dilithium into the firmware, the signing scheme can simply be set to Dilithium instead of the classical choices of ECC or RSA. We refer to Figure 3 for the control flow when executing Dilithium signature verification on the HSE.

4.2 Executing the Cryptographic Algorithms

As explained in Section 2.4, our platform contains a microprocessor based on the Arm Cortex-A53 running at 1000 MHz which we will refer to as the host. The

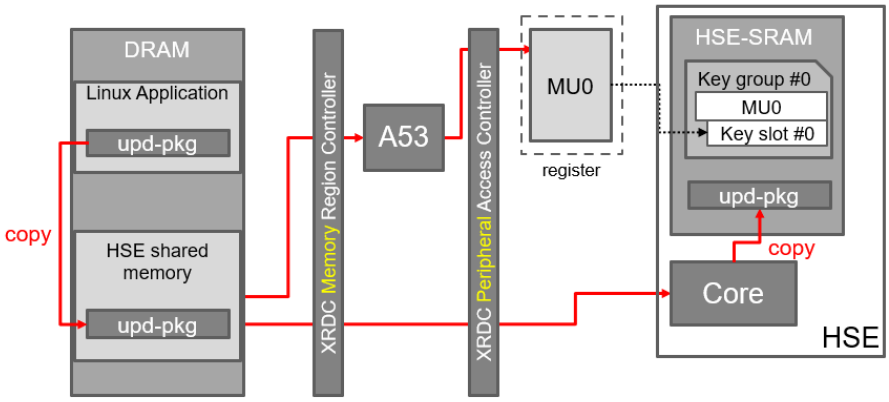

Fig. 3. Overview of delivering data to the HSE in order to execute its functionality.

Hardware Security Engine utilizes a microcontroller based on the Arm Cortex-M7 running at 400 MHz. As the Cortex-A53 has a much more extensive architecture than the Cortex-M7, it is no surprise that most software can be executed significantly faster. This is especially the case as it runs at a much higher frequency. The difference in performance can typically be offset by additional cryptographic hardware support in the HSE, but this is not yet available for the S32G2. For future platforms (e.g., S32G3 with SHA3 acceleration), the difference between the host and the HSE will favor the HSE much more.

4.3 Results

It is clear from Section 3.3 that the execution time of the signature verification algorithm typically is *not* the bottleneck in the setting of OTA update for automotive applications. However, to assess the impact of the ongoing migration towards post-quantum cryptography we quantify the impact of switching to the usage of Dilithium-2 signature verification. This impact is measured in the form of the latency of the verification algorithm.

Remark 1. The latency is *not* solely determined by the choice of signature scheme itself, but also by the length of the software update. Typically, all digital signature verification schemes *pre-hash* the variable-length input to a fixed-size digest, possibly including miscellaneous other data such as padding, a public key, a commitment, etc. Afterwards the digest is processed to create or check the final digital signature. This step is independent of the signature scheme chosen but the exact the choice of the hash function does differ in practice. For example, the ECDSA [51] signature scheme starts by applying a FIPS 180 [49] compliant hash function (e.g., SHA-256) to the input message. Dilithium signs (and verifies) arbitrary-length messages by hashing them together with the public key, using Keccak [6] (the permutation underlying SHA-3). The choice of hash

function has no impact on the security of the public-key signature scheme, it can, however, have significant impact on the performance in practice. For example, the S32G274A used offers hardware support for SHA-2 and not for (variants of) SHA-3. Therefore we focus on the setting where a SHA-512 hash over the software is signed instead and subsequently verified.

We adjusted the OTA update protocol such that it uses the post-quantum secure Dilithium algorithm. For our experiments we utilized the Dilithium-2 parameter set: this means a public-key of 1 312 bytes and verifying a signature of 2 420 bytes (see Table 1). The 64-byte pre-hashed software update together with the signature is validated using the Dilithium signature verification algorithm. As explained in Section 3.2 two approaches are considered: running the verification on the host processor itself and executing the Dilithium verification on the HSE. Our benchmark figures include all overhead required to run the signature verification: i.e., the timer starts and stops at the host processor before and after the complete signature verification. All together, the Dilithium-2 verification time is 0.5ms on the host processor and 11.1ms on the HSE.

4.4 Discussion

As seen from Section 4.3, the latency on the host processor is lower than on the HSE. This difference in performance is explained by the fact that the host processor and the HSE differ significantly in terms of the offered performance: e.g., the instruction set as well as the clock speed of both platforms differ significanly. For example, if we compare the Dilithium-2 verification results on a Cortex-M7 by [21] to the ones from [4] on a Cortex-A72 then one already sees a difference of 5–6 times. Hence, if we also take the difference in clock-speed into account (another factor of 2.5×) one expect that the host processor is around 14 times faster. However, we stress again that the host processor is unprotected against advanced attacks relevant in the automotive domain (see Section 3.1). The HSE, on the other hand, offers much more security features compared to the host processor. This includes protection against side-channel and fault attacks. The unprotected implementation in [21] requires 1 439 kilo cycles, measured on STM32F767ZI NUCLEO-144 development board. At 200 MHz this corresponds to about 7.1ms. This is faster than our implementation, but can be explained by the fact that the HSE includes fault protection and applies more aggressive time-memory trade-offs that impact performance further. Whereas the runtime RAM memory required in [21] is about 36 kB for Dilithium-2, the HSE requires less than 3 kB.

Looking ahead, much of the 11.1ms in the HSE signature verification is spent in the software implementation of the SHAKE-256 algorithm used in Dilithium verification. This does not come as a surprise as the embedded benchmarking platform PQM4 [28,29] (running on a ARM Cortex-M4) reports that over 80 percent of the verificaton time is spent in SHA3. Future generations (e.g., S32G3) will have dedicated SHA3 hardware available, pushing the performance to be

very close to the host processor. Nevertheless, in the context of a full over-the-air software update, a difference of 10ms in verification latency is negligible and will have no impact on the experience of the vehicle owner. Therefore the security features offered by the HSE significantly outweigh the small difference in performance.

References

1. AUTOSAR: Explanation of adaptive platform design. https://www.autosar.org/fileadmin/standards/R22-11/AP/AUTOSAR_EXP_SWArchitecture.pdf (2022)
2. AUTOSAR: Specification of update and configuration management. https://www.autosar.org/fileadmin/standards/R22-11/AP/AUTOSAR_SWS_UpdateAndConfigurationManagement.pdf (2022)
3. Banegas, G., Zandberg, K., Baccelli, E., Herrmann, A., Smith, B.: Quantum-resistant software update security on low-power networked embedded devices. In: Ateniese, G., Venturi, D. (eds.) Applied Cryptography and Network Security. pp. 872–891. Springer International Publishing (2022). https://doi.org/10.1007/978-3-031-09234-3_43
4. Becker, H., Hwang, V., Kannwischer, M.J., Yang, B.Y., Yang, S.Y.: Neon NTT: Faster Dilithium, Kyber, and Saber on Cortex-A72 and Apple M1. IACR TCHES **2022**(1), 221–244 (2022). https://doi.org/10.46586/tches.v2022.i1.221-244
5. Bernstein, D.J., Hülsing, A., Kölbl, S., Niederhagen, R., Rijneveld, J., Schwabe, P.: The SPHINCS$^+$ signature framework. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 2129–2146. ACM Press (Nov 2019). https://doi.org/10.1145/3319535.3363229
6. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Keccak. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 313–314. Springer, Heidelberg (May 2013). https://doi.org/10.1007/978-3-642-38348-9_19
7. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Kaliski Jr., B.S. (ed.) CRYPTO'97. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (Aug 1997). https://doi.org/10.1007/BFb0052259
8. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults (extended abstract). In: Fumy, W. (ed.) EUROCRYPT'97. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (May 1997). https://doi.org/10.1007/3-540-69053-0_4
9. Bos, J.W., Carlson, B., Renes, J., Rotaru, M., Sprenkels, A., Waters, G.P.: Post-quantum secure boot on vehicle network processors. In: 20th escar Europe – The World's Leading Automotive Cyber Security Conference (2022). https://doi.org/10.13154/294-9372
10. Bos, J.W., Renes, J., Sprenkels, A.: Dilithium for memory constrained devices. In: Batina, L., Daemen, J. (eds.) AFRICACRYPT 22. LNCS, vol. 2022, pp. 217–235. Springer Nature (Jul 2022). https://doi.org/10.1007/978-3-031-17433-9_10
11. Challener, D., Goldman, K.: Trusted Platform Module Library Specification, Family "2.0", Level 00, Revision 01.59 (2019), https://trustedcomputinggroup.org/work-groups/trusted-platform-module/
12. Cooper, D., Apon, D., Dang, Q., Davidson, M., Dworkin, M., Miller, C.: Recommendation for stateful hash-based signature schemes. SP 800-208, National Institute of Standards and Technology (2020)

13. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory **22**(6), 644–654 (1976). https://doi.org/10.1109/TIT.1976.1055638
14. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Dilithium: A lattice-based digital signature scheme. IACR TCHES **2018**(1), 238–268 (2018). https://doi.org/10.13154/tches.v2018.i1.238-268, https://tches.iacr.org/index.php/TCHES/article/view/839
15. Federal Office for Information Security (BSI): Technical guidelines for cryptographic mechanisms (tr-02102). https://www.bsi.bund.de/EN/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr02102/tr02102_node.html
16. Fritzmann, T., Vith, J., Flórez, D., Sepúlveda, J.: Post-quantum cryptography for automotive systems. Microprocessors and Microsystems **87**, 104379 (2021). https://doi.org/https://doi.org/10.1016/j.micpro.2021.104379, https://www.sciencedirect.com/science/article/pii/S0141933121005299
17. GlobalPlatform Technology: Root of Trust Definitions and Requirements Version 1.1 (GP_REQ_025) (2018), https://globalplatform.org/specs-library/globalplatform-root-of-trust-definitions-and-requirements/
18. Halder, S., Ghosal, A., Conti, M.: Secure over-the-air software updates in connected vehicles: A survey. Computer Networks **178**, 107343 (2020). https://doi.org/10.1016/j.comnet.2020.107343, https://www.sciencedirect.com/science/article/pii/S1389128619314963
19. Halder, S., Ghosal, A., Conti, M.: Secure over-the-air software updates in connected vehicles: A survey. Computer Networks **178**, 107343 (2020)
20. Hermelink, J., Pöppelmann, T., Stöttinger, M., Wang, Y., Wan, Y.: Quantum safe authenticated key exchange protocol for automotive application. In: 18^{th} escar Europe : The World's Leading Automotive Cyber Security Conference (Konferenzveröffentlichung) (2020). https://doi.org/10.13154/294-7549
21. Howe, J., Westerbaan, B.: Benchmarking and analysing the NIST PQC finalist lattice-based signature schemes on the ARM cortex M7. Cryptology ePrint Archive, Report 2022/405 (2022), https://eprint.iacr.org/2022/405
22. Hülsing, A., Bernstein, D.J., Dobraunig, C., Eichlseder, M., Fluhrer, S., Gazdag, S.L., Kampanakis, P., Kölbl, S., Lange, T., Lauridsen, M.M., Mendel, F., Niederhagen, R., Rechberger, C., Rijneveld, J., Schwabe, P., Aumasson, J.P., Westerbaan, B., Beullens, W.: SPHINCS$^+$. Tech. rep., National Institute of Standards and Technology (2020), available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions
23. Hülsing, A., Bernstein, D.J., Dobraunig, C., Eichlseder, M., Fluhrer, S., Gazdag, S.L., Kampanakis, P., Kölbl, S., Lange, T., Lauridsen, M.M., Mendel, F., Niederhagen, R., Rechberger, C., Rijneveld, J., Schwabe, P., Aumasson, J.P., Westerbaan, B., Beullens, W.: SPHINCS$^+$. Tech. rep., National Institute of Standards and Technology (2022), available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022
24. Hülsing, A., Butin, D., Gazdag, S.L., Rijneveld, J., Mohaisen, A.: XMSS: Extended Hash-Based Signatures. RFC 8391 (2018)
25. International Electrotechnical Commission (IEC): Functional safety of electrical/electronic/programmable electronic safety-related systems. IEC 61508 (2010)
26. International Organization for Standardization (ISO): Road vehicles - functional safety. ISO 26262 (2018)

27. Kampanakis, P., Panburana, P., Curcio, M., Shroff, C.: Post-quantum hash-based signatures for secure boot. In: Silicon Valley Cybersecurity Conference. Springer (2020). https://doi.org/10.1007/978-3-030-72725-3
28. Kannwischer, M.J., Rijneveld, J., Schwabe, P., Stoffelen, K.: PQM4: Post-quantum crypto library for the ARM Cortex-M4, https://github.com/mupq/pqm4
29. Kannwischer, M.J., Rijneveld, J., Schwabe, P., Stoffelen, K.: pqm4: Testing and Benchmarking NIST PQC on ARM Cortex-M4. Workshop Record of the Second PQC Standardization Conference (2019), https://cryptojedi.org/papers/#pqm4
30. Kannwischer, M.J., Schwabe, P., Stebila, D., Wiggers, T.: Improving Software Quality in Cryptography Standardization Projects. In: IEEE European Symposium on Security and Privacy, EuroS&P 2022 - Workshops, Genoa, Italy, June 6-10, 2022. pp. 19–30. IEEE Computer Society, Los Alamitos, CA, USA (2022). https://doi.org/10.1109/EuroSPW55150.2022.00010, https://eprint.iacr.org/2022/337
31. Koblitz, N.: Elliptic curve cryptosystems. Mathematics of Computation **48**, 203–209 (1987)
32. Lemke, K., Paar, C., Wolf, M.: Embedded security in cars. Springer (2006)
33. Lyubashevsky, V.: Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 598–616. Springer, Heidelberg (Dec 2009). https://doi.org/10.1007/978-3-642-10366-7_35
34. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_43
35. Lyubashevsky, V., Ducas, L., Kiltz, E., Lepoint, T., Schwabe, P., Seiler, G., Stehlé, D., Bai, S.: CRYSTALS-DILITHIUM. Tech. rep., National Institute of Standards and Technology (2022), available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022
36. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (May / Jun 2010). https://doi.org/10.1007/978-3-642-13190-5_1
37. Manna, M.L., Perazzo, P., Treccozzi, L., Dini, G.: Assessing the cost of quantum security for automotive over-the-air updates. In: 2021 IEEE Symposium on Computers and Communications (ISCC). pp. 1–6 (2021). https://doi.org/10.1109/ISCC53001.2021.9631426
38. McGrew, D.A., Curcio, M., Fluhrer, S.R.: Hash-Based Signatures. RFC 8554, RFC Editor (04 2019), https://www.rfc-editor.org/rfc/rfc8554.txt
39. Miller, V.S.: Use of elliptic curves in cryptography. In: Williams, H.C. (ed.) CRYPTO'85. LNCS, vol. 218, pp. 417–426. Springer, Heidelberg (Aug 1986). https://doi.org/10.1007/3-540-39799-X_31
40. Moran, B., Tschofenig, H., Birkholz, H., Zandberg, K., Øyvind Rønningstad: A concise binary object representation (CBOR)-based serialization format for the software updates for internet of things (SUIT) manifest. Internet-Draft draft-ietf-suit-manifest-22, Internet Engineering Task Force, https://datatracker.ietf.org/doc/draft-ietf-suit-manifest/22/, work in Progress
41. Moran, B., Tschofenig, H., Brown, D., Meriac, M.: A firmware update architecture for internet of things. RFC 9019 (2021). https://doi.org/10.17487/RFC9019, https://www.rfc-editor.org/info/rfc9019

42. National Institute of Standards and Technology: Post-quantum cryptography standardization. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization
43. Peikert, C., Regev, O., Stephens-Davidowitz, N.: Pseudorandomness of ring-LWE for any ring and modulus. In: Hatami, H., McKenzie, P., King, V. (eds.) 49th ACM STOC. pp. 461–473. ACM Press (Jun 2017). https://doi.org/10.1145/3055399.3055489
44. Pham, M., Xiong, K.: A survey on security attacks and defense techniques for connected and autonomous vehicles. Computers & Security **109**, 102269 (2021). https://doi.org/https://doi.org/10.1016/j.cose.2021.102269, https://www.sciencedirect.com/science/article/pii/S0167404821000936
45. Prest, T., Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: FALCON. Tech. rep., National Institute of Standards and Technology (2022), available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022
46. Ravi, P., Chattopadhyay, A., Baksi, A.: Side-channel and fault-injection attacks over lattice-based post-quantum schemes (kyber, dilithium): Survey and new results. Cryptology ePrint Archive, Report 2022/737 (2022), https://eprint.iacr.org/2022/737
47. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) 37th ACM STOC. pp. 84–93. ACM Press (May 2005). https://doi.org/10.1145/1060590.1060603
48. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the Association for Computing Machinery **21**(2), 120–126 (Feb 1978). https://doi.org/10.1145/359340.359342
49. Secure hash standard. National Institute of Standards and Technology, NIST FIPS PUB 180-4, U.S. Department of Commerce (Aug 2015)
50. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: 35th FOCS. pp. 124–134. IEEE Computer Society Press (Nov 1994). https://doi.org/10.1109/SFCS.1994.365700
51. Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). American National Standards Institute (ANSI), X9.62-1998 (Nov 2015)