THE NIST PQC PROCESS

UNOFFICIAL ABRIDGED VERSION

Sofía Celi and Thom Wiggers

Preface

- This is definitely not a complete overview
- We mainly just summarize the NIST report
- Some bias towards what we're familiar with
- We're not pitching a draft here or any path to go
- We want to help start the discussion in working groups and to start thinking about experimental designs

See NISTs report: https://csrc.nist.gov/publications/detail/nistir/8413/final

First PQC standards

Key Exchange

- CRYSTALS-Kyber

Signatures

- CRYSTALS-Dilithium
- FALCON
- SPHINCS+

https://twitter.com/cronokirby/status/1544386085836431360

Key Encapsulation Mechanisms

- See also HPKE (RFC9180)
- Kyber to be the first NIST KEM
  - Lattice based
  - See other talk today

KEMs are not Diffie-Hellman equivalent!

- KEMs have been used in TLS experiments
- but KEMs do not work in other protocols e.g. in Signal's X3DH

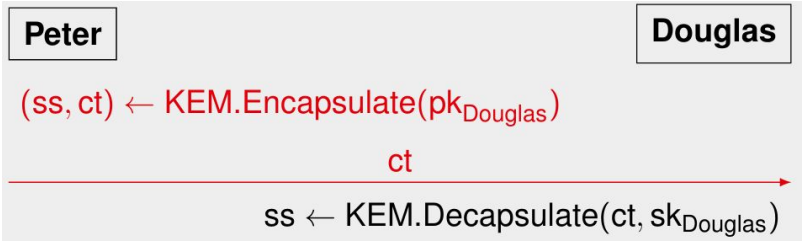

Fig: KEM key exchange is *interactive,* so Peter can't combine his secret key with Douglas' public key to authenticate himself (as you could do with DH)

Signature schemes

Dilithium:

- General purpose
- Very fast in software
- Quite large

“Dilithium … is, thus, the **primary signature algorithm** selected by NIST for standardization at this time

Scheme	public key bytes	signature bytes
Dilithium-II	1,312	2,420
Dilithium-III	1,952	3,293
Dilithium-V	2,592	4,595

Signature schemes

FALCON:

- Smaller than Dilithium
- Very hard to implement correctly
- Signing requires *constant time* 64-bit FPU operations

“FALCON was chosen for standardization because NIST has confidence in its security (**under the assumption that it is correctly implemented**) and because its small bandwidth may be necessary in certain applications.”

Scheme	public key bytes	signature bytes
Falcon-512	897	666
Falcon-1024	1,793	1,280

Signature schemes

SPHINCS+

- Hash-based signatures
- No XMSS/LMS state footgun
- Hundreds of hash function calls
  - Pretty slow
- There are currently **36** parameter sets

SPHINCS+ was selected for standardization because it … is based on an **entirely different set of assumptions** than those of our other signature schemes to be standardized

Variant	pk bytes	signature bytes
128-small	32	7,856
128-fast	32	17,088
192-small	48	16,224
192-fast	48	35,664
256-small	64	29,792
256-fast	64	49,856

Signature schemes: current IETF standards

XMSS (RFC8391) and LMS (RFC8554) are already standardized at IETF

These are **stateful** hash based signature schemes

- Not suitable if you e.g. like to:
  - backup your secret key
  - copy your secret key to multiple machines
  - copy a virtual machine image
  - can’t manage the state incredibly reliably
- But significantly smaller than SPHINCS+

Still in the race

The following KEMs go to round 4 and may still be standardized

- Classic McEliece
  - conservative, ancient
  - code-based
  - very big public keys but tiny ciphertexts
  - talk to NIST if you like it
- SIKE
  - Isogenies
  - Very small
  - Quite slow
- BIKE
  - Code-based
- HQC
  - Code-based
  - Larger keys than BIKE
  - Faster runtime than BIKE

NISTs second call for proposals

- Only signature schemes
  - “NIST … would like … **short signatures** and **fast verification** (e.g., UOV).”
- We may see:
  - UOV: 400 kb+ public keys (66kb compressed) but very small signatures
  - MAYO: small, but very new?
  - New code-based signatures?
  - New isogeny signatures? (very new)
- Don’t expect NIST standards **before 2030:**
  - “will similarly take several years”

Running code

Experiment with PQ using:

- liboqs (C, bindings in C++, Java, Rust, Python, Go, .NET available)

  Open-Quantum-Safe have experimental PQ forks of OpenSSH and OpenSSL.

- PQClean (cleaned up and usable reference code)
- PQM4 (embedded applications)

https://commons.wikimedia.org/wiki/File:Chemistry_Experiment_3D.JPG

Questions to ask today

- Does your protocol's key exchange fit the not-DH-like KEM API?
- Does your protocol need DH-like properties?
- Do the new public key, ciphertext and signature sizes fit in your protocol?
- Do the new schemes fit on your target platform?

Selection of notable and fun further reading

- NISTs report on round 3 https://csrc.nist.gov/publications/detail/nistir/8413/final
- PQ Key exchange over the internet https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html and https://blog.cloudflare.com/the-tls-post-quantum-experiment/
- Fitting PQ signatures into TLS https://blog.cloudflare.com/sizing-up-post-quantum-signatures/
- Kyber for a general audience (talk) https://media.ccc.de/v/rc3-2021-cwtv-230-kyber-and-post-quantum
- PQM4 benchmarks https://github.com/mupq/pqm4/blob/master/benchmarks.md
- Dilithium on low-stack devices https://eprint.iacr.org/2022/323
- Falcon on Cortex-M7 and funkyness in FPUs https://eprint.iacr.org/2022/405
- Breaking Rainbow Takes a Weekend on a Laptop https://eprint.iacr.org/2022/214
- PQ Authentication in TLS: benchmarks https://eprint.iacr.org/2020/071

THANK YOU!

@claucece
@thomwiggers