Parallel Implementation of MAYO on GPU*

JunHyeok Choi, DongCheon Kim, and Seog Chung Seo

Department of Financial Information Security, Kookmin University, Seoul, South Korea
{c981126, kindongsy, scseo}@kookmin.ac.kr

Abstract

With the advancement of quantum computers, NIST has been conducting a standardization project for post-quantum cryptography. In 2022, NIST selected one KEM and three DSAs as standardization candidates. However, due to the issue that most of the selected algorithms are lattice-based, NIST is currently holding an additional competition to select more DSAs in 2023. MAYO is a multivariate signature algorithm with a structure similar to the Oil and Vinegar scheme. It is characterized by its use of a small secret key and a large amount of stack memory. This paper aims to lay the groundwork for GPU optimization research on MAYO by identifying its performance bottlenecks and presenting the throughput when porting MAYO to the GPU in a simple manner. Additionally, we discuss the precautions in the porting process and suggest future research directions for optimizing MAYO.

Keywords: MAYO, PQC, GPU, Parallel Implementation

1 Introduction

The advancement of quantum computers threatens traditional cryptosystems like RSA and ECC, prompting NIST to launch the PQC Standardization competition in 2016. By 2022, Crystals-Kyber (PKE/KEM) and three DSAs (Crystals-Dilithium, SPHINCS+, and Falcon) were selected, but concerns about the dominance of lattice-based algorithms led to an additional DSA competition in 2023. MAYO, based on the UOV problem, is competing in this round [1]. With GPUs' parallel processing capabilities, research on post-quantum cryptography optimizations using CUDA is ongoing [2, 3]. This study identifies MAYO's bottlenecks and explores basic GPU parallelization strategies.

2 MAYO profiling

In this section, we discuss the performance bottlenecks of the MAYO algorithm. The C language implementation code provided by the MAYO team on GitHub was used. Among the four security levels of MAYO, MAYO-1 was executed on a CPU, and the clock cycles for each key internal operation were measured. For CPU performance measurement, we utilized an Intel(R) Core(TM) i7-13700K processor. The development and performance measurement environment was set up in Visual Studio on Windows, and the measurements were conducted in release mode, running 50 iterations to obtain the average values. Table 1 shows the core operations for key generation, signing, and signature verification in MAYO-1, as well as the corresponding performance bottlenecks for each operation.

*Proceedings of the 8th International Conference on Mobile Internet Security (MobiSec'24), Article No. P-20, December 17-19, 2024, Sapporo, Japan.

Table 1: MAYO-1 performance profiling

keygen		sign		verify	
cc_PK_PRF	1,070,523 (61.4%)	sign_cc_mayo_expand_sk	2,108,314 (59.9%)	verify_cc_mayo_expand_pk	1,105,363 (61.6%)
cc_P1_times_O	498,949 (28.6%)	sign_cc_P1_times_Vt	561,787 (16%)	verify_cc_bitsliced_m_calculate_PS_SPS	666,787 (37.2%)
...		...		...	
Total	1,743,228	Total	3,519,944	Total	1,794,260

3 MAYO on GPU

In this section, we present the throughput when porting each of the four security levels of MAYO to the GPU using the 1-thread-1-algorithm approach. For GPU performance measurement, we utilized an NVIDIA GeForce RTX 4090. Both the grid and block sizes were set to 4, and the average values were obtained by repeating the measurements five times. MAYO uses a large amount of stack memory during its computation process, which limits its operation on the GPU. To address this, the stack memory is converted to heap memory if necessary, and the size of the heap memory is increased using the cudaDeviceSetLimit function. Table 2 shows the throughput of the entire MAYO scheme on both CPU and GPU for each variant, as well as the latency on the GPU. Throughput represents the number of entire MAYO schemes that can be processed per second. It also presents the performance improvement ratio between the CPU and GPU.

Table 2: Performance comparison across different MAYO variants on CPU and GPU

Category	**MAYO-1**	**MAYO-2**	**MAYO-3**	**MAYO-5**
Throughput on CPU (MAYO per sec)	481.76	340.25	150.04	81.42
Throughput on GPU (MAYO per sec) (Performance improvement)	554.41 (+15.08%)	385.92 (+13.42%)	161.43 (+7.59%)	63.49 (-22.02%)
Latency on GPU (ms)	115.46	165.87	396.48	1008.09

4 Conclusion

This paper presents the performance bottlenecks of MAYO, a multivariate-based DSA submitted to NIST's additional DSA selection competition, and its throughput on a GPU. GPUs are specialized in throughput-focused architectures, which require executing a large number of operations in parallel simultaneously. However, since MAYO heavily relies on stack memory during its execution, processing many operations simultaneously using a 1-thread-1-algorithm approach becomes challenging. Therefore, future optimizations could involve reducing the amount of stack memory used in MAYO operations or dividing a single internal operation into multiple threads for parallel processing.

Acknowledgement: This work was supported by the National Research Foundataion of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2022R1C1C1013368).

References

[1] W. Beullens. Mayo: practical post-quantum signatures from oil-and-vinegar maps. In *International Conference on Selected Areas in Cryptography*, pages 355–376. Springer, 2021.

[2] S. Shen, H. Yang, W. Li, and Y. Zhao. cuml-dsa: Optimized signing procedure and server-oriented gpu design for ml-dsa. *Cryptology ePrint Archive*, 2023.

[3] T. Zhou, F. Zheng, G. Fan, L. Wan, W. Tang, Y. Song, Y. Bian, and J. Lin. Convkyber: Unleashing the power of ai accelerators for faster kyber with novel iteration-based approaches. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024(2):25–63, 2024.