Analysis of Post-Quantum Cryptography Algorithms

1st Finelli, G. Evan
Kennesaw State University
Computer Science Major
Marietta, Georgia
evanfinelli37@gmail.com

2nd Escobar, Zachary
Kennesaw State University
Computer Science Major
Marietta, Georgia
Zachary.Escobar65@gmail.com

3rd Ho, Harrison
Kennesaw State University
Computer Science Major
Marietta, Georgia
harrisonho.492@gmail.com

***Abstract*—As the advent of quantum computing poses significant threats to classical cryptographic systems, the development of quantum-safe encryption algorithms has become imperative. This study forms part of a larger effort led by Dr. Manohar Raavi in evaluating the efficiency of post-quantum cryptography (PQC) algorithms and developing methods to make the eventual switch to PQC algorithms seamless. We conducted comparative analysis of several lactic-based signature algorithms identified by the National Institute of Standards and Technology (NIST) for PQC standardization. We assessed algorithms submitted to NIST's Round 1 Additional Signatures. Through implementing these algorithms we assessed their computation speed and compared them to the lattice-based signature algorithm Falcon that was standardised in NIST's first call for proposal.**

***Index Terms*—algorithm, cryptography, NIST, quantum**

I. Introduction

In response to the escalating threat posed by quantum computing to the cryptographic standards of today, NIST has embarked on an initiative to identify viable PQC algorithms and standardise them. As these algorithms are standardised, the next step will be implementing them into the already existing protocols that exist today. Dr. Manohar Raavi, a Graduate Professor at Kennesaw State University, is exploring methods to help facilitate the eventual switch from modern cryptography algorithms to PQC solutions. This work covers many areas of computer science from topics in networking protocols to quantum computing and everything in between. To assist in this endeavor, our team, MR-1 Blue Team Cryptography, has been tasked with evaluating the computation speeds of lattice-based signature algorithms from NIST's round one additional signatures. We further evaluated these algorithms by comparing them to on of NIST's already standardised PQC algorithms, Falcon. This will help visualize how these new lattice-based signature algorithms submissions compare to an already accepted one. Understanding the capabilities of these algorithms will help in identifying the uses cases for them in the broader conversation of implementing PQC algorithms into current systems. This paper will give background on the subject of NIST's call for proposal and then trace the developmental history of the project, detailing how our project and focus evolved over time. Following this we will detail our methodology, present our findings and discuses the implications of these results.

II. Background

In December of 2016, NIST publicly announced that they were accepting submissions to the PQC standardization process [1]. NIST held three main rounds of submissions. Algorithms were evaluated at each round. Only the most viable and promising algorithms made it to the next round. Sixty-nine algorithms were submitted to the first round. At the end of round three, in July of 2022, four algorithms were selected to be standardised. One Public-Key Encryption (KEM) algorithm, CRYSTALS-Kyber, and three digital signatures, CRYSTALS-Dilithium, FALCON, and SPHINCS+. Besides SPHINCS+, all of the algorithms are based on structured lattices and the computational hardness of problems involving them [1]. NIST still has four KEM algorithms under consideration that have gone on to a fourth round. Due to no more digital signatures being under consideration for standardization, NIST announced, with the announcement of the chosen algorithms, that another call for proposals for digital signatures would come by the end of the summer. In September of 2022, NIST announced this call for proposals of additional digital signature schemes. They requested that these algorithms focus on general purpose signature schemes that had short signatures, fast verification and not lattice-based so as to diversify the post-quantum signature standards. They were, however, still open to lattice-based signatures so long as they "significantly outperform CRYSTALS-Dilithium and FALCON in relevant applications and/or ensure substantial additional security properties to be considered for standardization" [1]. They received 40 applications. They received six code-based signatures, one Isogeny Signature, seven lattice-based signatures, seven MPC-in-the-Head signatures, ten Multivariate signatures, four Symmetric-based signatures, and five Other signatures. As our project changed, (see Section III, Project Evolution) we decided to focus on evaluating the lattice-based signatures. Since they had the requirement of outperforming CRYSTALS-Dilithium and FALCON, we implemented those as well and tested them to allow us to draw conclusions about the performance if the new signatures. The lattice-based signatures for round one additional signatures include EagleSign, EHTv3 and EHTv4, HAETAE, HAWK, HuFu, Raccoon, and SQUIRRELS.

III. Project Evolution

Starting the research project began with Dr. Raavi having us do some preliminary research to familiarise and/or refresh

ourselves with some cryptography basics. We looked into the differences between Symmetric and Asymmetric cryptography. Then Dr. Raavi gave us our project goal. To implement and test the computation speed of the keypair, sign and verify functions of all 40 algorithms that were submitted to NIST's round one additional signatures. We started by documenting the private key size, public key size, and signature size, in bytes, of all the algorithms. This information would allow us to compare the run times of the algorithms along with how large their key and signature sizes were to see the if there is a correlation of key size to run time. This would also allow us to see if any algorithms ran faster despite a larger key and/or signature size. Once that was complete we started on implementing the algorithms. Very quickly we realized that this would prove to be very challenging. After repeated failures in attempting to implement and run even one algorithm, we decided to consolidate our efforts. Dr. Raavi suggested we focus on the lattice-based signature submissions. We concluded that evaluating these would give us the most worth while results since Lattice-based algorithms were requested to significantly outperform CRYSTALS-Dilithium and FALCON for round one additional signatures. With this in mind, we now had to also implement and evaluate CRYSTALS-Dilithium and FALCON so that we could compare their results with the new submissions.

IV. Methodology

A. Challenges

Implementing and testing the algorithms was not a straight forward process. All algorithms come with a website as well as a zip file containing the code needed to run the algorithm. The first step to every algorithm was reading its documentation and any README files inside the code. We quickly noticed that each algorithms was unique. The order of files, name of files, instructions, make files and various other things were different. Progress was slow and effort was often unfruitful. Under Dr. Raavi's guidance we looked at NIST's call for proposal document. This document outlined all requirements for submission. We looked for anything that could help point us in the right direction. We also looked at open quantum safe's open source GitHub repository of PQC algorithms, liboqs, for direction on implementing algorithms. Liboqs contains the several PQC algorithms including NIST's standardised algorithms such as FALCON. We attempted to look at these algorithms and use their implementation in liboqs as a template to integrate other algorithms. However, Liboqs pulls the existing algorithms from a libary called PQClean where the developers rewrote each algorithm to easily be implemented in encryption libraries. This meant any algorithms we wanted to add support for would require a rewrite as well. One example of this was with Hawk that used different data types than those found in Liboqs api, this means we would have to go through the code and replace the variables and method calls. This is mostly a time consuming process, but due to our lack of experience with these algorithms, we weren't sure if the authors chose those datatypes for a reason and if any changes we made would negatively affect preformance. Our next issue was trying to figure out how to implement the new algorithm into a forked PQClean so that Liboqs would detect it and automatically update its configuration files. We had to take this route because even when we did work out all the errors and compiled properly, the new algorithms would not appear in the speed test and we believed properly implementing it into PQClean would solve this. There was no guide on how to do that so the only choice we've had so far is to copy the existing files for algorithms and change their code to connect to the newer algorithms. However, at this time this has not lead to the algorithms properly running in the speed tests.

B. Successes

Our first success came when we attempted to run EagleSign and HaeTae using Kennesaw State University's High Performance Computing (HPC) Environment [2], [3]. The idea behind using the HPC was that it would give us consistent results since the computing power wouldn't be affected by issues on our own local devices. It also allowed us to have an environment where we can all run tests using the exact same hardware. However, HaeTae and EagleSign were the only two submissions that came prepackaged with test files. For the rest of the algorithms we would have to write the tests from scratch and these tests could be inefficient due to our lack of knowledge surrounding the algorithms actual logic. Our next plan is to use Open Quantum Safe's Liboqs and try to implement the algorithms into the library's api using the already existing algorithms as reference. Liboqs generates the arguments and comes packaged with executable tests that handle all implemented algorithms, which allows us to focus on the algorithms rather than the test programs. We were able to successfully test CRYSTALS-Dilithium and FALCON. From this we were now able to compare EagleSign and HaeTae.

V. Results

In this section, we present the findings of the lattice-based algorithms that were successfully compiled and tested. The results demonstrate the computational speed of the KeyPair, Sign, and Verify functions. The algorithms successfully tested were EagleSign at security levels three and five as well as HaeTae at security levels two, three, and five. Figure 1 shows how each algorithm compares to each other. We can see that the HaeTae algorithms did significantly better than EagleSign at security level 5. At level two and three, however, it took longer than EagleSign's level 3 algorithm. Figure two shows the results of Dilithium and Falcon. They show considerably less cycle/tics needed to perform the same functions. As of now, this is because our testing for EagleSign and HaeTae was done using their built in testing functions. Since they are not using the same test parameters, we can not yet draw any conclusions on whether or not EagleSign and HaeTae outperform Dilithium and/or FALCON.

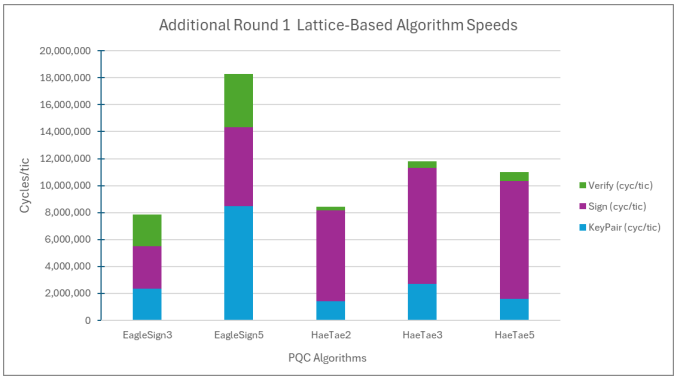

Fig. 1: Lattice-Based Algorithm Results

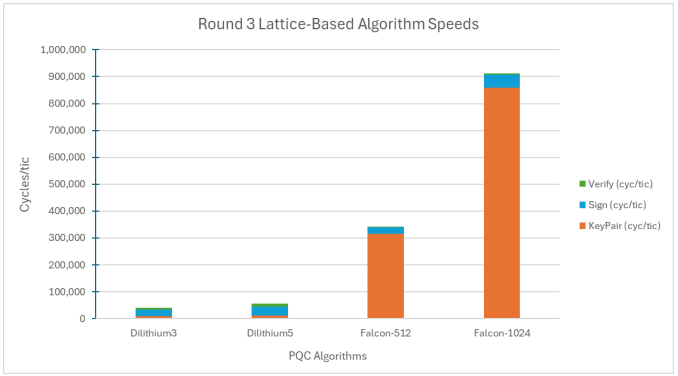

Fig. 2: Standardised Lattice-Based Algorithm Results

VI. Conclusion

Our results as of now are inconclusive. We are still attempting to implement the lattice-based algorithms into LibOQS so that they can be tested against the same test parameters. While our research did not yield the conclusive results we were aiming for, significant insights have been gained through the process. The challenges we encountered in attempting to implement and test these PQC algorithms highlighted areas for improvement in our methodology. It is clear that immediately trying to implement these algorithms was too difficult. More preliminary research and analysis of the problem and our teams capabilities and understanding of the subject would have helped us greatly. We were able to run two of the algorithms with promising outlooks for a few more. We were then able to run and test Dilitihium and FALCON on LibOQS. We will be able to draw a meaningful conclusion once we are able to implement and run our round 1 additional lattice-based algorithms on LibOQS.

References

[1] "Round 1 additional signatures," https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures Last accessed: 04/14/2024, 2022.

[2] D. Sow, A. Hounkpevi, S. Djimnaibeye, and M. Seck, "Eaglesign : A new post-quantum elgamal-like signature over lattices," https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/EagleSign-spec-web.pdf Last accessed: 04/26/2024, NIST, 2023.

[3] J. D. T. G. D. H. M. K. G. L. J. S. D. S. M. Y. Jung Hee Cheon, Hyeongmin Choe, "Haetae," https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/haetae-spec-web.pdf Last accessed: 04/26/2024, NIST, 2023.