

# Requirements for Post-Quantum Cryptography on Embedded Devices in the IoT

Derek Atkins  
CTO, Veridify Security

## Abstract

As NIST proceeds through Round 3 to whittle down and choose the PQC methods it will standardize, members of the commercial community who plan to use the output of NIST would like to ensure their needs are also being met. Specifically, while the needs of the US Federal Government are absolutely vital, and large organizations like Google, Amazon, Microsoft, Apple, etc. would be able to directly leverage anything NIST creates for the Federal Government, there are other use-cases that are just as vital but have a significantly different set of operating requirements and restrictions, namely restricted resources for embedded “Internet of Things” (IoT) devices. This paper provides the background of various IoT platforms in common use, their available resources, and summarizes with the point that NIST should include at least one method that will fit and successfully operate in these restricted environments, along with a suggestion for how NIST might achieve that by prioritizing RAM usage over ROM usage while also prioritizing low clock-cycle (faster) methods.

## Introduction

The Internet of Things (IoT) is the label given to the billions of connected devices leveraging small, embedded controllers that run devices that monitor and control our every-growing connected world. Many of these IoT devices, specifically in certain industries like utilities, medical devices, public infrastructure, building automation, automotive, and even industrial applications are built with devices that can live in deployment for 10, 20, or even 40-50 years. Obviously as these long-lived devices get smarter and start communicating, the cryptography used to protect those devices needs to be secure long into the future, a future where quantum computers may arrive and break existing, legacy, standard cryptography. This is why the IoT requires standardized PQC.

The IoT uses small, resource-constrained, embedded processors in order to limit power, size, and cost of the billions (and growing) devices. These are not 64-bit, 2-3GHz Intel Haswell devices. These are not devices with 512-bit-wide vector SIMD units. Indeed, these devices are not necessarily even an ARM Cortex M4! Rather, they are small (32-, 16-, and sometimes even 8-bit MCUs) with small amounts of RAM, limited flash/storage, and clock speeds measured in Megahertz, not Gigahertz. These are some of the devices that are controlling our vital infrastructure and must remain secure, but still require cryptography that does not take seconds, let alone minutes, to execute.

This paper is a short overview of the current state of portions of the IoT, the state of the controllers being used, their resources, and a detailed suggestion to NIST on how to consider all IoT devices, including low-resource devices, when progressing from Round 3.

## An analysis of low-end embedded MCUs

While many large corporations are focused on large-scale computations leveraging large CPUs [CPUs], the IoT continually looks for small, inexpensive, low-resource sources [Trends]. While the number of ARM and RISC-V 32-bit processors is increasing, there are still a growing number of devices that leverage alternative cores, 16-bit processors like the MSP-430 or even 8-bit MCUs like the 8051 and AVR-8. Even in the 32-bit range, the number of ARM Cortex M3 or even M0 processors is still

growing, and the need to secure these devices does not disappear just because they are smaller. Indeed, it is the use-cases of these devices that drive the requirement to secure them, because they are being used in critical infrastructure.

The major restrictions on these devices are three-fold: amount of RAM available, amount of ROM/Flash/Storage available, and clock speed (execution performance).

Tackling these topics in reverse order: The clock speed of these devices sits in the Megahertz range, often as low as 8-24MHz, although sometimes rising into the 100-300MHz range (see [Trends]). Obviously, the larger devices will have faster clocks, but it's not uncommon for these small devices to sit with clock speeds in the low double-digits. When considering performance requirements, this implies a desire to keep signature verification down to a few million cycles and, similarly, KEM computations in a similar level.

Along with performance issues would be the available multiplier hardware and (lack of) floating point implementations. One significant issue with these low-end devices is that many may not contain a  $32 \times 32 \rightarrow 64$  multiplier. Sometimes it has a  $32 \times 32 \rightarrow 32$  (high/low) like you find in the RISC-V RV32 ISA. Sometimes is only has a  $16 \times 16$  like on the MSP-430! These sub-optimal multiplier implementations imply performance reductions, which must be considered.

The saving grace for performance is that for most IoT use-cases, the public-key operations can be amortized over long-lived sessions, so one does not require real-time public-key performance for every message transmitted or received. The downside is that session setup may take longer. The key factor is that this setup not take "too long." Alas, this will necessarily be a subjective observation based on the actual use-case. The real factor is to keep the cycle count as low as possible as is a requirement of most IoT implementations.

The next major obstacle to IoT deployment is code storage (ROM/Flash) size. The good news is that storage is getting larger and less expensive [Flash], so while some devices may be limited to only 16-32KB of flash, (e.g. the [XMC1100] with ARM Cortex M0 contains 64K Flash and 16K RAM, the [RSL10] contains a Cortex-M3 @ 48MHz with 384KB Flash and 32K RAM) it's not uncommon to see much larger sizes, even 1-2MB of storage! On the other hand, the 8-bit [PIC] only contains 56K Flash and 4K RAM.

The biggest hurdle, however, is RAM. RAM sizes are definitely small, and not increasing at all at the same pace as ROM/Flash storage. It's not uncommon for devices to only have 8-16KB of RAM, with the larger sizes being limited to 64KB. For example, the [RL78], with a maximum 32MHz clock with 16-bit processor, has various packages with anywhere from 1-512KB Flash but only 0.5-32K of RAM.

### **Suggested Requirements to NIST PQC**

The PQM4 project [PQM4] contains performance metrics [Metrics] for many of the Round 3 methods on an ARM Cortex M4. These metrics include code size (ROM), RAM usage, and performance cycle counts (among other data). We propose that these metrics be included in decision-making for NIST's PQC project.

While the PQM4 is focused on the M4 processor (as is NIST), we propose an assumption about porting to smaller devices, and encourage NIST to consider devices smaller than the M4 when evaluating PQC methods. Specifically, we propose an assumption that the code and RAM usage will not materially

increase when porting a method from an M4 to M3 or M0. By making this assumption, we can use the RAM and ROM limits of smaller devices as a key to determine if a proposed PQC method will fit.

When comparing real-world IoT device limitations to the PQM4 performance matrix and assuming that code size and RAM usage will remain at a similar level when ported down to an M3 or M0, it is quite clear that the RAM limitations become a more decisive factor.

For example, comparing Falcon and Dilithium signature scheme signature validation, we notice that Dilithium code size is 12-20KB, but RAM usage is 40-70KB, whereas Falcon code is 160KB but only requires 500B of RAM (with a tradeoff of 80KB code size for 4-8K RAM). Based on realistic low-end resource availability, for all practical purposes Dilithium just does not fit due to its RAM consumption. Even if Dilithium were 100 times better than Falcon, these devices do not contain enough RAM to run it even if they wanted to!

Similar analyses should be made for the KEMs. The good news is that Kyber and Saber both have good resource usage both in terms of code size and RAM consumption, and provide some RAM/ROM implementation tradeoffs. While we're still sitting in the code-size of 5-10KB with 3-24KB RAM, these fit on most smaller IoT devices.

## **Conclusions**

We encourage NIST to consider devices smaller than the ARM Cortex M4 when finalizing its PQC candidates. Specifically, they should consider the resource constraints of a Cortex M0, if not even smaller processors.

Based on current IoT device resources, namely Flash growing faster than RAM, we propose that NIST focus on smaller RAM usage vs smaller code-size.

Using these metrics, we would propose that Falcon be chosen over Dilithium for signatures, and that Saber or Kyber be selected as a KEM, with Kyber taking precedence due to its lower resource requirements.

## **References**

[CPUs] <https://github.com/karlrupp/microprocessor-trend-data>

[Flash] <https://www.microcontrollertips.com/embedded-use-of-nand-and-nor-flash-memory-is-evolving/>

[Metrics] <https://github.com/mupq/pqm4/blob/master/benchmarks.md>

[PIC] <http://ww1.microchip.com/downloads/en/DeviceDoc/30010107A.pdf>

[PQM4] <https://github.com/mupq/pqm4>

[RL78] <https://www.renesas.com/us/en/products/microcontrollers-microprocessors/rl78-low-power-8-16-bit-mcus/rl78-introductory-guide>

[RSL10] <https://www.onsemi.com/products/connectivity/wireless-rf-transceivers/rsl10>

[Trends] <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.630.9281&rep=rep1&type=pdf>

[XMC1100] <https://www.infineon.com/cms/en/product/microcontroller/32-bit-industrial-microcontroller-based-on-arm-cortex-m/32-bit-xmc1000-industrial-microcontroller-arm-cortex-m0/xmc1100/>