Radboud University

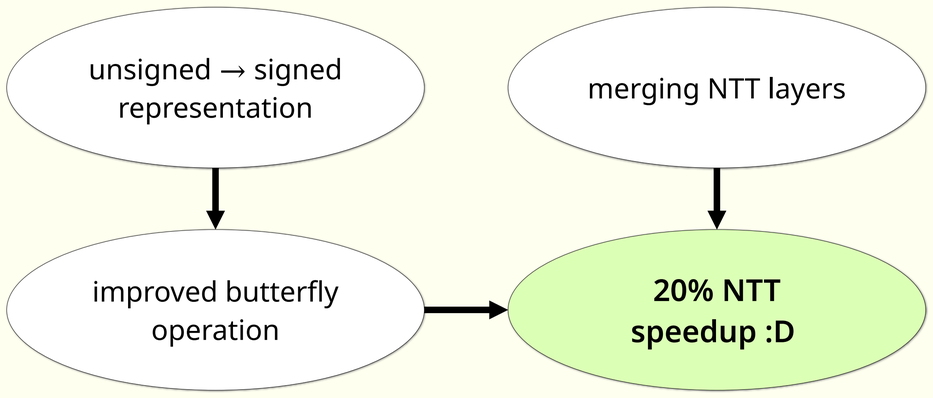
Compact Dilithium Implementations on Cortex-M3 and Cortex-M4

https://ia.cr/2020/1278

Denisa O. C. Greconici, Matthias J. Kannwischer, Amber Sprenkels

denisa.greconici@gmail.com, matthias@kannwischer.eu, amber@electricdusk.com

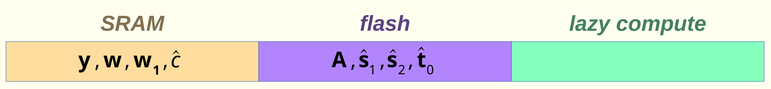
Performance



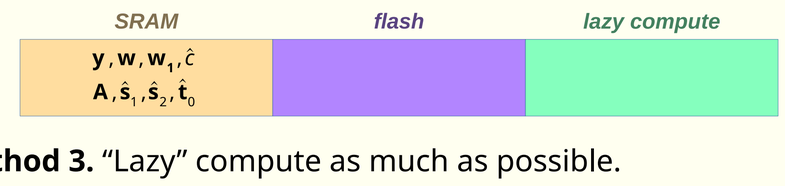
Memory

Some Dilithium values need a lot of storage. We investigate how their storage location impacts the memory usage and speed.

Method 1. Store the public key in flash memory.



Method 2. Everything in SRAM.



Method 3. "Lazy" compute as much as possible.

Cortex-M3 challenges

- `smull` & `smlal` not constant time!
- have to use Schoolbook (SB) multiplication on Cortex-M3
- only needed for computing with *secrets*

9 cycles

SBSMULL

```
; Input:  a = a0 + a1*2^16
;         b = b0 + b1*2^16
;         c = c0 + c1*2^32
; Output: c = c + a*b
;           = c0 + c1*2^32
mul  tmp, a0, b0
adds c0, c0, tmp
mul  tmp, a1, b1
adc  c1, c1, tmp
mul  tmp, a1, b0
mla  tmp, a0, b1, tmp
adds c0, c0, tmp, lsl #16
adc  c1, c1, tmp, asr #16
```

7 cycles

SBSMLAL

```
; Input: a = a0 + a1*2^16
;        b = b0 + b1*2^16
; Output: c = a*b = c0 + c1*2^32
mul c0, a0, b0
mul c1, a1, b1
mul tmp, a1, b0
mla tmp, a0, b1, tmp
adds c0, c0, tmp, lsl #16
adc c1, c1, tmp, asr #16
```

Results

		Cortex-M4		Cortex-M3	
Algorithm & method	**Work**	**Speed [kcc]**	**Stack [B]**	**Speed [kcc]**	**Stack [B]**
Keygen (1)	*This work*	3 545	15 916[ab]	4 503	15 703
Keygen (2 & 3)	[GKOS18]	2 320	50 488	—	—
	This work	2 013	8 940	2 562	9 007
Sign (1)	[RGCB19, scen. 2]	5 495	—	—	—
	This work	4 578	17 628[b]	8 730	18 083
Sign (2)	[GKOS18]	8 348	86 568	—	—
	[RGCB19, scen. 1]	7 085	—	—	—
	This work	6 053	52 756	10 667	53 959
Sign (3)	*This work*	23 550	9 948	33 229	10 495
Verify	[GKOS18]	2 342	54 800	—	—
	This work	1 917	10 028	1 917	10 028

[a] Updated value was incorrect in the original paper.
[b] Uses an additional 34 896 bytes of flash space.

Platforms & benchmarking

- Dilithium version is NIST round-2
- Results describe security level "Dilithium3" (more in the paper)
- Cortex-M4 platform: STM32F407 Discovery board @ 24 MHz
- Cortex-M3 platform: Arduino Due (SAM3X8E) @ 16 MHz

Takeaways

- 20% faster NTT on Cortex-M4
- 7% faster keygen
- 15% faster signing
- 9% faster verification
- New Cortex-M3 implementation :D

Scheme comparison (updated to 2022)

Algorithm	Public key [B]	Signature [B]	Sign [kcc]	Verify [kcc]
Falcon-1024 [PQM4]	1 793	1 280	85 161	978
SPHINCS+ (shake256-192s-robust) [PQM4]	48	16 224	93 294 080	83 021
Dilithium3 [PQM4,AHKS22]	1 952	3 293	6 610	2 691
Dilithium3 [BRS22]			36 303	7 249

Acknowledgements

- This work has been supported by the European Commission through the ERC Starting Grant 805031 (EPOQUE). <3
- @Bo-Yin Yang: Thank you for all your valuable comments and the Institute of Information Science at Academia Sinica in Taipei, Taiwan, for housing and supporting MJK during the COVID-19 pandemic. This paper would not have been written otherwise.
- @Peter Pessl: Thank you for pointing out various improvements to our stack optimizations.

References

[GKOS18]: Tim Güneysu, Markus Krausz, Tobias Oder, and Julian Speith. *Evaluation of lattice-based signature schemes in embedded systems.* ICECS 2018

[RGCB19]: Prasanna Ravi, Sourav Sen Gupta, Anupam Chattopadhyay, and Shivam Bhasin. *Improving Speed of Dilithium's Signing Procedure.* CARDIS 2019, https://ia.cr/2019/420.pdf

[LDK+19]: Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, and Damien Stehlé. *CRYSTALS-DILITHIUM.* Submission to the NIST Post-Quantum Cryptography Standardization Project, 2019. https://pq-crystals.org/dilithium

[AHKS22]: Amin Abdulrahman, Vincent Hwang, Matthias J. Kannwischer, Amber Sprenkels. *Faster Kyber and Dilithium on the Cortex-M4.* ACNS 2022, https://ia.cr/2022/112 (recent work)

[BRS22]: Joppe W. Bos, Joost Renes, Amber Sprenkels. Dilithium for Memory Constrained Devices. AfricaCrypt 2022, https://ia.cr/2022/323 (recent work)

[PQM4]: Matthias J. Kannwischer and Joost Rijneveld and Peter Schwabe and Ko Stoffelen. PQM4: Post-quantum crypto library for the ARM Cortex-M4, https://github.com/mupq/pqm4