周报

2025 年 2 月 18 日

摘要

- 论文阅读与对比分析
- 创新方向确定
- 论文写作

论文创新点对比

为扩展研究视角并探索创新方向，在深入研读了 [WDC+25] 后，我们进一步阅读了另外两篇论文 [KCS24] 和 [NDL+24]。GPU 平台之所以被选取，主要源于其在并行计算性能上具有显著优势。通过比较这三篇论文的创新点（见表 1），可以看出，各论文均针对 SPHINCS+ 提出了并行优化方法，但其具体的优化策略各有侧重。

表 1: 三篇论文创新点比较

论文	创新点
Kim et al. [KCS24]	关键组件并行处理方案，存在多次 CUDA 内核调用效率瓶颈
Wang et al. [WDC+25]	CUSPX 框架，创新性并行策略与负载均衡机制
Ning et al. [NDL+24]	GRASP 方案，基于自适应并行与内核融合技术

创新点讨论

上述研究主要关注两个核心性能指标：吞吐量和延迟。其中，吞吐量衡量在固定核心数条件下 GPU 处理签名任务的能力，其效果主要受并行效率（PE）的影响；而延迟则反映了签名任务的并行执行程度。为提升吞吐量，相较于采用静态并行数设置，我们计划根据各组件的运行时长动态调整并行数，从而进一步提高 PE。此外，代码阅读过程中发现目前底层的 HASH 函数采用串行实现，这在一定程度上制约了并行效率。因此，计划将其替换为 GPU并行 HASH 函数，以期降低延迟并进一步提升系统性能。

性能指标关注

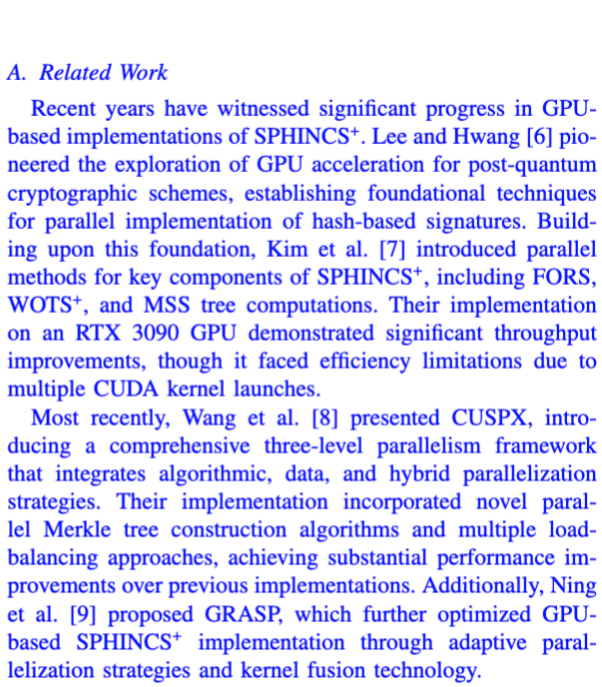
- 吞吐量：并行效率（PE）
- 延迟：任务并行度

改进方向

- 动态并行数调整
- GPU并行 HASH 函数实现

论文写作

- 目标期刊：IEEE Transactions on Circuits and Systems II: Express Briefs. 页数限制：5 页以内. **Introduction** 部分 (图 1) 完成.

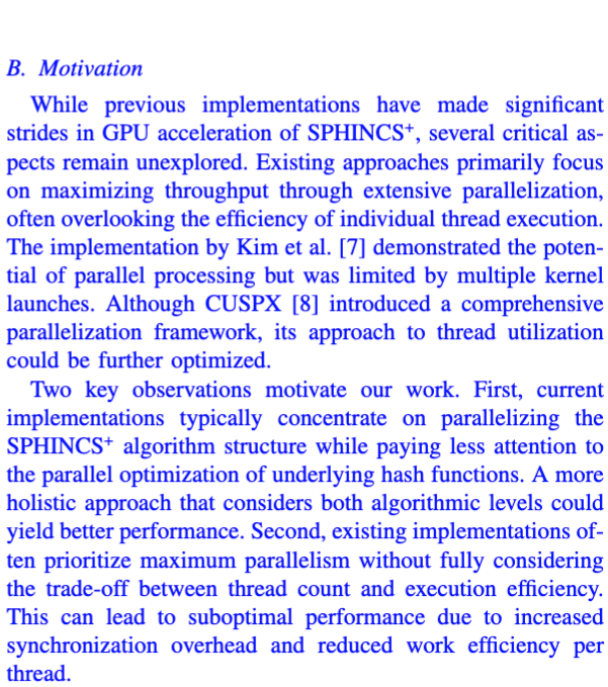


A. Related Work

Recent years have witnessed significant progress in GPU-based implementations of SPHINCS⁺. Lee and Hwang [6] pioneered the exploration of GPU acceleration for post-quantum cryptographic schemes, establishing foundational techniques for parallel implementation of hash-based signatures. Building upon this foundation, Kim et al. [7] introduced parallel methods for key components of SPHINCS⁺, including FORS, WOTS⁺, and MSS tree computations. Their implementation on an RTX 3090 GPU demonstrated significant throughput improvements, though it faced efficiency limitations due to multiple CUDA kernel launches.

Most recently, Wang et al. [8] presented CUSPX, introducing a comprehensive three-level parallelism framework that integrates algorithmic, data, and hybrid parallelization strategies. Their implementation incorporated novel parallel Merkle tree construction algorithms and multiple load-balancing approaches, achieving substantial performance improvements over previous implementations. Additionally, Ning et al. [9] proposed GRASP, which further optimized GPU-based SPHINCS⁺ implementation through adaptive parallelization strategies and kernel fusion technology.

(a) 相关工作部分

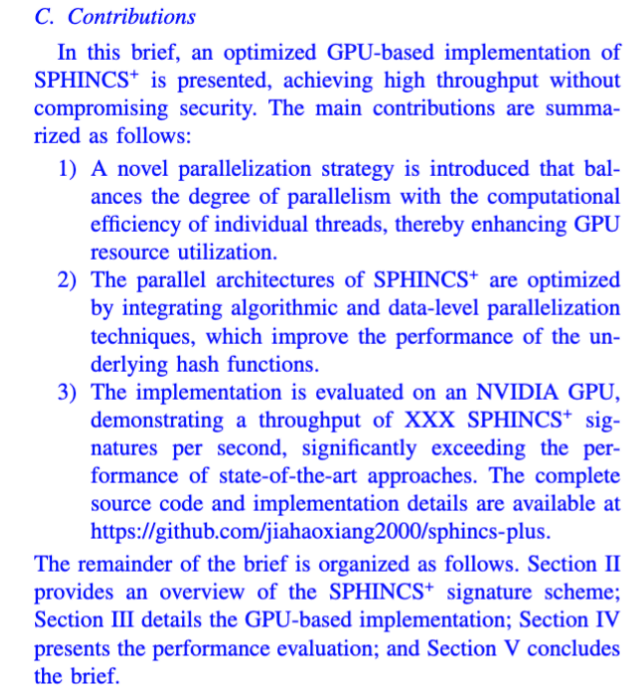


B. Motivation

While previous implementations have made significant strides in GPU acceleration of SPHINCS⁺, several critical aspects remain unexplored. Existing approaches primarily focus on maximizing throughput through extensive parallelization, often overlooking the efficiency of individual thread execution. The implementation by Kim et al. [7] demonstrated the potential of parallel processing but was limited by multiple kernel launches. Although CUSPX [8] introduced a comprehensive parallelization framework, its approach to thread utilization could be further optimized.

Two key observations motivate our work. First, current implementations typically concentrate on parallelizing the SPHINCS⁺ algorithm structure while paying less attention to the parallel optimization of underlying hash functions. A more holistic approach that considers both algorithmic levels could yield better performance. Second, existing implementations often prioritize maximum parallelism without fully considering the trade-off between thread count and execution efficiency. This can lead to suboptimal performance due to increased synchronization overhead and reduced work efficiency per thread.

(b) 动机说明



C. Contributions

In this brief, an optimized GPU-based implementation of SPHINCS⁺ is presented, achieving high throughput without compromising security. The main contributions are summarized as follows:

1) A novel parallelization strategy is introduced that balances the degree of parallelism with the computational efficiency of individual threads, thereby enhancing GPU resource utilization.
2) The parallel architectures of SPHINCS⁺ are optimized by integrating algorithmic and data-level parallelization techniques, which improve the performance of the underlying hash functions.
3) The implementation is evaluated on an NVIDIA GPU, demonstrating a throughput of XXX SPHINCS⁺ signatures per second, significantly exceeding the performance of state-of-the-art approaches. The complete source code and implementation details are available at https://github.com/jiahaoxiang2000/sphincs-plus.

The remainder of the brief is organized as follows. Section II provides an overview of the SPHINCS⁺ signature scheme; Section III details the GPU-based implementation; Section IV presents the performance evaluation; and Section V concludes the brief.

(c) 创新点部分

图 1: 组合展示：相关工作、动机与创新点

老师评语

继续推进

下周计划

1）GPU 并行 HASH 函数设计与实现，降低延迟；
2）动态并行数调整策略，提高吞吐。

参考文献


DongCheon Kim, Hojin Choi, and Seog Chung Seo.
Parallel implementation of SPHINCS+ with gpus.
IEEE Trans. Circuits Syst. I Regul. Pap., 71(6):2810–2823, 2024.


Yijing Ning, Jiankuo Dong, Jingqiang Lin, Fangyu Zheng, Yu Fu, Zhenjiang Dong, and Fu Xiao.
GRASP: Accelerating hash-based PQC performance on GPU parallel architecture.
Cryptology ePrint Archive, Paper 2024/1030, 2024.


Ziheng Wang, Xiaoshe Dong, Heng Chen, Yan Kang, and Qiang Wang.
Cuspx: Efficient gpu implementations of post-quantum signature sphincs⁺.
IEEE Transactions on Computers, 74(1):15–28, 2025.