周报-向嘉豪 (2025 年 2 月 17 日)

摘要: 本周工作主要围绕 SPHINCS$^+$ 在 GPU 平台上的加速实现展开。我们深入阅读并比较了 Kim et al. [KCS24]、Wang et al. [WDC$^+$25] 与 Ning et al. [NDL$^+$24] 三篇文献，探讨了各研究在关键组件并行处理、并行策略设计以及内核融合技术等方面的创新点。文献分析揭示了当前实现中在吞吐量和延迟方面存在的瓶颈，为进一步提升系统性能提供了改进思路，如采用 GPU 并行 HASH 函数及动态调整并行度等策略。此外，我们还明确了论文写作的风格要求，为投稿的《IEEE Transactions on Circuits and Systems II: Express Briefs》简报文章做准备。

下周计划: 1) 针对当前 SPHINCS$^+$ 实现中采用的 CPU 串行 HASH 函数，设计并实现 GPU 并行 HASH 函数，并进行初步性能评估；2) 探索并实现基于各个组件运行时长的动态并行数调整策略，以期进一步提升整体吞吐量；

1 论文与代码阅读

阅读: 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。

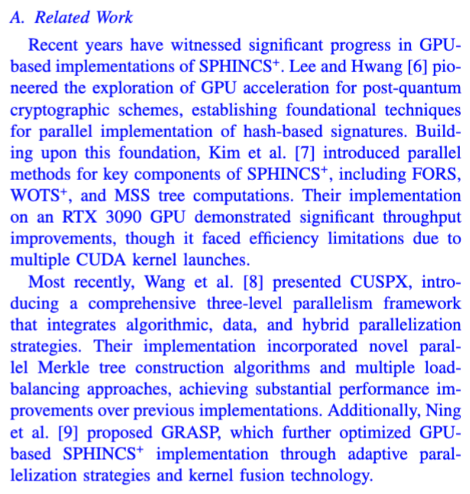
表 1: 三篇论文创新点比较

论文	创新点
Kim et al. [KCS24]	针对 SPHINCS$^+$ 的关键组件（FORS、WOTS$^+$、MSS 树）提出并行处理方案，在 RTX 3090 GPU 环境下实现了吞吐量的显著提升，但因多次 CUDA 内核调用而存在效率瓶颈。
Wang et al. [WDC$^+$25]	提出了 CUSPX 框架，结合算法级、数据级与混合并行策略，引入了创新性的并行 Merkle 树构建算法和多重负载均衡机制，从而实现了较以往更为显著的性能提升。
Ning et al. [NDL$^+$24]	基于自适应并行策略与内核融合技术提出 GRASP 方案，进一步优化了 GPU 上 SPHINCS$^+$ 的实现效果，显著提升了整体运行效率。

创新点讨论: 本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 CPU 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。

2 论文写作

写作风格: 为向《IEEE Transactions on Circuits and Systems II: Express Briefs》简报投稿，i.e., 该期刊于 2022 年和 2025 年分别接受了 GPU 加速 AES 和 PQC 实现方面的研究，为此我们使用 GPU 加速 PQC 实现，在其投稿范围内。该简报文章的写作风格要求简洁明了，重点突出，且需在5 页以内。因此，我们将在写作过程中注重逻辑严谨、重点突出，并在图表设计上下功夫，以提升文章的可读性。
具体写作: 本周主要完成文章的介绍部分，就 GPU 实现 SPHINCS$^+$ 相关工作进行了梳理，并对本文的动机和创新点进行了阐述。具体见图 1。

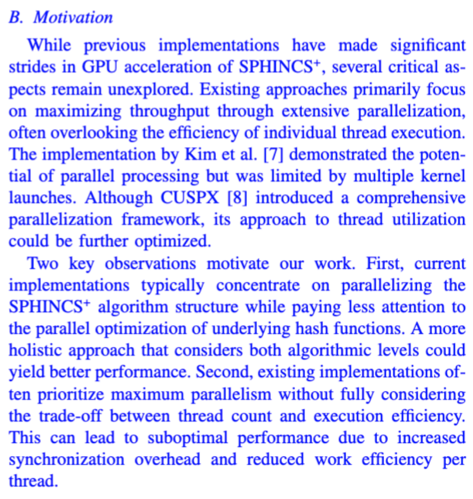


A. Related Work

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

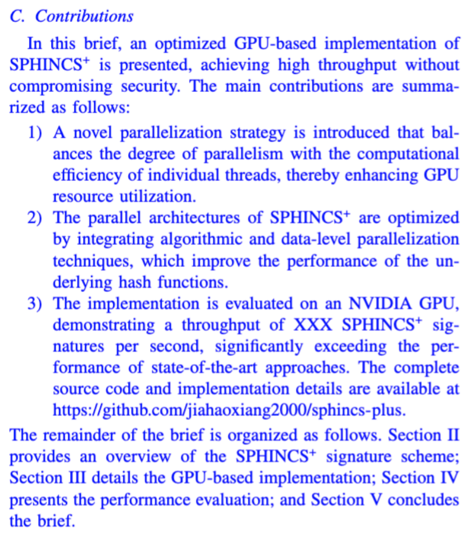


B. Motivation

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

参考文献

[KCS24] 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.

[NDL$^+$24] 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.

[WDC$^+$25] Ziheng Wang, Xiaoshe Dong, Heng Chen, Yan Kang, and Qiang Wang. Cuspx: Efficient gpu implementations of post-quantum signature sphincs<sup>+</sup>. *IEEE Transactions on Computers*, 74(1):15–28, 2025.