# Performance comparisons of post-quantum cryptography algorithms on low-power devices

Josef Olsson Isak Stenström
*Email: {josol381,isast379}@student.liu.se*
Supervisor: Niklas Carlsson, {niklas.carlsson@liu.se}
Project Report for Information Security Course
*Linköping University, Sweden*

## Abstract

*New cryptographic algorithms are being developed that are resistant to quantum computers. For the new algorithms to be useable they need to have sufficient performance, even on low-power devices.*

*This report discusses three different algorithms that are part of the standardization effort. It performs performance tests on the algorithms and compares their performance and resource usage on different platforms.*

## 1. Introduction

Quantum computers are posing a problem to current cryptographic algorithms which is why research is being conducted to create new algorithms that will not be affected by quantum computers. For an algorithm to be a candidate for wide use it needs, amongst other criteria, to have satisfactory performance, even on low-power systems such as IoT-devices.

This report evaluates a sample of new, potentially quantum safe cryptographic algorithms (commonly referred to as Post-Quantum Cryptography, PQC) to analyze their suitability for use on low-power devices.

This report aims to answer the questions:

- How does the relative performance between the algorithms differ between low- and high-power devices?
- What is the resource usage for the algorithms and how do they compare for low-power devices?

The tested algorithms are:

- Falcon-512
- Rainbow-$I_a$
- RedGeMSS128

These are evaluated using a Raspberry Pi and a x86 desktop computer.

## 2. Background

Most of the commonly used cryptographic algorithms are broken by Shor's algorithm [1] and the use of quantum computers. Fortunately, quantum computers do not yet have the capacity to break algorithms with large keys.

New algorithms are being standardized that are resistant to quantum computers [2]. These are called post-quantum algorithms and build on problems that quantum computers cannot solve in polynomial time.

The work in this report is based on the report "Retrofitting Post-Quantum Cryptography in Internet Protocols: A Case Study of DNSSEC" [3]. It explores the possibility of using the new algorithms that are being proposed for standardization in DNSSEC. It finds that there are three algorithms that are suitable for use in DNSSEC. Part of their research was developing test scripts that compared the performance of the algorithms. These scripts are open source and were used to generate the data for this report.

Recently, the security of Rainbow-$I_a$ has been called into question [4]. A regular laptop can break the encryption in a relatively short time. This is indeed quite bad but another version of the algorithm or another algorithm based on the same scheme could perhaps be developed that is not prone to this attack. Therefore, this report will include Rainbow-$I_a$ in the study since it could still yield valuable insights for the standardization process.

## 3. Methods

The algorithms were evaluated on two different systems, one being a Raspberry Pi Model B+ rev 1.2, which has a BCM2835 ARM processor and runs Raspbian 11, and the other a desktop computer with an Intel i7 4770k x86 processor which runs Ubuntu 21.10.

Before evaluating the algorithms, they need to be able to run on the different platforms. This involves installing the algorithms and all their dependencies. The test scripts come with installation instructions, and since they were developed for x86 hardware it should then be trivial to get the algorithms working on it. Getting the algorithms working on the Raspberry Pi could be more challenging.

We gathered four different measurements:

- Number of signs per second
- Number of verifies per second
- Memory usage
- Power consumption

The first two measure the number of operations that each algorithm can perform on the given hardware. This data comes directly from the test scripts provided in the paper [3].

The memory usage was measured using the GNU Time command [5]. This measured the maximum resident set size of the test script.

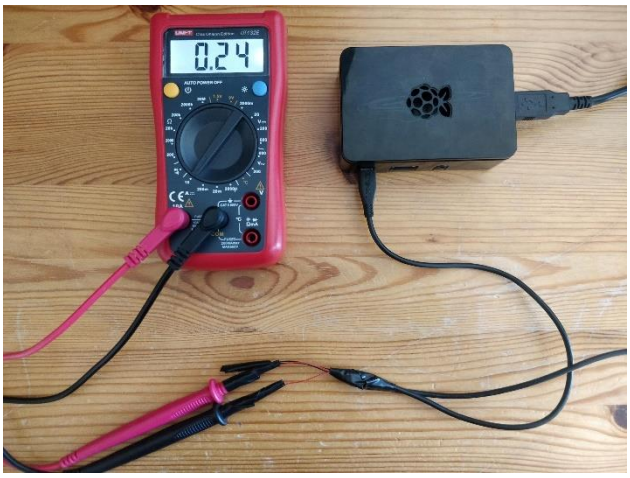


**Figure 1: Setup for measuring the current**

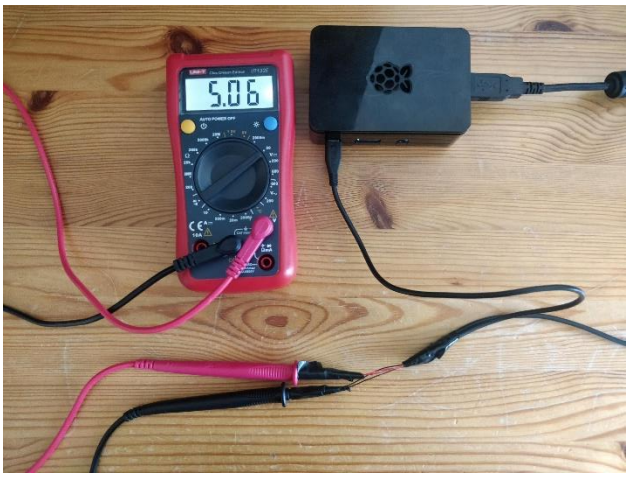


**Figure 2: Setup for measuring the voltage**

The power consumption was only calculated for the Raspberry Pi, not the desktop computer due to lacking suitable equipment for such a task. It was calculated from two measurements, current and voltage and they were measured using a multimeter. Both were measured by cutting the power cable and measuring from the conductors within. The current was measured but placing the multimeter in series with the power lead, and the voltage by measuring between the positive and ground. During the consumption tests, only the power cable and a keyboard was connected to the device. The setup is shown in the two figures above.

# 4. Result

This section presents the data gathered by the method described in the previous section.

## 4.1 Installation of algorithms

Getting all algorithms to run on the PC was difficult but possible. Both Falcon-512 and Rainbow-$I_a$ worked with the instruction from their test scripts. RedGeMSS128 depended on the library XKCP [6], including a function in the library that was no longer a part of the library, it was removed or renamed in a reorganization in a prior version. By using a two year old version of XKCP the algorithm ran without problems.

The same steps worked on the Raspberry Pi with the exception that the old version of XKCP did not support ARM processors. ARM support was added in later versions of XKCP, but there were no versions that both supported the Raspberry Pi and had the function needed for the algorithm. There was an attempt to modify the algorithm to use the new version of the library but it proved unsuccessful.

Since RedGeMSS128 does not run on both devices it was not a part of the measurements.

## 4.2 Performance measurements

The performance measurements were repeated ten times and the data is shown in Appendix 1: Measurement data. The data was compiled into Table 1 which shows the mean of the measurements, the 95% confidence interval, and the coefficient of variation, for each algorithm and device. The current and the voltage have no confidence interval since the available precision of the measurements was insufficient to capture any differences, and therefore those measurements were only performed for three tests in each series.

Table 2 shows the relative performance of the PC compared to the Raspberry Pi for the different operations and algorithms. This means that the PC performed 34.6 times more signs per second than the Raspberry Pi with the Falcon-512 algorithm.

**Table 1: Measurement averages**

| | Raspberry Pi | | | PC | |
|---|---|---|---|---|---|
| | Idle | Falcon-512 | Rainbow-$I_a$ | Falcon-512 | Rainbow-$I_a$ |
| Current (mA) | 195 | 225 | 225 | - | - |
| Voltage (V) | 5.06 | 5.06 | 5.06 | - | - |
| Mean signs per second | - | 126.6 | 280.3 | 4372.9 | 7058.1 |
| 95% confidence interval | - | 126 - 127 | 264 - 296 | 4335 - 4411 | 7007 - 7110 |
| Coefficient of variation | - | 0.7% | 9.3% | 1.4% | 1.2% |
| Mean verifies per second | - | 931.1 | 317.6 | 26233.7 | 7688.7 |
| 95% confidence interval | - | 930 - 933 | 317 - 318 | 26169 - 26299 | 7640 - 7737 |
| Coefficient of variation | - | 0.3% | 0.2% | 0.4% | 1.0% |
| Mean memory usage (KiB) | - | 2688.8 | 2526.4 | 3667.2 | 3342.8 |
| 95% confidence interval | - | 2652 - 2726 | 2496 - 2557 | 3638 - 3697 | 3284 - 3402 |
| Coefficient of variation | - | 2.2% | 2.0% | 1.3% | 2.8% |

**Table 2: Performance multiples of the measurement averages for the PC compared to the Raspberry Pi**

| | Falcon-512 | Rainbow-$I_a$ |
|---|---|---|
| Signing multiple | 34.6 | 25.2 |
| Verify multiple | 28.2 | 24.2 |
| Memory usage multiple | 1.4 | 1.3 |

## 4.3 Observations

For all data series, the coefficient of variation is relatively small. This shows that the raw data values vary minimally and thus the confidence interval is also quite narrow. The exception is signing with Rainbow-$I_a$ on the Raspberry Pi, which has a much higher coefficient of variation than the others.

### 4.3.1 Memory usage

It is important to note that the measured memory usage not only measures the memory usage of the algorithms, but also includes the memory usage of the test script. It might therefore not be an accurate representation of the resource usage of the algorithm.

The only major variation in the memory usage is between the Raspberry Pi and PC, not between the algorithms. Therefore, any major factor seems device specific, such as hardware or the operating system.

### 4.3.2 Performance multiples

The performance multiples shows that the performance gap on Falcon-512 between the two devices is smaller for verifies than it is for signs, so the Raspberry Pi is better at verifies than signs, compared to the PC. For Rainbow-$I_a$ there is no significant difference.

The performance multiples show a 40% increase in memory usage on the PC. As discussed in section 4.3.1, this has little meaning.

### 4.3.3 Power consumption

The voltage measured on the Raspberry Pi was constant both during the testing and while idling, so the power consumption is entirely proportional to the current consumption. During execution of the test scripts, the Raspberry Pi used all its processing power regardless of the algorithm evaluated. Due to this, it is irrelevant to compare the performance per Watt of the different algorithms since it will come to the same conclusion as just comparing the performance.

# 5. Discussion

This section will discuss the results of the previous section.

## 5.1 Possible installation of RedGeMSS128

It might be possible to modify a more recent version of XKCP to both work with RedGeMSS128 and on the Raspberry Pi. Analysis of the changes in the library has shown that the functionality required by RedGeMSS128 still seems to be present, so with more time and knowledge of XKCP it seems probable that RedGeMSS128 could work on the Raspberry Pi.

### 5.2 Choice of devices

The selection of the devices used was primarily based on what was available. Both devices are more than eight years old but are of approximately the same age and thus comparing them is reasonable. It would have been a better comparison if there were more devices in the data, but there was no access to such devices.

Newer hardware could have more optimizations and new instructions that increase the performance of these algorithms on top of the normal performance increase of newer processors. Due to this, a comparison of the performance on newer and more relevant hardware could lead to other conclusions if the hardware features different optimizations that affect the performance differently.

### 5.3 Number of devices

To make any good and well-founded conclusions about the algorithms' performance, more than one low- and high-power device is needed.

The comparison in this report is intended to be between low- and high-power devices, but since the sample size of both are one, it becomes merely a comparison between two completely different devices. There are a host of different variables that could influence the results that are not necessarily inherent to the type of device. Thus, one should be careful when drawing conclusions from this data. More devices of both types should be used to get more confidence in the conclusion.

### 5.4 Alternative research approach

This report has compared the performance of a single low-power device to a single PC. A more interesting comparison would be multiple low-power devices compared to a set performance requirement that the device needs to meet to be a candidate for a specific use case, such as the analysis in the original paper [3]. Further, more specialized, research could build on this idea and develop a performance requirement and build its analysis on it.

## 6. Conclusion

There are a vast number of applications that will use the algorithm chosen for standardization, all of which have different priorities. Some of the applications will rely heavily on signing, while others will mostly verify. Therefore, it is difficult to say that one of the algorithms is better suited for the general use case, since they cater to different needs. For IoT sensors that would overwhelmingly send data rather than receive, Falcon-512 would be the best choice. For a logging server that would mostly receive data, Rainbow-$I_a$ would instead be the best choice, of course assuming that the vulnerability is fixed and the algorithm performs equivalently to the tests.

The results show that the memory usage is lower on low-power devices and that the relative performance of low-power devices compared to high-power devices is higher for Rainbow-$I_a$ than for Falcon-512. Since these results are produced with a sample size of one for each of the device categories, more tests are needed to get a definitive result.

## References


[1] 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.

[2] National Institute of Standards and Technology, "Announcing Request for Nominations for Public-Key Post-Quantum Cryptographic Algorithms," 20 December 2016. [Online]. Available: https://csrc.nist.gov/news/2016/public-key-post-quantum-cryptographic-algorithms. [Accessed 28 April 2022].

[3] M. Müller, J. de Jong, M. van Heesch, B. Overeinder and R. van Rijswijk-Deij, "Retrofitting Post-Quantum Cryptography in Internet Protocols: A Case Study of DNSSEC," *SIGCOMM Comput. Commun. Rev.*, vol. 50, no. 40, p. 49–57, October 2020.

[4] W. Beullens, "Breaking Rainbow Takes a Weekend on a Laptop," Cryptology ePrint Archive, Report 2022/214, https://ia.cr/2022/214, 2022.

[5] "GNU Time," [Online]. Available: https://www.gnu.org/software/time/. [Accessed 21 April 2022].

[6] G. Bertoni, J. Daemen, S. Hoffert, M. Peeters, G. Van Assche and R. Van Keer, "XKCP/XKCP: eXtended Keccak Code Package," [Online]. Available: https://github.com/XKCP/XKCP. [Accessed 27 April 2022].

## Appendix 1: Measurement data

### Measurement data for the Raspberry Pi

The values for the current is a range between which the current fluctuated during the measurements. Note that only the first three tests for each algorithm have data for voltage and current.

| | Current (mA) | Voltage (V) | Signs per second | Verifies per second | Memory usage (KiB) |
|---|---|---|---|---|---|
| Idle | 180-210 | 5.06 | 0 | 0 | - |
| Falcon-512 | 210-240 | 5.06 | 127.1 | 927.1 | 2664 |
| | 210-240 | 5.06 | 127.5 | 931.9 | 2604 |
| | 210-240 | 5.06 | 127.6 | 931.1 | 2656 |
| | | | 126.2 | 930.5 | 2768 |
| | | | 125.5 | 934.2 | 2596 |
| | | | 126.1 | 932.0 | 2732 |
| | | | 124.8 | 926.4 | 2756 |
| | | | 127.1 | 931.8 | 2732 |
| | | | 126.0 | 933.4 | 2732 |
| | | | 127.6 | 932.5 | 2648 |
| Rainbow-$I_a$ | 210-240 | 5.06 | 292.9 | 318.8 | 2524 |
| | 210-240 | 5.06 | 292.5 | 318.1 | 2528 |
| | 210-240 | 5.06 | 296.1 | 317.8 | 2612 |
| | | | 225.6 | 317.3 | 2616 |
| | | | 293.9 | 317.5 | 2524 |
| | | | 294.8 | 317.2 | 2476 |
| | | | 294.1 | 317.4 | 2468 |
| | | | 231.5 | 316.9 | 2520 |
| | | | 291.0 | 317.7 | 2468 |
| | | | 290.8 | 317.2 | 2528 |

**Measurement data for the PC**

| | Signs per second | Verifies per second | Memory usage (KiB) |
|---|---|---|---|
| Idle | 0 | 0 | - |
| Falcon-512 | 4274.2 | 26191.3 | 3608 |
| | 4290.4 | 26034.1 | 3756 |
| | 4320.2 | 26385.2 | 3708 |
| | 4387.4 | 26235.7 | 3604 |
| | 4421.1 | 26159.6 | 3604 |
| | 4442.0 | 26219.3 | 3680 |
| | 4409.1 | 26192.7 | 3664 |
| | 4318.2 | 26239.3 | 3696 |
| | 4427.8 | 26428.1 | 3692 |
| | 4438.3 | 26251.2 | 3660 |
| Rainbow-$I_a$ | 6926.8 | 7570.4 | 3280 |
| | 6922.3 | 7570.6 | 3452 |
| | 6951.8 | 7571.0 | 3300 |
| | 7108.3 | 7749.7 | 3448 |
| | 7149.9 | 7746.1 | 3404 |
| | 7095.6 | 7734.4 | 3256 |
| | 7100.8 | 7705.7 | 3256 |
| | 7098.5 | 7742.8 | 3176 |
| | 7111.8 | 7746.2 | 3448 |
| | 7114.8 | 7750.3 | 3408 |