OpenSSH PQC past, present, future

RWPQC 2024

Damien Miller

djm@openssh.com

@damienmiller.bsky.social (BlueSky)

@djm@cybervillains.com (Fedi)

OpenSSH overview

- Project started in September 1999
- Persistently the most popular server implementation
- Responsible for some of the SSH protocol modernisation over the last 20 years
- First (only?) SSH implementation to offer PQC key agreement by default

SSH protocol

- SSHv2 is a 1990s crypto protocol
  - Initially Encrypt-and-MAC (predates AEAD)
  - Classical key agreement, initially modp-DH
  - ECDH, ECDSA came later
- Flexible vendor extension mechanism
  - Many new algorithms added since initial RFCs: ciphers, MACs, AEADs, signature schemes, key agreement
  - Allowed deeper retrofit of protocol over time, e.g. Encrypt-then-MAC, AEADs
  - Recently, PQC key agreement

History of PQC in SSH

- **2018/03** OpenSSH adds experimental support for XMSS signatures. Disabled by default.
- **2018/12** TinySSH added support for hybrid Streamlined NTRU Prime / X25519 KEM `sntrup4591761x25519-sha512`
- **2019/01** OpenSSH added interoperable implementation labeled as experimental
- **2020/12** OpenSSH replaces implementation with `sntrup761x25519-sha512`
- **2021/11** OpenSSH includes `sntrup761x25519-sha512` in the default client/server algorithms proposal
- **2022/02** OpenSSH promotes this algorithm the highest-preference position

Motivation, etc

- XMSS was implemented as an experiment to learn how a stateful signature scheme could be added to SSH
  - Never enabled, will likely be removed when we add support for one of the NIST finalists.
- PQC KEM was added for fear of "store now, decrypt later attacks"
- Guiding assumptions:
  - Cryptographically-relevant QC available in mid-2030s ("quantum computing is always 15 years away")
  - LTS operating system lifecycle is ~5 years
  - Want to be ready 10+ years early.

Algorithm selection “Why Streamlined NTRU Prime?”

- NIST PQC selection process had not concluded
  - Yet worried about possibility of early QC
- TinySSH gave us an **implementation to interoperate against**
- Code-based or Lattices seemed the **conservative choices**
- Code-based had practical key-size problems, therefore Lattices
- **Will only consider hybrid** PQC/classical KEMs
- *(I am not a lawyer!)* **Seemed safe wrt IPR**. Patents either granted or expired
- NTRU Prime has a **good-quality, portable reference implementation**

Where next

- Want to implement NIST finalist KEM ASAP
  - Waiting for someone to tell us ML-KEM to is ready
- Have had requests to implement a code-based system too as a hedge against new cryptanalytic results
- Wary of adding everyone's favourite algorithm
  - It takes at least five years to remove anything.
  - PQC code is complex attack surface, scared of implementation bugs
- Desire to implement a PQC signature scheme becoming more urgent - CA keys may have 10+ year lifespans.

Future challenges

- OpenSSH is mostly used on workstations, laptops and servers. It does not target the (important) embedded use-case, e.g. network infrastructure
- PQC for embedded systems is more challenging
  - Increased CPU, longer messages
  - Updating devices often requires a dumpster
  - Can’t afford to speculate on algorithm choice
- PQC hardware support
  - Aforementioned embedded case
  - Smartcards, tokens, secure elements, HSMs
  - Support up the stack (PKCS#11, FIDO)
  - Urgent!

Final thoughts

- The SSH protocol is an nice place to experiment
  - Clear extension mechanism
  - Hierarchy of optionality: compile-time / optional / enabled / best-preference
  - Relatively forgiving performance requirements (versus, say, TLS)
  - Nice to see projects like OQS-OpenSSH take advantage of this
- State of PQC feels like cryptography in the late 1990s: abundance of algorithms, different tradeoffs, cryptanalysis moving quickly, hard for application authors to decide
- Looks like we'll need algorithm agility for a while longer 🙃

Thanks