Post-Quantum Key Exchange Mechanism for Safety Critical Systems

Tim Fritzmann[1], Jonas Vith[1], and Johanna Sepúlveda[2]

[1] Technical University of Munich, Munich, Germany[**]
[2] AIRBUS Defence and Space GmbH, Ottobrunn, Germany
{tim.fritzmann,jonas.vith}@tum.de, johanna.sepulveda@airbus.com

Abstract. The high degree of automation demanded in the automotive industry has turned vehicles into powerful mobile computing platforms. Various types of data are collected, distributed and processed. Some of the processed information is sensitive and safety critical. Thus, secure encrypted communication is a new and increasing requirement in automotive systems. The extremely long lifespan of vehicles leads to the necessity of employing cryptography that is also resistant against future attacks. The skyrocketing evolution of quantum computers in particular poses a significant threat to such a systems, thus it is also important to consider quantum attacks in automotive environments. To ensure long-term automotive security, able to resist traditional and quantum attacks, we propose the first post-quantum cryptography implementation on the Infineon AURIX microcontroller, which was specially designed to fulfill the performance and safety requirements of safety-critical applications. AURIX is currently widely deployed in the automotive sector. Our implementation shows that it is feasible to integrate post-quantum cryptography in automotive applications. Moreover, we show that optimization techniques can lead to a speed improvement of approximately 10.2 %. This can be specially important for real-time applications.

Keywords: Safety Critical · Automotive Industry · Secure Communication · Post-Quantum Cryptography · NewHope.

1 Introduction

The automotive industry has been revolutionized by an increased digitalization and new technologies, such as autonomous driving, augmented reality dashboards, and automatic emergency braking. The increasing connectivity to other vehicles (V2V), to the roadside infrastructure (V2I) or to other devices leads to many new applications. However, it also increases the attack surface and the risk for severe damage and accidents. During the last decades, the automotive industry started to secure the systems against malicious attacks. Public-Key

[**] Currently Jonas Vith is working at Rohde & Schwarz GmbH & Co. KG, Munich, Germany. The work has been developed while he was at Technical University of Munich.

Cryptography (PKC) provides the basis for establishing secured communication channels between multiple parties. It is used to support confidentiality, authenticity and non-repudiation of electronic communications and data storage.

However, current employed PKC cryptosystems such as the Rivest-Shamir-Adleman (RSA) cryptosystem, which is based on the factorization of large numbers, or Elliptic Curve Cryptography (ECC), which is based on the discrete logarithm problem, are considered insecure to attacks performed by a quantum computer. The foreseeable breakthrough of quantum computers therefore represents a risk for all communication systems. By executing Shor's [1] and Grover's [2] quantum algorithms, these computers will be able to solve the problems on which classical PKC (RSA, ECC) relies in polynomial time and decrease the security level of symmetric cryptosystems, respectively. While symmetric cryptosystems can be easily secured against quantum computers by choosing larger key sizes, securing PKC requires new algorithms that rely on different mathematical problems.

To ensure long-term communication security, quantum-resistant (also called post-quantum) cryptography must be adopted. Post-quantum cryptography relies on mathematical problems that are secure against attacks from both traditional and quantum computers. The progress in the development of quantum computers in particular poses a significant threat for applications with long lifecycles (e.g., automotive, airplanes and satellites) where deployed devices are hard to update. As a reaction, the National Institute of Standards and Technology (NIST) started the process for post-quantum standardization in 2017 [3]. The goal is to select a set of appropriate post-quantum solutions which are able to meet the security, performance, cost, and adaptability requirements of current and future applications. Post-quantum cryptography can be divided into five different classes: hash-based, multivariate-based, isogeny-based, code-based, and lattice-based. Lattice-based cryptography is due to its efficiency and relatively small public and secret key sizes one of the most promising alternatives to replace traditional algorithms.

NewHope is a lattice-based post-quantum protocol of the NIST competition which has called a lot of attention and has good chances to become standardized. This protocol, for example, has already been used in a test phase by Google and was employed in their web browser Chrome [4]. However, bringing post-quantum cryptography into different applications can be challenging. In particular, the automotive industry is known for strict performance requirements, high safety standards, and limited resources.

In order to investigate the suitability of post-quantum cryptography for automotive systems, we present the first implementation of the post-quantum protocol NewHope on an AURIX (Automotive Realtime Integrated NeXt Generation Architecture) microcontroller. The AURIX microcontroller from Infineon is used in many automotive applications such as secure on-board communication, powertrain applications (e.g. fuel injection) and safety applications (e.g. braking electronic control unit). To improve the performance of our NewHope implementation, we identify critical operations of the algorithm and apply several optimization methods. These methods include data relocation to faster memory, inline assembly, and unrolling of loops.

The rest of the paper is organized as follows. Section 2 summarizes the current state of automotive cryptography. Section 3 presents the previous work on post-quantum and in particular on lattice-based cryptography. Section 4 describes the post-quantum scheme NewHope. Section 5 discusses the AURIX platform, which was used to implement the NewHope algorithm. The results of the optimized NewHope implementation on the AURIX platform are provided in Section 6. Section 7 compares NewHope with ECC-based schemes. Finally, Section 8 concludes the article.

2 Current State - Automotive Cryptography

In the past, safety was always of high interest in the automotive industry while security considerations were nearly not present. The relevance of security has drastically grown with the Vehicle-to-Everything (V2X) connection because information has to be transported over insecure channels, which allows new attack scenarios having an impact on the behavior of the car. Automotive vehicles tend to get connected more and more in order to provide an increased comfort and better usability for the customer. This includes communication to different vehicles, provided by Vehicle-to-Vehicle (V2V) services or communication to nearby infrastructure, which is provided by Vehicle-to-Infrastructure (V2I) services. These services allow to share important information about different events which may occur on the road, such as glazed frost, traffic jams or accidents. This does not only increase the safety of all road users, but also decreases the time spent standing still, since traffic jams can be reduced. Besides these safety relevant services, Vehicle-to-Everything (V2X) services allow different types of applications, like music or video streaming, automated service control and many more, to be integrated into automotive vehicles and therefore increase the comfort for the end-user. All these services allow different attacks to be executed remotely, and therefore are in need of appropriate encryption techniques, in order to be secure. A summary of employing cryptography for V2X communication was presented in [5].

The IEEE standard 1609.2-2016 "Standard for Wireless Access in Vehicular Environments - Security Services for Applications and Management Messages" is a particularly important standard for security in the automotive industry. In order to realize secured communications, the standard recommends to use the following cryptographic components:

- digital signatures using Elliptic Curve Digital Signature Standard (ECDSA);
- asymmetric encryption using Elliptic Curve Integrated Encryption Scheme (ECIES);
- symmetric authenticated encryption using Advanced Encryption Standard in CCM mode (AES-CCM).

The first two components will be completely broken when a large-scaled quantum computer exists. So far no work exists that investigates the applicability of post-quantum cryptography for automotive systems. Therefore, in this work we present the first implementation of a post-quantum candidate on a well known automotive microcontroller in order to replace asymmetric encryption schemes using ECC. In contrast to RSA or ECC, post-quantum signature and key exchange/encryption cryptographic operations are not very similar. Thereby, an analysis of the practicability of a quantum-resistant digital signature scheme is out of the scope of this work.

3 Related Work - Post-Quantum Cryptography

An important step in the area of lattice-based cryptography was done through the introduction of the Learning With Errors (LWE) problem by Regev in 2005 [6]. It claims to be as hard as worst case lattice problems and is assumed to be secure against quantum attacks [7]. Therefore, it is a promising candidate for the replacement of traditional mathematical problems, such as integer factorization and finding discrete logarithms. The Ring-Learning With Errors (R-LWE) problem is a variant of the LWE problem and is frequently used in modern lattice-based cryptographic algorithms as it allows to reduce the key sizes and to speed up the computation [8].

A very efficient PKC-protocol based on the R-LWE problem was introduced by Alkim *et al.* [9] in 2016. Their protocol—called NewHope—uses several optimizations methods, such as the Number Theoretic Transform (NTT), to accelerate the scheme. Moreover, the authors used Advanced Vector Extensions (AVX), which are supported by Intel processors since the Sandy Bridge generation, to significantly speed up their design. In the same year, Alkim *et al.* proposed the first microcontroller implementations of NewHope on the famous ARM-Cortex M0 and M4 platforms [10]. The authors have shown in their publication that by careful optimization at assembler level it is possible to speed up critical operations.

In 2017, NewHope was submitted with small modifications described in [11] to the NIST standardization process of post-quantum cryptography. In this context, the NewHope team published an open reference implementation of their NewHope implementation [12] on which this work is based on. An ARM-Cortex M4 implementation of this NewHope implementation was presented in [13] and a RISC-V implementation based on a RI5CY-core was presented in [14]. Recently, NewHope was announced to be in the second round of the NIST competition and is thus one of the remaining 26 candidate algorithms out of 82 initial submissions [15]. Out of these 26 candidates 17—also NewHope—belong to the category of public-key encryption / key-establishment. The other 9 remaining schemes belong to the category of digital signature schemes.

4 New Hope

In this chapter, we describe the post-quantum public-key encryption / key-establishment algorithm NewHope. The NewHope cryptosystem provides two different variants: a basic version secure against Chosen-Plaintext Attacks (CPA) and an extended version against Chosen-Ciphertext Attacks (CCA). While the CPA secure version is sufficient for a standard key-establishment algorithm, public-key encryption algorithms require protection against CCA. In this article, we focus on NewHope as standard key-establishment algorithm. For a description of algorithmic changes required to offer CCA security, we refer to [12]. However, a key-establishment algorithm is usually sufficient because once a key is exchanged using PKC, symmetric cryptography can be used. This combination of PKC and symmetric cryptography (also denoted as hybrid encryption) is commonly used as symmetric cryptography is usually much faster than pure public-key encryption communication. Therefore, public-key encryption is only suitable of encrypting a small amount of data.

4.1 Notation

All mathematical operations in NewHope are performed in the ring $\mathcal{R} = \mathbb{Z}_q[x]/(x^n+1)$, where n and q are both integers. The elements of $\mathcal{R}$ can be represented as a polynomial $\boldsymbol{a} = \sum_{i=0}^{n-1} a_i x^i$ of degree $n-1$, where all coefficients a_i are reduced modulo q. Let S be a probability distribution and let $a \xleftarrow{\$} S$ denote the sampling process, i.e. random values are chosen from S and are assigned to the coefficients of $\boldsymbol{a}$. Finally, let Ψ_k be a centered binomial distribution with standard deviation $\sigma = \sqrt{k/2}$.

4.2 Protocol

$$
\begin{array}{clccl}
 & \textbf{Alice (server):} & & & \textbf{Bob (client):} \\
\textcircled{1} & \text{seed} \xleftarrow{\$} \{0,1\}^{256} & & & \\
 & \boldsymbol{a} \leftarrow \text{Gen\_a}(\text{seed}) & & & \\
\textcircled{2} & \boldsymbol{s}, \boldsymbol{e} \xleftarrow{\$} \Psi_k^n & & \textcircled{4} & \boldsymbol{s'}, \boldsymbol{e'}, \boldsymbol{e''} \xleftarrow{\$} \Psi_k^n \\
\textcircled{3} & \boldsymbol{b} \leftarrow \boldsymbol{as} + \boldsymbol{e} & \xrightarrow{\boldsymbol{b},\ \text{seed}} & \textcircled{5} & \boldsymbol{a} \leftarrow \text{Gen\_a}(\text{seed}) \\
 & & & \textcircled{6} & m \xleftarrow{\$} \{0,1\}^{256} \\
 & & & & \boldsymbol{d} \leftarrow \text{Encode}(m) \\
 & & & & \boldsymbol{u} \leftarrow \boldsymbol{as'} + \boldsymbol{e'} \\
\textcircled{8} & \boldsymbol{d'} \leftarrow \boldsymbol{c} - \boldsymbol{us} & \xleftarrow{\boldsymbol{u},\ \boldsymbol{c}} & & \boldsymbol{c} \leftarrow \boldsymbol{bs'} + \boldsymbol{e''} + \boldsymbol{d} \\
 & m' \leftarrow \text{Decode}(\boldsymbol{d'}) & & \textcircled{7} & \mu \leftarrow \text{SHA3}(m) \\
\textcircled{9} & \mu \leftarrow \text{SHA3}(m') & & &
\end{array}
$$

Protocol 1: NewHope protocol. All polynomials in $\mathcal{R}$ are printed in bold and all 256-bit vectors in normal font.

Protocol 1 provides an abstract overview of the NewHope scheme. Note that not all details of the algorithm are shown in order to reduce the complexity. The goal of the algorithm is that two parties, Alice and Bob, agree on a secret key μ, which then can be used for a fast symmetric encryption scheme. The protocol can be divided into three main subroutines: *Key Generation*, *Encapsulation/Encryption*, and *Decapsulation/Decryption*.

Key Generation (Step 1 – Step 3). The goal of the key generation is to generate the secret key $sk = \boldsymbol{s}$ and public key $pk = (\boldsymbol{b}, \text{seed})$.

- In Step 1, Alice samples the 256-bit seed from a True Random Number Generator (TRNG). This seed is then forwarded to the function *Gen_a* where the seed is expanded with a Pseudo Random Number Generator (PRNG) in order to generate the public polynomial $\boldsymbol{a}$. As PRNG the extendable output function SHAKE is used. The coefficients of $\boldsymbol{a}$ are uniformly distributed between 0 and $q-1$.
- In Step 2, Alice creates the secret polynomial $\boldsymbol{s}$ and error polynomial $\boldsymbol{e}$, where all n coefficients of the polynomials are sampled from the binomial distribution Ψ_k. In comparison to the coefficients of $\boldsymbol{a}$, the binomially distributed coefficients of $\boldsymbol{s}$ and $\boldsymbol{e}$ have a small amplitude. The polynomial $\boldsymbol{s}$ is the secret key for Alice.
- In Step 3, Alice creates the R-LWE instance $\boldsymbol{b}$ by multiplying the public polynomial $\boldsymbol{a}$ with the secret $\boldsymbol{s}$ and adding the error polynomial $\boldsymbol{e}$ to this product. This R-LWE instance hides the secret $\boldsymbol{s}$ such that $\boldsymbol{b}$ can be sent securely (together with the public seed) over the public insecure channel to Bob. The security of this R-LWE instance rises with increasing polynomial degree, an decreasing modulo reduction parameter q or an increasing variance of the binomially sampled coefficients of the error polynomial $\boldsymbol{e}$. When the parameters are appropriately set, it is infeasible for an attacker to recover the secret or to distinguish $\boldsymbol{b}$ from uniform noise—even with the help of a quantum computer.

Encapsulation/Encryption (Step 4 – Step 7). The goal of the Encapsulation/Encryption is to encapsulate/encrypt the key/message m and to generate the ciphertext $ci = (\boldsymbol{u}, \boldsymbol{c})$.

- In Step 4, Bob generates similar to Step 2 a secret key polynomial $\boldsymbol{s'}$, and the error polynomials $\boldsymbol{e'}$ and $\boldsymbol{e''}$.
- In Step 5, Bob uses the same seed as Alice to generate the public polynomial $\boldsymbol{a}$.
- In Step 6, the key m, which is a 256-bit vector, is created. In a public-key encryption setting, the message m would be an input. The key is then encoded into the polynomial $\boldsymbol{d}$. During this step, some redundancy is included in order to allow an error correction during the decoding. Now, Bob creates the two R-LWE instances $\boldsymbol{u}$ and $\boldsymbol{c}$ to hide the secret polynomial $\boldsymbol{s'}$ and key polynomial $\boldsymbol{d}$. Both R-LWE instances can be sent securely to Alice.

– In Step 7, Bob can take the message m as input to the cryptographic hash function SHA3. The output of this function is the shared key for Bob.

Decapsulation/Decryption (Step 8 – Step 9). The goal of the Decapsulation/Decryption is to decapsulate/decrypt the ciphertext $ci = (\boldsymbol{u}, \boldsymbol{c})$ using the secret key sk in order to retrieve m' and agree on a shared secret μ.

– In Step 8, Alice obtains $\boldsymbol{d'}$ by subtracting $\boldsymbol{us}$ from $\boldsymbol{c}$. This step eliminates the largest noise term $\boldsymbol{ass'}$ in $\boldsymbol{c} - \boldsymbol{us} = \boldsymbol{bs'} + \boldsymbol{e''} + \text{Encode}(m) - (\boldsymbol{as'} + \boldsymbol{e'})\boldsymbol{s} = \boldsymbol{ass'} + \boldsymbol{es'} + \boldsymbol{e''} + \text{Encode}(m) - \boldsymbol{ass'} - \boldsymbol{e's}$ and only a relative small difference noise term remains on the message. The decoding step now removes with a high probability the remaining noise.
– In Step 9, Alice can finally use her message m' to agree on the same secret μ as Bob.

5 AURIX Microcontroller

The AURIX microcontroller from Infineon is a widely used *Harvard*-based microcontroller family used for (safety-critical) automotive applications, such as braking or steering systems, automotive power-train controllers or networking and cellular communications. It is used in applications, where a correct operation must be guaranteed, also in difficult environments with high temperature ranges, steep temperature gradients or other harsh environmental impacts. The AURIX is built upon the *TriCore* architecture from Infineon, which merges the real-time capabilities of microcontrollers with the computational power of a Digital Signal Processor (DSP) and the high performance of a Reduced Instruction Set Computer (RISC) in one programmable core [16].

5.1 Aurix Architecture

An abstract view of the AURIX architecture is illustrated in Fig. 1. The AURIX architecture features three cores running at a maximum frequency of 300 MHz. Directly at each of the cores the AURIX has fast RAM blocks containing a Program Scratchpad RAM (PSPR) and a Data Scratchpad RAM (DSPR). These two memory types provide a fast and deterministic program fetch as well as data access. This is particularly important for performance critical code and also an advantage in cryptographic applications like NewHope since applicable cache attacks, which may get information from a data dependent runtime caused by caches, can be ruled out by construction. Note that cryptographic operations, e.g. lookup tables, should never be loaded partially to rule out cache attacks.

The Cross Bar Interconnect (SRI) implements a direct connection to further memory blocks and a connection between all connected masters and slaves, allowing non-blocking parallel and thus efficient communication of different masters with different slaves/masters. This interconnection runs at the CPU-clock frequency and allows thus a high data throughput.

The System Peripheral Bus (SPB) can be used to connect the core with SPI, FlexRay and other internal or external peripherals.

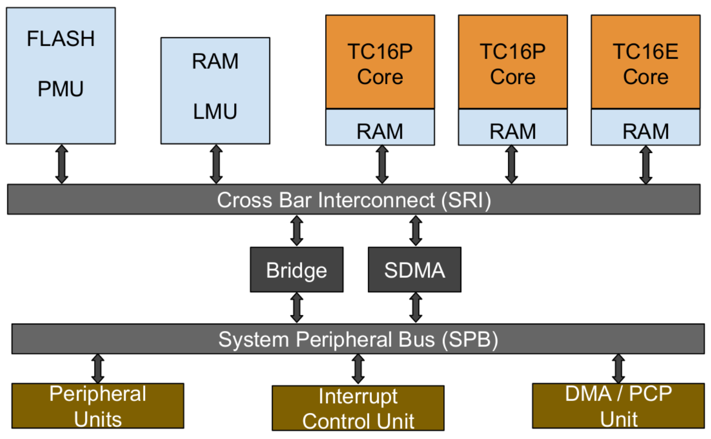

Fig. 1: AURIX architecture with up to three cores and fast local random access memories.

5.2 Aurix TriCore

The core has two main pipelines with four stages each, and a smaller pipeline for the loop control. As all three pipelines can be used in parallel, up to three instructions per clock cycle are theoretically possible [16]. The core of the AURIX consists of four main modules: the *Integer Processing Unit*, a *Load/Store Unit*, a *Fetch Unit*, and several system registers *CSFRs*.

The *Integer Processing Unit* consists of a Multiply-Accumulate Unit (MAC) and an Arithmetic Logic Unit (ALU). The MAC unit is especially useful to speed up the execution, e.g., of signal processing applications, which heavily rely on MACs. It is capable of calculating two 16×16 MACs or one 32×32 MAC per cycle. Furthermore, the *TriCore* architecture features *packed arithmetic*, which allows to partition 32-bit words into sub-words, which can then be loaded, used and stored in parallel, increasing the performance in certain applications [17]. See Figure 2 for an illustration of the concept of packed arithmetic.

The *Load/Store Unit* operates in parallel to the *Integer Execute Unit* [16]. This allows to achieve lower processing latencies since instructions can be executed in parallel and cycles spent waiting for data from the memory can be reduced.

One further benefit of the AURIX compared to different microcontrollers is that on the AURIX it is possible to have both 16 and 32 bit opcode instructions. This allows to decrease the code size of a program by approximately 30 to 40% [16].

5.3 Code Optimization Methods

Since the AURIX is a superscalar processor, it can achieve a lower CPI (average number of cycles needed per instruction) than singlescalar microcontrollers, like

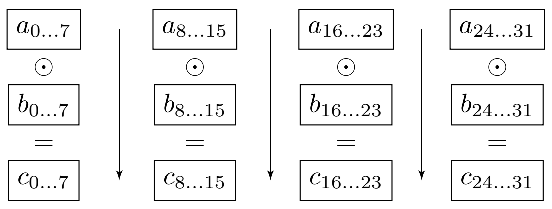

Fig. 2: Packed arithmetic allows to operate multiple sub-words in parallel. Either four 8-bit data packets are operated in parallel, or two 16-bit packets are operated in parallel. $a_{0\ldots7} \odot b_{0\ldots7} = c_{0\ldots7}$, where $\odot$ is an arithmetic operation with packed arithmetic support (e.g., addition or subtraction).

the ARM-Cortex M4. The best CPI is obtained in a loop where load/store instructions and integer instructions are available such that each pipeline is filled most of the time. This is best reached by a typical signal processing task. Nevertheless, in non-signal processing tasks, a decrease of the CPI can still be measured since the compiler has more degrees of freedom in interleaving instructions and therefore can reduce wait states. To speed up the performance, we use the optimization methods: data relocation, code profiling, inline assembly, unrolling of loops, and the usage of faster arithmetic.

Data Relocation. On an AURIX microcontroller when executing instructions which work on variables located in a memory instead of registers different latencies can be measured when different memory regions are involved. This is depicted in Table 1, where the amount of memory stalls is shown that are added when reading from or writing to different memories.

Table 1: Memory stalls induced by accessing different memory regions [18]. When working on CORE0, frequently accessed variables should be placed in the DSPR_0, since it provides the lowest access latency.

CORE0	DSPR_0	PSPR_0	$\text{PSPR}_{1,2}$
read	0	8	8
write	0	0	0
instr. fetch	N/A	0	8

Table 1 clearly shows that the efficiency of a program running on an AURIX microcontroller heavily depends on the location of individual data elements. Frequently accessed variables and arrays which may have been declared as `const` are placed in the PSPR. Since variables in the PSPR cannot be access as quickly as in the DSPR, it is desirable to relocate them into the faster DSPR. An increased efficiency can be hereby easily obtained by providing pragmas to the preprocessor. This is achieved by inserting the `#pragma section` command with the right section (e.g., `".data"`) before and after the variable or array that should be relocated. This ensures that the preprocessor knows in which part of the memory the variable should be placed.

Code profiling. For an efficient data relocation and improvement of the code, a profiler like the *TASKING Embedded Profiler* is helpful. It allows to spot possible bottlenecks of an implementation by showing different performance measures. Besides identifying hotspots, the profiler also shows the number of stalls occurring at individual lines of assembly code, and thus is a good tool to spot falsely placed variables.

Further optimization techniques. Independent to the platform, further optimization techniques like inline assembly, unrolling of loops or the usage of faster arithmetic operations, like shifting instead of multiplication and division by 2, can be applied in order to further increase the performance of any program. This, of course is an exhaustive and iterative process, especially when having the compiler optimization flags (e.g., `-O2` or `-O3`) enabled, where the compiler already maximizes the performance of the program in trade to an increased code footprint. Still, even with optimization flags enabled, it is possible to increase the performance of certain code sections.

6 NewHope on AURIX

As a starting point for the AURIX TC297x implementation, we used the optimized reference implementation of the NewHope version submitted to NIST[3]. More specifically, we selected NewHope in CPA-mode with the strongest security level (NewHope1024-CPA).

Table 2 shows a comparison of the cycle count of the optimized NewHope1024-CPA reference implementation on the AURIX microcontroller and the same version running on the Cortex M4. It can be clearly seen, that the AURIX outperforms the Cortex M4. This can be explained by the superscalar architecture of the AURIX which can achieve a lower CPI since it can issue multiple instructions simultaneously, compared to a singlescalar architecture.

6.1 Optimized AURIX Implementation

In order to efficiently optimize the NewHope AURIX implementation, we created a hotspot chart using the *TASKING Embedded Profiler*. Figure 3 shows the hotspot graph for NewHope1024 on the AURIX. The graph clearly indicates which functions consume most of the cycles, and thus should be the starting point for any optimizations. The results show that in the NewHope algorithm most of the cycles are spent in the *Keccak State Permutation* function. The state

[3] The optimized reference implementation of NewHope can be found at NIST's official PQ-Cryptography site: `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions`

Table 2: Cycle count of NewHope1024 in CPA-mode as described in subsection 4.2. The cycles of the Cortex M4 have been taken from [13,12]. The AURIX TC297x clearly outperforms the Cortex M4 due to its superscalar architecture which allows the execution of multiple instructions in parallel.

Implementation	Key Generation	Encapsulation/ Encryption	Decapsulation/ Decryption	Total
Cortex M4 [13,12]	2669559	3910871	163887	6744317
AURIX ref.	1249939	1819215	326294	3395448

permutation is the heart of the SHAKE and SHA3 functions and is in particular heavily used in Step 1, Step 2, and Step 4 of Protocol 1 in order to create the random polynomials $\boldsymbol{a}$, $\boldsymbol{s}$, $\boldsymbol{s'}$, $\boldsymbol{e}$, $\boldsymbol{e'}$, and $\boldsymbol{e''}$.

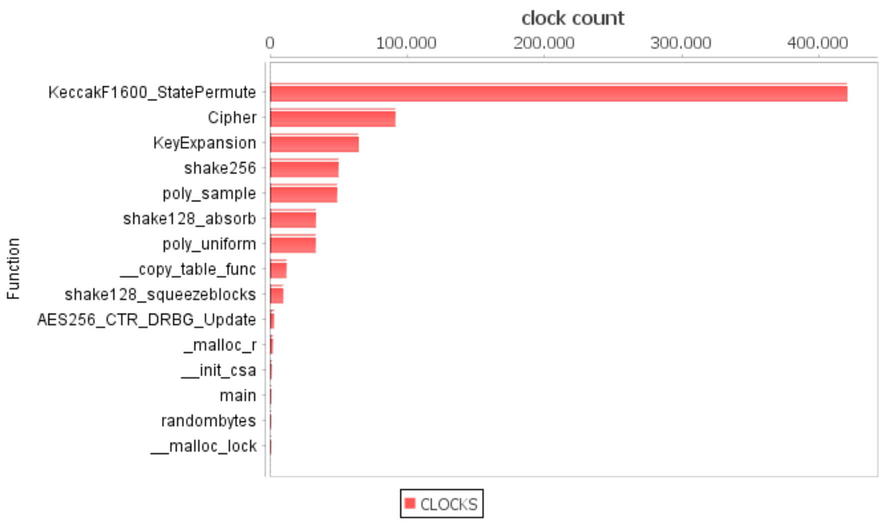

Fig. 3: Hotspot chart of NewHope1024 in CPA-mode using the TASKING Embedded Profiler.

By relocating frequently accessed data elements like the round constants of the *Keccak* implementation, it is possible to achieve a decreased runtime of *NewHope*. Further, by iteratively swapping arithmetic operations with faster arithmetic operations (e.g., `<<` / `>>` vs. $\times$ / $\div$ by powers of 2, or `&(m-1)` vs. %`m` for faster modulo operation, where `m` is a power of 2), the performance of *NewHope* can be increased. Table 3 shows the speedup obtained by manually fine tuning certain code sections and relocating often accessed arrays from the PSPR to the DSPR. All in all, an improvement of approximately 10.2% has been achieved by fine tuning the NewHope implementation on the AURIX TC29x. Compared to the cycles given for the Cortex M4, this accounts to an overall improvement of approximately 219%.

Table 3: Cycle count of the manually fine tuned NewHope1024 in CPA-mode.

Implementation	Key Generation	Encapsulation/ Encryption	Decapsulation/ Decryption	Total
AURIX ref.	1249939	1819215	326294	3395448
AURIX opt.	1126265	1641684	315780	3083720

7 Comparison to Elliptic Curve Cryptography

Most of the currently employed cryptographic applications are based on ECC. It is much faster than RSA and has extremely low key sizes. However, the Elliptic Curve Diffie Hellman (ECDH) will be broken when powerful quantum computers exist. Table 4 presents the security level and the minimum amount of exchanged data from Alice to Bob and vise versa for a single key-agreement. The table shows that NewHope has a very high level of security against both classical and quantum attacks. However, the amount of data that has to be exchanged is much higher in comparison to modern ECC-based schemes. This is due to larger public keys and ciphertexts. In return for the higher amount of exchanged data, NewHope appears to be faster than comparable ECC schemes. For example, the authors in [19] reported that NewHope is faster than the famous NIST curve *NIST P-256*. The authors in [10] came also to the conclusion that NewHope has a higher performance. They implemented NewHope on the microcontrollers Cortex M0 and M4. Their NewHope Cortex M0 implementation outperforms the results of a Curve25519 implementation by more than a factor of two. Unfortunately, no ECC implementation results were found for the AURIX such that a comparison with this platform is not possible.

Table 4: Comparison of the post-quantum protocol NewHope1024 in CPA-mode with the widely used Eliptic Curve Diffie Hellman (ECDH). The table shows the required amount of exchanged data and the classical as well as the post-quantum security level.

Cryptographic Function	Data Alice-Bob	Data Bob-Alice	Security Level	PQ Security Level
NewHope1024	1824 Bytes	2176 Bytes	257-bit	233-bit
ECDH	32 Bytes	32 Bytes	128-bit	-

8 Conclusion

The demand for secure communication in the automotive industry has rapidly grown in the past years. In particular, the Vehicle-to-Everything (V2X) communication turns the vehicle into a fully networked system. The vehicular communication system can be protected using cryptographic components. PKC is used for digital signatures and key-establishment or encryption of small data. Once a key has been established, symmetric cryptography can be used for fast encryption of large data. Currently, ECC is used to realize asymmetric cryptography and is also recommended in the standard IEEE 1609.2-2016. However, as vehicles have an extremely long lifespan also future attacks performed on quantum computers should be considered. In this article, we analyzed the suitability of a quantum-resistant key-establishment / public-key encryption scheme for the automotive industry by implementing the protocol on the AURIX microcontroller, which fulfills the tough requirements of the automotive industry. Moreover, several optimization methods were presented to decrease the execution time of the algorithm. The analysis shows that the implemented algorithm, NewHope, is a strong alternative for ECC-based schemes. It has a high level of security and can be implemented such that it is faster than ECC schemes. The only drawback in comparison to ECC are larger key and ciphertext sizes. However, this first study shows that post-quantum cryptography can be very competitive and considered for the automotive industry. Future work requires an analysis of quantum-resistant digital signature schemes for automotive applications.

Acknowledgments. The authors want to thank the support of HighTec, who provides a set of Linux licenses for the development toolchain used in this work. Also, we want to thank Infineon Technologies AG, who provides the hardware used in this project. This work was partly funded by the German Ministry of Education, Research and Technology in the context of the project Sec4CARs (grant number 16KIS0795).

References

1. P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on.* Ieee, 1994, pp. 124–134.
2. G. L.K., "A fast quantum mechanical algorithm for database search," in *28th Annual ACM Symposium on the Theory of Computing*, May 1996, p. 212.
3. National Institute of Standards and Technology, "Announcing request for nominations for public-key post-quantum cryptographic algorithms," 2016, https://csrc.nist.gov/news/2016/public-key-post-quantum-cryptographic-algorithms.
4. M. Braithwaite. (2016) Experimenting with post-quantum cryptography. https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html, visited 2017-09-10. [Online]. Available: https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html

5. T. Schütze, “Automotive security: Cryptography for car2x communication,” in *Embedded World Conference*, vol. 3. Citeseer, 2011, pp. 4–24.
6. O. Regev, “On lattices, learning with errors, random linear codes, and cryptography,” *Journal of the ACM (JACM)*, vol. 56, no. 6, p. 34, 2009.
7. ——, “The learning with errors problem,” *Invited survey in CCC*, vol. 7, 2010.
8. V. Lyubashevsky, C. Peikert, and O. Regev, “On ideal lattices and learning with errors over rings,” in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2010, pp. 1–23.
9. E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, “Post-quantum key exchange - a new hope,” in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 327–343.
10. E. Alkim, P. Jakubeit, and P. Schwabe, “Newhope on arm cortex-m,” in *International Conference on Security, Privacy, and Applied Cryptography Engineering*. Springer, 2016, pp. 332–349.
11. E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, “Newhope without reconciliation.” *IACR Cryptology ePrint Archive*, vol. 2016, p. 1157, 2016.
12. E. Alkim, R. Avanzi, J. Bos, L. Ducas, A. de la Piedra, T. Pöppelmann, P. Schwabe, and D. Stebila, “NewHope: Algorithm Specifcations and Supporting Documentation,” 2017, https://newhopecrypto.org/data/NewHope_2017_12_21.pdf.
13. T. Oder, T. Schneider, T. Pöppelmann, and T. Güneysu, “Practical cca2-secure and masked ring-lwe implementation,” *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 142–174, 2018.
14. 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)*. IEEE, 2019, pp. 1148–1153.
15. G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, Y.-K. Liu, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, and D. S.-T. and, “Status report on the first round of the nist post-quantum cryptography standardization process,” 2019, https://nvlpubs.nist.gov/nistpubs/ir/2019/NIST.IR.8240.pdf.
16. “TriCore^{TM} 1.3 32-bit Unified Processor Core,” https://www.infineon.com/dgdl/TC1_3_ArchOverview_1.pdf?fileId=db3a304312bae05f0112be86204c0111, accessed: 2018-12-13.
17. “TriCore 32-bit Unified Processor DSP Optimization Guide Part 1: Instruction Set, v1.6.4,” 2003-01.
18. “TASKING VX-toolset for TriCore User Guide, v6.2r2,” 2018-03.
19. V. Valyukh, “Performance and comparison of postquantum cryptographic algorithms,” 2017.