Lattice Signatures

Carl and Dustin

FALCON

Fast-Fourier Lattice-based Compact Signatures over NTRU

Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, Zhenfei Zhang

Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé

Overview

- (qTESLA broken, only parameter set left isn't competitive)

- CRYSTALS-Dilithium
  - Fiat-Shamir with aborts
  - Simpler: uniform sampling
  - Compact and efficient

- Falcon
  - Hash and sign over NTRU lattices
  - More complicated: floating point ops, gaussian sampling
  - Efficient and more compact

Round 2 changes

- Dilithium
  - Included option to be non-deterministic (a one line change in code)
  - Optimized their implementation more
  - Added an AES option, instead of SHAKE (to show potential speedup of having hardware instructions)

- Falcon
  - Removed their level 3 parameter set (simplifies their spec considerably)
  - (Sort of) added a key-recovery option
  - In Aug/Sep, they provided a constant-time implementation

How they work

- Design goal of both was to minimize |PK|+|sig size|

Dilithium Sketch

A:=XOF(ρ), **t**:=**A**s_1+s_2
Public key: ρ,t_1

Sign(μ)

y ← D with uniform small coefficients
c := H(HighBits(**A**y), μ)
z := y + cs_1
RejectionSample(z, cs_1, cs_2)
(Must hold: HighBits(**A**y)=HighBits(**A**z-c**t**))
Create a hint **h** such that
HighBits(**A**z-c$\mathbf{t}_1$) & **h** → HighBits(**A**z-c**t**)
Signature = (z, **h**, c)

Verify((z, h, c), μ)

Use **A**z-c$\mathbf{t}_1$ and **h** to get
w:= HighBits(**A**z-c**t**)

Check that z has small coefficients
and
c=H(**w**,μ)

Falcon in a Nutshell

PQSHIELD

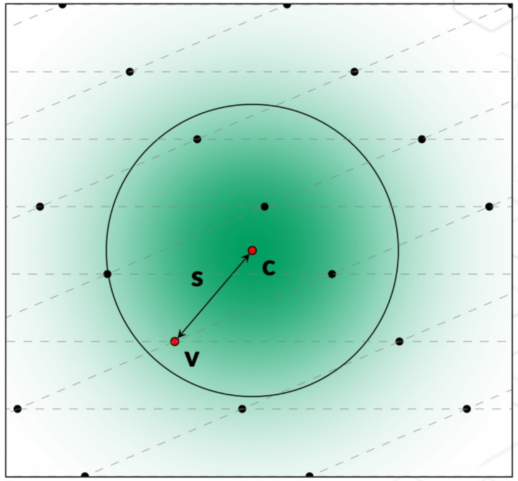
We work over the cyclotomic ring $\mathcal{R} = \mathbb{Z}_q[x]/(x^n + 1)$.

→ **Keygen()**
1. Gen. matrices **A**, **B** with coefficients in $\mathcal{R}$ such that:
   - $\mathbf{BA} = 0$
   - **B** has small coefficients
2. pk ← **A**
3. sk ← **B**

→ **Sign(m, sk)**
1. Compute **c** such that $\mathbf{cA} = H(m)$
2. **v** ← "a vector in the lattice Λ(**B**), close to **c**"
3. $\mathbf{s} \leftarrow \mathbf{c} - \mathbf{v}$

The signature sig is $\mathbf{s} = (s_1, s_2)$

⇒



→ **Verify(m, pk, sig)**
Accept iff:
1. **s** is short
2. $\mathbf{sA} = H(m)$

Parameter Sets

	Sec Level	PK size	SK size	Signature size
Dilithium	1	1184	2800	2044
Dilithium	2	1472	3504	2701
Dilithium	3	1760	3856	3366
Falcon	1	897	1281	617
Falcon	4/5	1793	2305	1233
Falcon (message recovery)	5	1793		768
Falcon (key recovery)	5	64		2506

Performance

The Basics:

- Falcon and Dilithium tend to lead the pack in **speed of signing & verifying**. (Falcon may have a slight edge there.)
- Falcon’s **key generation** is slower; Dilithium’s is fast.
- They are both pretty good in terms of key/signature size.

Speed comparison (from John's presentation)

Performance on Intel/AMD Desktop Machines*

scheme	keygen	sign	verify	sign+verify
dilithium2	3.1	11.5	1.0	3.4
falcon512dyn	363.9	15.0	0.5	3.8
falcon512tree	362.3	8.5	0.4	2.2
gemss128	5831.4	51984.3	20.1	11862.0
bluegemss128	6684.5	10282.8	37.5	2372.3
redgemss128	4842.4	223.4	35.4	78.3
picnicl1fs	0.2	133.5	32.4	55.4
rainbow1a	19678.3	4.4	0.7	1.6
SPHINCS128-f	57.6	1744.1	29.8	420.4
SHPINCS128-s	1843.4	27387.5	12.4	6250.8

* Averaged from 37 machines

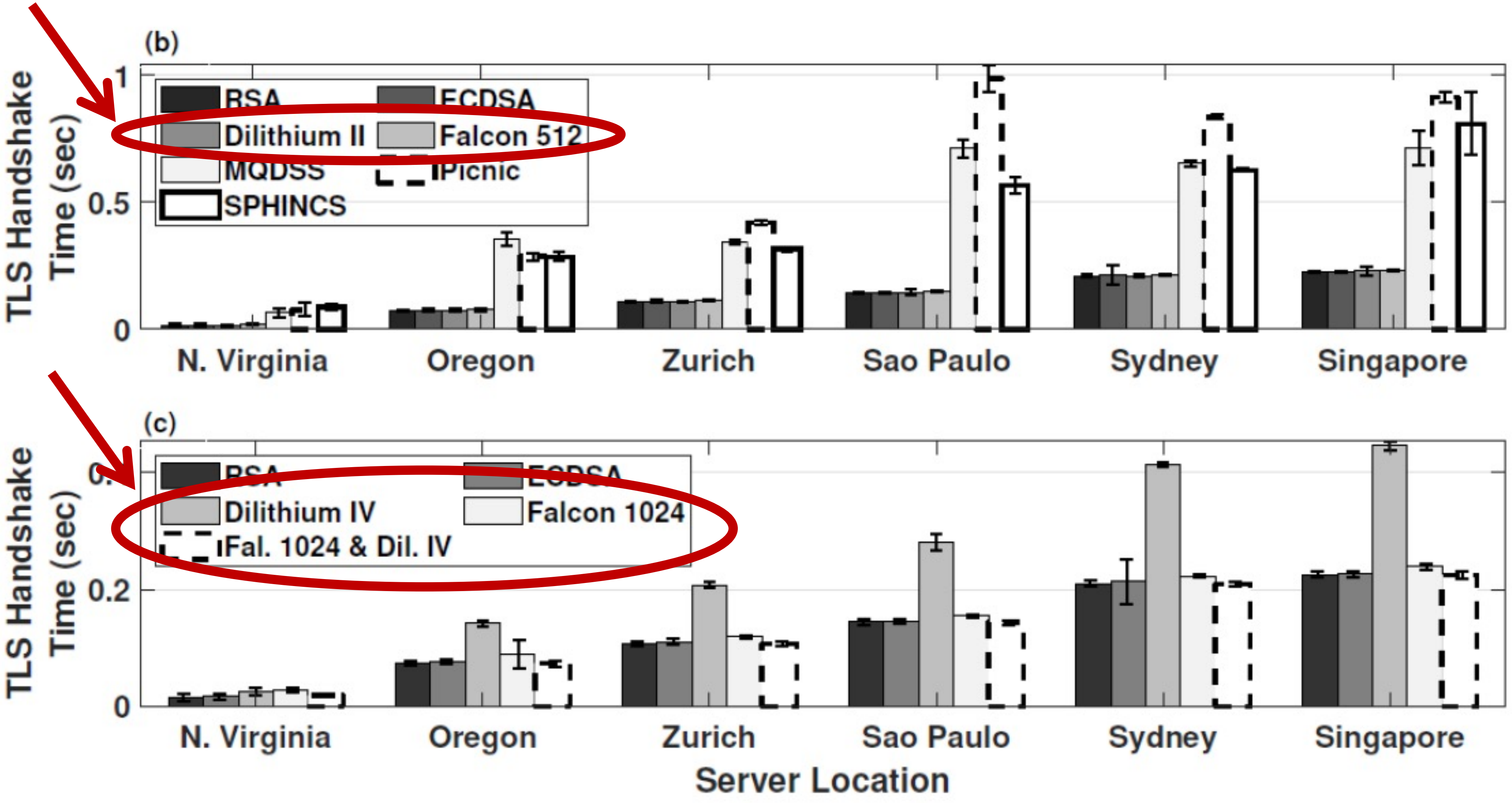
Numbers indicate how many times slower than EdDSA 25519

Size comparison (from John's presentation)

scheme	sk size	pk size	sig size	pk + sig
ed25519	64	32	64	96
rsa3072	384	384	384	384
dilithium2	2800	1184	2044	3228
falcon512dyn*	1281	897	659	1556
gemss128*	14520	417408	33	417441
picnic2l1fs	49	33	12306	12339
rainbow1a	100209	152097	64	152161
Sphincsf128**	64	32	16976	17008

(numbers are in bytes)

“Post Quantum Authentication in TLS 1.3” (from Angela’s presentation)



Hardware Implementations?

NIST Post-Quantum Cryptography- A Hardware Evaluation Study

Kanad Basu, Deepraj Soni, Mohammed Nabeel, and Ramesh Karri

https://eprint.iacr.org/2019/047.pdf

A High-Level Synthesis Approach to the Software/Hardware Codesign of NTT-based Post-Quantum Cryptography Algorithms

Duc Tri Nguyen, Viet B. Dang and Kris Gaj
Department of Electrical and Computer Engineering, George Mason University, Fairfax, VA, U.S.A.
{dnguye69, vdang6, kgaj}@gmu.edu

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8977896

"This is the first hardware benchmarking and uses a common evaluation framework to study area vs performance vs security trade-offs."

Unfortunately, it covers Dilithium but not Falcon.

TABLE II
Results of the implementations of the NTT unit for selected Round 2 PQC Candidates, using Zynq UltraScale+.

Algorithms	N	q		DSPs	BRAM 36K	LUT	FF	Slices	Max Freq (MHz)	Clock Cycles	Latency (μs)
NewHope & FALCON	1024	12,289	RTL	4	5	849	802	163	476	1,324	2.78
			HLS	4	5	865	822	175	455	1,324	2.91
			HLS/RTL	1.0	1.0	1.02	1.02	1.07	0.96	1.0	1.05
qTESLA	1024	8,404,993	RTL	8	8	1,286	2,160	283	467	1,363	2.92
			HLS	8	8	1,939	3,423	453	455	1,363	2.99
			HLS/RTL	1.0	1.0	1.51	1.58	1.60	0.97	1.0	1.03
CRYSTALS-DILITHIUM	256	8,380,417	RTL	8	2	1,899	2,041	392	445	294	0.66
			HLS	8	2	1,977	2,329	401	434	294	0.67
			HLS/RTL	1.0	1.0	1.04	1.14	1.02	0.97	1.0	1.02

Security

Both are based on lattice problems over $\boldsymbol{Z_q}$.

LWE

LWE = distinguish noisy samples of a linear transformation from truly random samples

Security

Both are based on lattice problems over $\boldsymbol{Z_q}$.

LWE

SIS

SIS = Find a short vector in the null space of a linear transformation

Security

Both are based on lattice problems over $\boldsymbol{Z_q}$.

LWE

MLWE

SIS

MLWE = LWE when the linear transformation is a matrix of polynomials *(from* $\boldsymbol{Z_q[X]/(X^n+1)}$*, in this case)*

Security

Both are based on lattice problems over $\boldsymbol{Z_q}$.

LWE	MLWE
SIS	MSIS

MSIS = SIS when the linear transformation is a matrix of polynomials *(from* $\boldsymbol{Z_q[X]/(X^n+1)}$*, in this case)*

Security

Both are based on lattice problems over $\boldsymbol{Z}_{\boldsymbol{q}}$.

LWE

MLWE

SIS

MSIS

NTRU-SIS

NTRU-SIS = SIS with a different polynomial structure

Security

Dilithium is based on:

- The (Q)ROM model
- MLWE
- MSIS *(with different parameters, presumably)*

(This relationship is complicated in the case of a quantum adversary.)

Falcon is based on (?):

- The (Q)ROM model
- NTRU-SIS
- *Assumptions about floating-point arithmetic?*

Dilthium's discussion of security is a good deal more detailed than Falcon's.

Attacks

According to their specs, the best known <u>theoretical</u> attacks on Falcon & Dilithium are simply lattice reduction attacks.

After an online search, we didn’t find anything to suggest otherwise.

Side-Channel Attacks

There are lots of papers about side-channel attacks.

BEARZ Attack FALCON: Implementation Attacks with Countermeasures on the FALCON signature scheme

https://eprint.iacr.org/2019/478.pdf

Side-channel Assisted Existential Forgery Attack on Dilithium - A NIST PQC candidate

https://eprint.iacr.org/2018/821.pdf

Differential Fault Attacks on Deterministic Lattice Signatures

https://tches.iacr.org/index.php/TCHES/article/view/7267

Masking Dilithium

https://link.springer.com/chapter/10.1007/978-3-030-21568-2_17

Side-Channel Attacks

Constant-time implementation issues are discussed (at least briefly) in both Falcon and Dilithium's specs.

New Efficient, Constant-Time Implementations of Falcon

Thomas Pornin

https://falcon-sign.info/falcon-impl-20190918.pdf

- This is likely more of an issue for FALCON, with its complex floating point implementation

IP status

- Dilithium
  - No patents listed on their signed statements
  - Jacob noticed they include a “hint” that could conceivably be taken by Ding to be reconciliation. Hint allows big savings on public key
- Falcon
  - Patent from 2001 listed on their signed statements. It’s from NTRU creators. Expires in 2025. They checked the box “without compensation”
  - “A method, system and apparatus for performing user identification, digital signatures and other secure communication functions in which keys are chosen essentially at random from a large set of vectors and key lengths are comparable to the key lengths in other common identification and digital signature schemes at comparable security levels. The signing technique of an embodiment of the identification/digital signature scheme hereof uses a mixing system based on multiplication in a ring and reduction modulo an ideal q in that ring; while the verification technique uses special properties of products of elements whose validity depends on elementary probability theory. The security of the identification/digital signature scheme comes from the interaction of reduction modulo q and the difficulty of forming products with special properties. “

Documentation, Simplicity, Flexibility….

- Dilithium
  - Uniform sampling
  - Same *q* and same ring for all parameter sets $Z_q[x]/(x^{256}+1)$
  - Mainly need 2 operations: SHAKE and polynomial multiplications in ring
  - Documentation is pretty good
  - They say there is a ZK-privacy primitive that can be built from Dilithium
  - Uses NTT
- Falcon
  - Bimodal Gaussian sampling
  - Documentation clear, but more complicated
  - Same *q* and ring for all parameter sets
  - Uses NTT
  - When combined with New Hope, there is a practical IBE scheme
  - Modular – can switch lattice type or trapdoor sampler
  - Message/key recovery options

Round 2 happenings

- Official comments and forum discussion:
  - Yunlei Zhao (of KCL) reminded everybody that he has a smaller signature scheme than Dilithium, which is basically the same
  - Markku notes that both Dilithium and Falcon are well suited for constrained environments among all the Round 2 signatures
  - A TLS experiment by Cisco concluded Dilithium and Falcon are the best options among the Round 2 signatures
  - Panos noted that both Dilithium and Falcon have classical security numbers that are much lower than 128 bits. Stehle responds that these numbers are from loose lower bounds, and don’t factor in practical real world costs, which would put them at the right security level.

More Round 2 happenings

- Research results announced
  - A paper on Fiat-Shamir in the QROM model was published, which applies to Dilithium, Picnic and MQDSS
  - Falcon presented their work at our 2nd workshop on a constant time implementation. A few weeks later they gave an update to correct secret leakages and fixed the reported performance numbers
  - For Dilithium, an outside group published a few papers on fault attacks, optimizing the implementation, and masking
  - For Falcon, some side-channel analysis work posted on eprint.iacr, as well as a paper on hardware implementation of Gaussian sampling

Advantages and Disadvantages

- Dilithium
  + Uniform distribution
  + Compact
  + Efficient
  + Simple to implement
  + Strong team
  + Shares framework w/ Kyber
  + Security is conservative?
  + Easy to scale
  - No level 5
  - Needs more side-channel work
  - (IP maybe a concern?)

- Falcon
  ++ Compact
  + Efficient
  + Strong team
  + Tight ROM and QROM proofs
  - Slow KeyGen
  - Gaussian distribution
  - No level 3
  - Needs more side-channel work
  -- Floating point complexity

Summary

- Ask Dilithium for level 5
  - Falcon doesn't have level 3. They used to, but they dropped it for Round 2

- Both seem to be pretty good options

- We recommend both advance on