Australia's National Science Agency

High Throughput Lattice-based Signatures on GPUs: Comparing Falcon and Mitaka

Wai-Kong Lee, Raymond K. Zhao, Ron Steinfeld, Amin Sakzad, Seong Oun Hwang

Post-quantum Cryptography (PQC)

- Classical Public Key Cryptography (PKC) in use e.g. ECDH can be broken by quantum algorithms.
  - Harvest now, decrypt later attack.
- National Institute of Standards and Technology (NIST) has standardized *Post-Quantum Cryptography* (PQC) algorithms in 2022.
  - ML-KEM, ML-DSA, SLH-DSA, Falcon, …
- Existing applications using PKC e.g. web security needs to be quantum-safe.
  - May need to develop new protocols.
- *Everyone* needs to migrate to quantum-safe!

PQC Research in CSIRO's Data61

- Our people:

Dongxi Liu

Raymond K. Zhao

Jiafan Wang

PQC Research in CSIRO's Data61

- Our strength: We have *both* cryptography and cryptographic engineering expertise for PQC.
  - We develop solutions that meet different requirements.
  - We develop prototype implementations for new PQC designs.
  - We are working with industrial partners to develop and implement new PQC migration technologies.

Background

- Cloud server needs to handle thousands of digital signatures per second at peak time.
  - e.g. Alibaba: 583,000 transactions/sec → 583,000 signings, 1,166,000 verifications per second.
  - With only CPU, challenging even for powerful servers.
- Cloud starts to equip with *Graphics Processing Units* (GPU) thanks to AI.

Falcon and Mitaka Signature

- Digital signatures in use are not quantum-safe. Need to migrate to PQC.
  - *Falcon* is one of the NIST standardized PQC digital signatures.
  - *Mitaka* is a later more parallelizable variant of Falcon.
    - Techniques adopted by the *SOLMAE* signature in Korean PQC competition.
- Question: How to efficiently implement Falcon/Mitaka on GPU?
  - We develop the first GPU implementation for Falcon and Mitaka.
  - Collaboration between Gachon University, CSIRO's Data61, and Monash University.
  - Published in IEEE Transactions on Parallel and Distributed Systems.

CSIRO

Problem 1: Recursive vs Iterative *ffSampling*

- *ffSampling* is the most time-consuming operation in Falcon signature generation.
- The sampling process traverse through a Falcon tree.
- It was first implemented by the authors in a recursive manner.
- Not parallelizable due to dependency between leaves.
- Very slow on a GPU due to extensive stack management and lack of efficient API support.

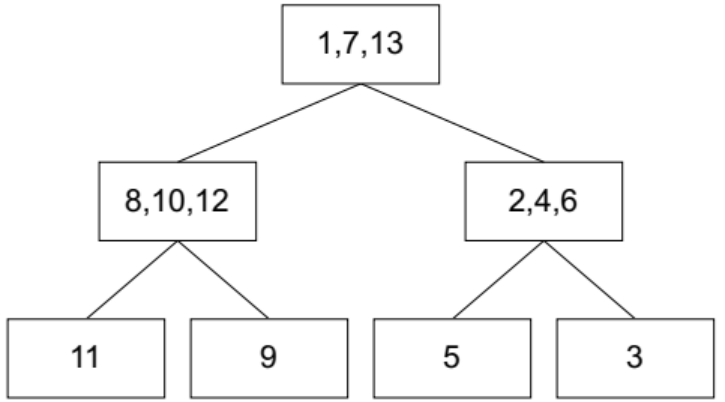


Fig. 2: A Falcon tree of height 3. Labels on the tree nodes indicate the order in which the tree nodes are traversed by the ffSampling algorithm (where 1 is visited first, 2 second, etc).

Solution 1: Proposed Iterative *ffSampling*

- Existing GPU implements recursive function call by dynamic parallelism, which launches kernel within a kernel. This introduces significant overhead if there are many levels of recursion.
- Proposed an iterative version of *ffSampling* to replace the original recursive version.
- Do not rely on the recursive API (dynamic parallelism).

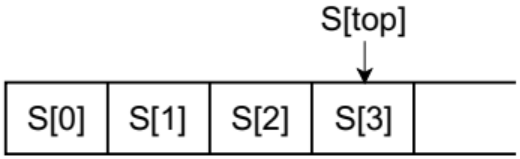


Fig. 3: A stack array S. Each array element contains a stack frame.

- Stack can be managed by the programmer in a simpler way (Fig. 3). Only a 1-D array is required.

Solution 1: Micro-benchmarking

- We compared the performance of recursive and iterative *ffSampling.*
- The proposed iterative version is 11.44× – 14.39× faster than the original recursive version.

TABLE 8: Throughput of Falcon-512 signature generation using recursive and iterative *ffSampling*

	GPU	Falcon-512 Recursive		Falcon-512 Iterative	
		Op/s	ms	Op/s	ms
K=16384	RTX 3080	2439	6717.51	27908	587.07
	V100	2898	5653.36	37100	441.62
	T4	1009	16237.86	12934	1266.74
	A100	4025	4070.56	35595	279.61

Problem 2: Parallel Granularity

- Previous works adopt either coarse- or fine-grained parallelism.
- Coarse-grain:
  - 1 thread 1 signature/KEM.
  - Serial implementation, does not fully exploit GPU resources.
  - Easy to implement.
- Fine-grain:
  - Many threads compute 1 signature/KEM.
  - Parallel implementation, better use of GPU resources.
  - More work required to implement this.

Solution 2: Proposed Mixed Parallel Granularity

- Proposed a mixed parallel granularity that combines the best of both world.
- Many parallel blocks, each block computes 1 signature.
- Within each block, parallelize 1 signature with many threads.
- For some parts not possible to parallelize, we implement it serially.

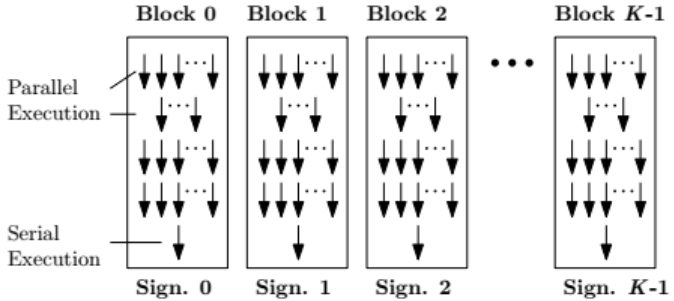


Fig. 1: GPU-based signature server: parallelizing the implementation of signatures on a GPU.

Problem 3: Slow Mitaka Verification

- Falcon verification is fast because it is parallelizable and operates on integer arithmetic.
- However, the original Mitaka verification was implemented using double precision floating point arithmetic. It is slower than the Falcon verification.
- Mitaka verification shares many similarity with Falcon, hence it is also possible to work on the integer domain.
- In this paper, we evaluate Mitaka verification in both floating point and integer domain.

Solution 3: Proposed Mitaka Implementation on the Integer Domain

- The FFT/iFFT is converted to NTT/iNTT.
- Integer arithmetic uses 32-bit operations, which is natively supported by GPU platforms.
- Double precision floating point arithmetic (FP64) operates on 64-bit, not natively supported by GPU platforms, thus slow.

TABLE 7: Throughput of Mitaka-512 verification on floating point and integer units

	GPU	Mitaka-512 FP64		Mitaka-512 INT32	
		Op/s	ms	Op/s	ms
K=16384	RTX 3080	695931	23.54	2007649	8.16
	V100	843035	19.44	2130201	7.69
	T4	331619	49.41	918191	17.84
	A100	1421046	11.53	3790838	4.32

- On an A100 GPU, the integer version (INT32) of Mitaka verification is 2.67× faster than the floating point version (FP64).
- This result is also 1.39× faster than Falcon- 512 verification on the same GPU device.

Other Techniques:

- Mitaka signature generates random samples in on-demain basis. This is slow for a GPU implementation. We proposed a technique to generate random samples in batch.
- Polynomial arithmetic, FFT/iFFT and NTT/iNTT are embarrassingly parallel algorithms, which fit into GPU implementation easily.
- Hash operations are parallelized by 25 threads, following the fine-grained implementation of SHA3 proposed by Lee et al. [44]. This is also the fastest SHA3 fine-grained implementation on GPUs to date.

Experimental Platforms

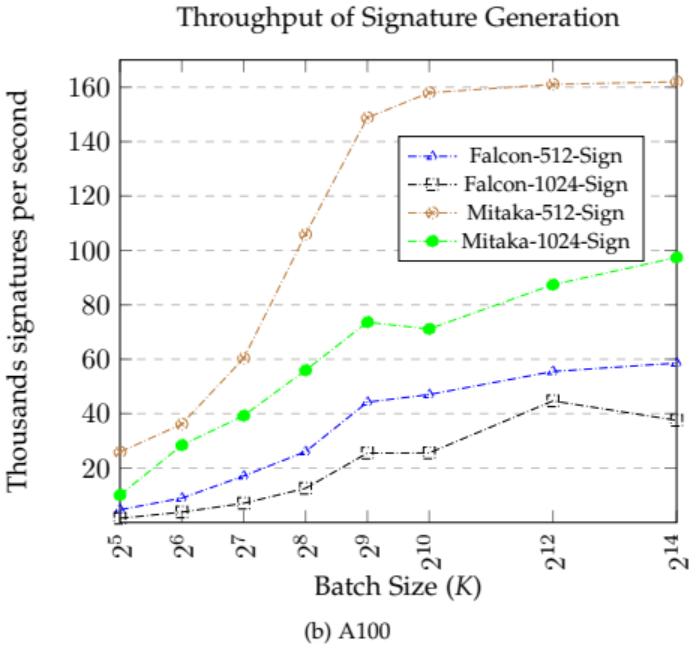
- Experiments were carried out on two separate platforms.
- Four state-of-the-art NVIDIA GPU architectures: Volta (V100, 2017), Turing (T4, 2018) and Ampere (A100, 2020; RTX 3080, 2021) were used.
- *K* GPU blocks are launched to generate/verify *K* signatures in parallel.
- Within each block, multiple threads are used to compute one signature.

TABLE 4: Experimental Platforms Used

	Platform-1	Platform-2		
	Desktop Workstation	Cloud System		
GPU	RTX 3080	V100	T4	A100
CUDA Cores	8704	5120	2560	8192
Architecture	Ampere	Volta	Turing	Ampere
Compute capability	8.6	7.0	7.5	8.0
Clock (GHz)	1.710	1.246	0.585	1.410
Memory bandwidth (GB/s)	760	900	300	1935
No. Streaming Multiprocessor (SM)	68	80	40	64
Compiler	CUDA 11	CUDA 11		
CPU	Intel i9-10900K	Intel Xeon Gold 6150		
Clock	3.70 GHz	2.2 GHz		

Signature Throughput on a A100 GPU

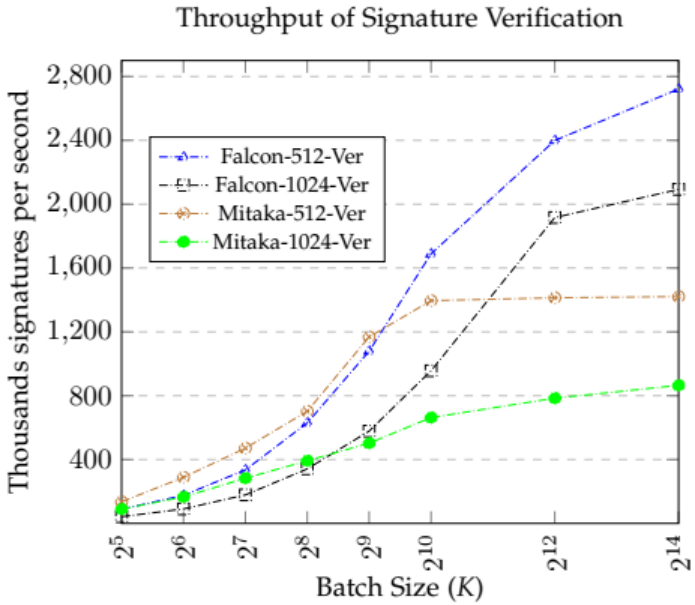
- On a A100 GPU, Mitaka-512 signature generation is 2.76× faster than Falcon-512.
- Falcon is hard to parallelize due to the serial *ffSampling* algorithm.
- Mitaka can be fully parallelized, thus achieving much higher throughput than Falcon.



(b) A100

Verification Throughput on a A100 GPU

- On the other hand, Falcon-512 verification is 1.91× faster than Mitaka-512.
- Mainly because Falcon uses integer arithmetic in verification, but Mitaka uses double precision floating point.
- Refer to slide no. 13, Mitaka verification is faster than Falcon when both are using integer arithmetic.



(b) A100

Comparison with Existing Works

- Our Falcon-512 implementation on RTX 3080 is 7.78× and 52.43× faster than the AVX2 implementation for sign and verify, respectively.
- Our GPU implementation is 9.15× (sign)/14.45× (verify) faster than FP64 Mitaka-512 (reference implementation from the authors).

TABLE 6: Comparing with CPU and state-of-the-art implementations.

		Fal-512	Mit-512	Fal-1024	Mit-1024
		CPU, Op/s			
AVX2	Sign	3167	-	2850	-
	Verify	18230	-	18319	-
FP64	Sign	3587	8087	2045	4146
	Verify	36491	48153	17694	20565
		GPU, Op/s			
This work [1] (RTX 3080)	Sign	27908	74010	15239	34025
	Verify	1913380	695931	1217317	309841
XMSS_10 [46] (SHA-256)	Sign	225396/186913[2]			
	Verify	730450/605739[2]			
Dilithium [26]	Sign	717306/580676[3]			
	Verify	1960182/1586814[3]			
SPHINCS [14] (ChaCha)	Sign	5152/17516[4]			
	Verify	106390/361726[4]			

[1] The highest throughput with $K = 16384$.
[2] Performance scaled by the number of cores, 10496/8704. RTX 3090 was used in [46].
[3] Performance scaled by 10752/8704 RTX 3090 Ti was used in [26].
[4] Performance scaled by 2560/8704. GTX 1080 was used in [14].

CSIRO

Conclusions

- Our work pioneered the attempt to implement Falcon and Mitaka on various state-of-the-art GPU platforms.
- The proposed implementation techniques allows Falcon and Mitaka to achieve very high signature and verification throughput.
- Can help in accelerating the adoption of NIST standard in applications that require high throughput signatures.
- The proposed iterative *ffSampling* can also be adopted to hardware implementation like FPGA.
- Our Mitaka GPU implementation could be adopted to implement SOLMAE on GPU.

Thank you

CSIRO's Data61
Raymond Zhao
Postdoctoral Fellow

Raymond.zhao@data61.csiro.au