

wolfSSL

Post-Quantum Cryptography (PQC) Update



Anthony Hu
Senior Software Developer

Tokyo, Japan
November 2024

Session Agenda

1. **NIST PQC Standards and CNSA 2.0**
2. **wolfSSL PQC History and Current Status**
3. **NIKE vs KEM**
4. **Benchmarking ECC vs ML-DSA and ML-KEM**
5. **wolfSSL PQC Readiness and Migration Efforts**
6. **What Others are Doing Together with wolfSSL**
7. **Barriers to Migration**



Company Overview

Medical Devices

Automotive / ADAS / ECUs

Industrial Systems

Internet of Things

Aviation

Satellite

Appliances

Battlefield Communication

VoIP

Electric Grid

Federal Governments

Games

We Secure the **Internet** by **Securing Data**

Sensors Intelligent Systems

Routers

Smart Energy

Databases

Mobile Phones

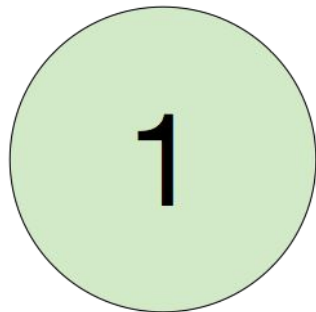
Cloud Services

Connected Home

Applications

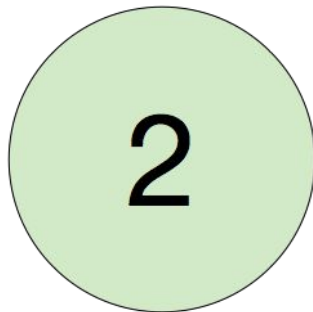
Printers

Data at Rest



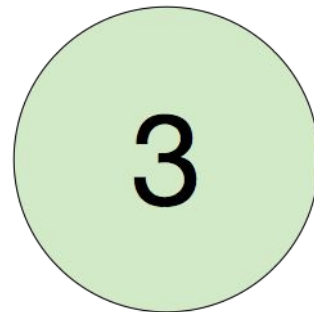
- Secured with **Cryptography**

Data in Transit



- Secured with **SSL/TLS, SSH**
- Transfer Mediums:
TCP/UDP/Bluetooth/Serial/CA
N-BUS/ARINC, etc

Firmware Updates



- Secured with **SSL/TLS, crypto, MQTT, secure boot**
- Prevent malicious firmware flashing and updates

wolfSSL Technology Partners

arm

intel®

NXP

MARVELL™

TEXAS
INSTRUMENTS

ST
life.augmented

SAMSUNG

WINDRVR

SIEMENS

Green Hills®
SOFTWARE

IAR
SYSTEMS

RENESAS
PARTNER Platinum

infineon

ANALOG
DEVICES
AHEAD OF WHAT'S POSSIBLE™

MICROCHIP

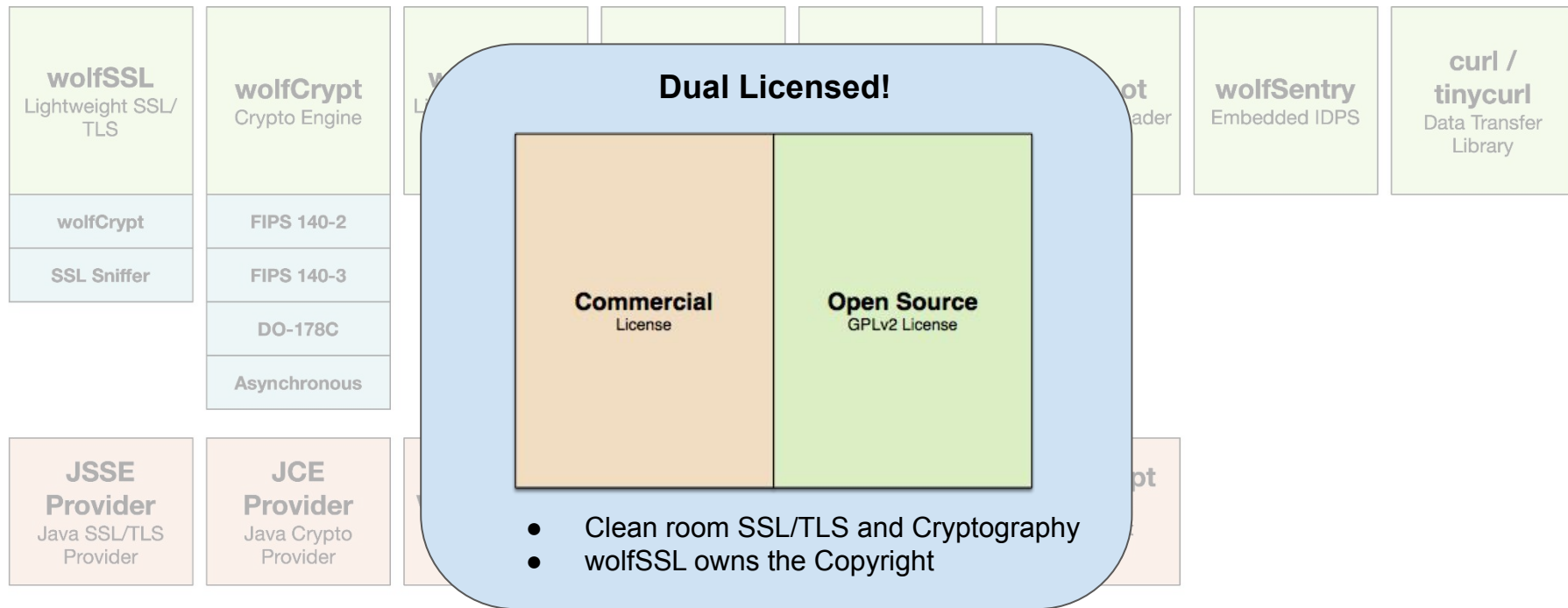
Atmel®

XILINX

wolfSSL Open Source Partners



wolfSSL Products



Conventional Algorithms Vs. Post-Quantum Algorithms

- **Conventional Algorithms**

- Vulnerable to quantum computers.
- In the presence of a sufficiently powerful quantum computer, the **private key** can be derived from the **public key**. For symmetric, the **secret key** brute force search can be reduced by **half**.
- Examples include ECC, RSA, DH, AES

- **Post-Quantum Cryptography (PQC) Algorithms**

- No known attacks, even in the presence of a quantum computer
- Generally newer algorithms using new math or hashes (Lattice based cryptography, Isogenies)
- Examples include ML-KEM, ML-DSA, SLH-DSA, LMS, XMSS

Post-Quantum Readiness

- **Immediate Concerns**

- Data Harvesting (Harvest Now, Decrypt Later)

- Encrypted data **harvested now** by malicious actors, then **decrypted later** when quantum computers are available. Is this within the time you need it to stay confidential?

- Long-Lived Devices (Deploy and Forget)

- Devices being **deployed** to the field and then **forgotten** will be susceptible to attack later when quantum computers are available since they have not been updated.

- **Mitigations and Migration**

- **Use Hybrid Signature schemes** (ex: ECDSA + ML-DSA via dual algorithm certificates)

- **Use Hybrid Key Establishment** (ex: ECDHE + ML-KEM)

- **Use Larger symmetric key size** (256-bit cipher)

- **Bonus: Stay FIPS 140-3 compliant** (NIST Certificate #4718; sunsets in 5 years in 2029. Most others are expiring in 2 years only.)

IETF / NIST

Stateful Hash-Based Signatures

Stateful Hash-Based Signatures

- 2016 NIST Post-Quantum Standardization for Public-Key algorithms required signature algorithm submissions to be “**stateless**”
 - “**stateful**” signature algorithms did not meet the API requirements, standardization was **separate from 2016 competition**, coordinated along with **IETF**
 - “**Stateful**” means that the private key changes each time it is used.
- **Stateful Hash-Based Signatures:**
 - Not vulnerable to quantum computers
 - Well studied and very old
 - Better performance than stateless algorithms for sign/verify, but **very slow keygen**
 - **Require careful state management; misuse is easy and catastrophic; all signatures must be revoked**
 - Appropriate for applications where **private key resides in an HSM and private key operations are offline (i.e.: firmware signing)**
 - Gave a head start to digital signature scheme PQC migration
 - Will be mentioned when we talk about FIPS-205

Stateful Hash-Based Signatures

- **IETF** standardized both of the following Stateful Hash-Based Signature algorithms:
 - **XMSS** (RFC 8391, 2018) - eXtended Merkle Signature Scheme
 - **LMS** (RFC 8554, 2019) - Leighton-Micali Hash-Based Signatures

INTERNET RESEARCH TASK FORCE (IRTF) Request for Comments: 8391 Category: Informational ISSN: 2070-1721	INFORMATIONAL Errata Exist A. Huelsing TU Eindhoven D. Butin TU Darmstadt S. Gazdag genua GmbH J. Rijneveld Radboud University A. Mohaisen University of Central Florida May 2018
-----------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

XMSS: eXtended Merkle Signature Scheme

Abstract

This note describes the eXtended Merkle Signature Scheme (XMSS), a hash-based digital signature system that is based on existing descriptions in scientific literature. This note specifies Winternitz One-Time Signature Plus (WOTS+), a one-time signature scheme; XMSS, a single-tree scheme; and XMSS^{MT}, a multi-tree variant of XMSS. Both XMSS and XMSS^{MT} use WOTS+ as a main building block. XMSS provides cryptographic digital signatures without relying on the conjectured hardness of mathematical problems. Instead, it is proven that it only relies on the properties of cryptographic hash functions. XMSS provides strong security guarantees and is even secure when the collision resistance of the underlying hash function is broken. It is suitable for compact implementations, is relatively simple to implement, and naturally resists side-channel attacks. Unlike most other signature systems, hash-based signatures can so far withstand known attacks using quantum computers.

INTERNET RESEARCH TASK FORCE (IRTF) Request for Comments: 8554 Category: Informational ISSN: 2070-1721	INFORMATIONAL Errata Exist D. McGrew M. Curcio S. Fluhrer Cisco Systems April 2019
-----------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------

Leighton-Micali Hash-Based Signatures

Abstract

This note describes a digital-signature system based on cryptographic hash functions, following the seminal work in this area of Lamport, Diffie, Winternitz, and Merkle, as adapted by Leighton and Micali in 1995. It specifies a one-time signature scheme and a general signature scheme. These systems provide asymmetric authentication without using large integer mathematics and can achieve a high security level. They are suitable for compact implementations, are relatively simple to implement, and are naturally resistant to side-channel attacks. Unlike many other signature systems, hash-based signatures would still be secure even if it proves feasible for an attacker to build a quantum computer.

This document is a product of the Crypto Forum Research Group (CFRG) in the IRTF. This has been reviewed by many researchers, both in the research group and outside of it. The Acknowledgements section lists many of them.

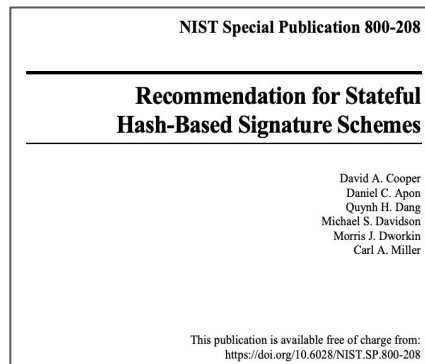
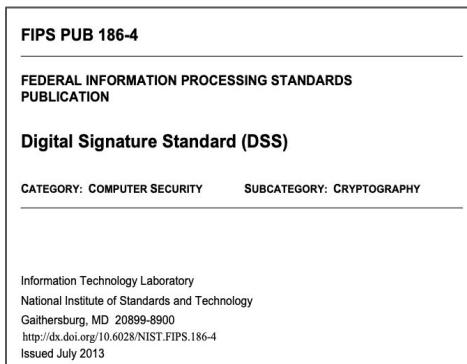
Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Research Task Force

Stateful Hash-Based Signatures

- **NIST SP 800-208 - “Recommendation for Stateful Hash-Based Signature Schemes”**
 - Supplements **FIPS 186** by approving two stateless hash-based schemes
 - Profiles **LMS**, **XMSS**, and their multi-tree variants:
 - Hierarchical Signature Scheme (**HSS**)
 - Multi-tree XMSS (**XMSS^{MT}**)
 - **Approves some, but not all** parameter sets from **RFC 8391 / 8554**
 - SHA-256 or SHAKE256 with 192 or 256-bit output



NIST

PQC FIPS Standards

NIST Post-Quantum Cryptography Levels

- **NIST Post Quantum Security Levels**
 - **Level 1** - Equivalent to **AES 128-bit** block cipher key search
 - **Level 2** - Equivalent to **SHA2-256-bit** hash collision search
 - **Level 3** - Equivalent to **AES 192-bit** block cipher key search
 - **Level 4** - Equivalent to **SHA2-384-bit** hash collision search
 - **Level 5** - Equivalent to **AES 256-bit** block cipher key search

NIST Post-Quantum Cryptography Standardization

- **FIPS 203**

- Specifies **ML-KEM**
- Based on **CRYSTALS-KYBER**
- Defines 3 parameter sets (ECDH sizes are in parenthesis for comparison)

Variant	Encapsulation Key	Decapsulation Key	Ciphertext	Shared Secret Key	AES Equivalence
ML-KEM-512	800 (64)	1632 (32)	768 (64)	32	128
ML-KEM-768	1184 (96)	2400 (48)	1088 (96)	32	192
ML-KEM-1024	1568 (131)	3168 (66)	1568 (131)	32	256

- Appropriate for replacement of quantum-vulnerable **key exchange algorithms** (ex: ECDH, FFDH)
- **Performance is very good**, cryptographic **artifact sizes are larger** than ECDH / FFDH

NIST Post-Quantum Cryptography Standardization

- **FIPS 204**

- Specifies **ML-DSA**
- Based on **CRYSTALS-Dilithium**
- Defines 3 parameter sets (ECDSA sizes are in parenthesis for comparison)

Variant	Public Key	Private Key	Signature	AES Equivalence
ML-DSA-44	1312 (64)	2528 (32)	2420 (64)	128
ML-DSA-65	1952 (96)	4000 (48)	3293 (96)	192
ML-DSA-87	2592 (131)	4864 (66)	4595 (131)	256

- Appropriate for replacement of quantum-vulnerable **digital signature algorithms** (ex: ECDSA, RSA)
- **Performance is on par**, cryptographic **artifact sizes are larger** than ECDSA / RSA

NIST Post-Quantum Cryptography Standardization

- **FIPS 205**

- Specifies **SLH-DSA**
- Based on **SPHINCS+** winner
- Defines 12 parameter sets

Table 1. SLH-DSA parameter sets

	n	h	d	h'	a	k	lg_w	m	sec level	pk bytes	sig bytes
SLH-DSA-SHA2-128s	16	63	7	9	12	14	4	30	1	32	7 856
SLH-DSA-SHAKE-128s											
SLH-DSA-SHA2-128f	16	66	22	3	6	33	4	34	1	32	17 088
SLH-DSA-SHAKE-128f											
SLH-DSA-SHA2-192s	24	63	7	9	14	17	4	39	3	48	16 224
SLH-DSA-SHAKE-192s											
SLH-DSA-SHA2-192f	24	66	22	3	8	33	4	42	3	48	35 664
SLH-DSA-SHAKE-192f											
SLH-DSA-SHA2-256s	32	64	8	8	14	22	4	47	5	64	29 792
SLH-DSA-SHAKE-256s											
SLH-DSA-SHA2-256f	32	68	17	4	9	35	4	49	5	64	49 856
SLH-DSA-SHAKE-256f											

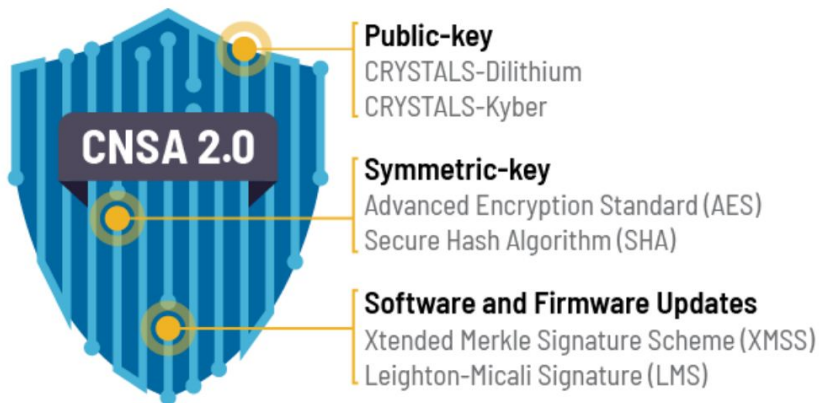
- Appropriate for replacement of **Stateful Hash-based Signature Schemes LMS/XMSS** (but NOT suggested for use in CNSA 2.0 guidance)

NSA (National Security Agency)

Commercial National Security Algorithm Suite 2.0 (CNSA 2.0)

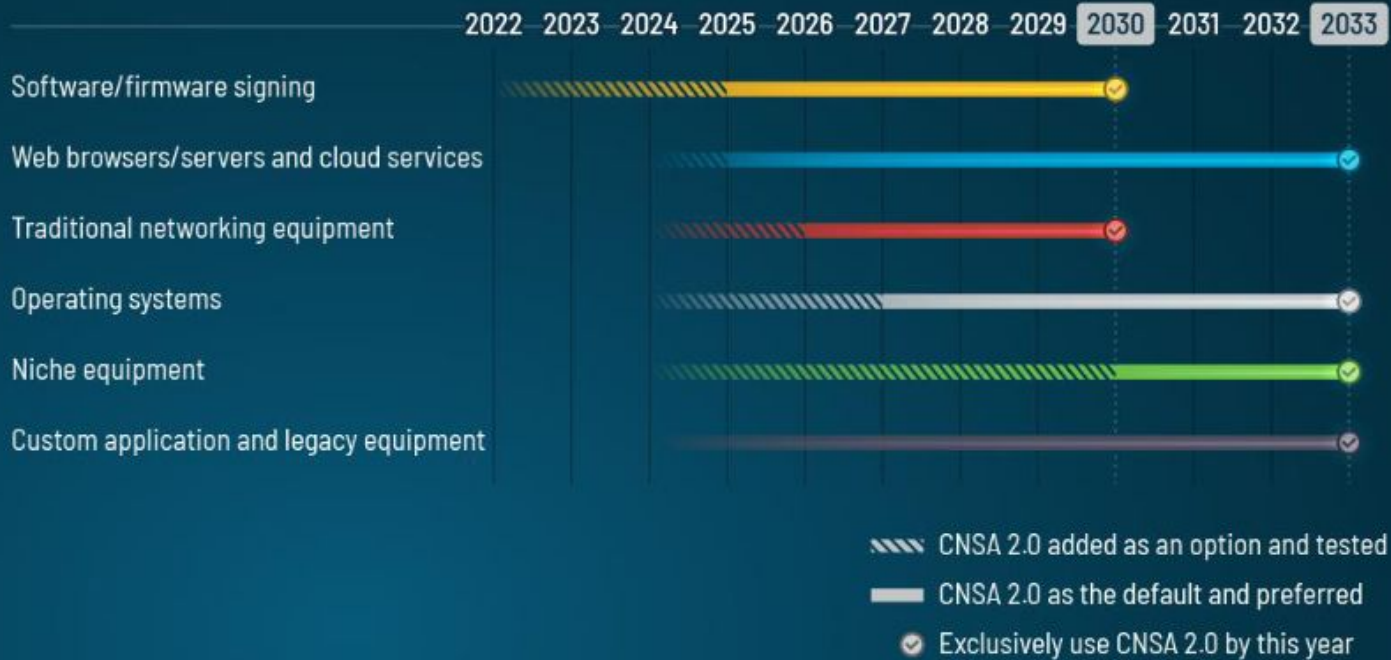


Announcing the Commercial National Security Algorithm Suite 2.0



- Notifies parties involved in **National Security Systems (NSS)** that **new requirements are coming**
- Requirements will **mandate a switch to post-quantum algorithms by 2030**
- Mandates dropping requirements for conventional algorithms and requiring only post-quantum algorithms by **2033**
- Released **September 2022**

CNSA 2.0 Timeline



wolfSSL

PQC History and Current Status

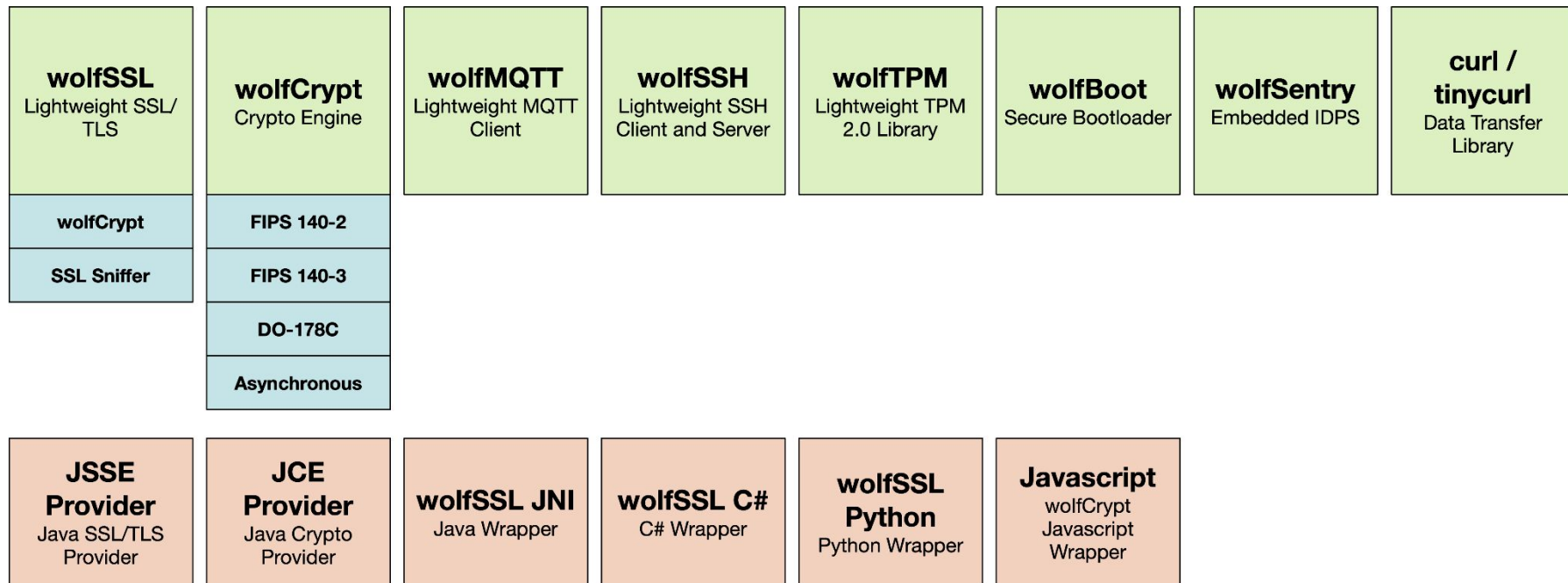
wolfSSL and Post-Quantum Cryptography

- **wolfSSL** has included **NTRU** support since **2010!**
 - We originally partnered with Security Innovation to bring experimental **NTRU** support to wolfSSL
 - **NTRU**
 - “**N**-th degree **T**runcated polynomial **R**ing **U**nits”
 - Deprecated and removed from wolfSSL as it is no longer being considered for standardization

wolfSSL and Post-Quantum Cryptography

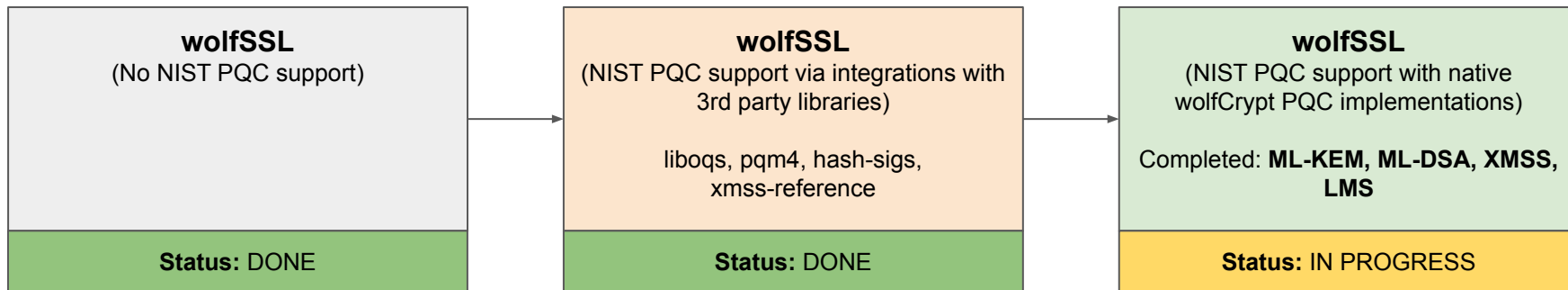
- In **2015**, wolfSSL added support for **QSH** (Quantum Safe Hybrid) handshake extension into TLS 1.2
 - <https://tools.ietf.org/html/draft-whyte-qsh-tls12-02> ; expired
 - Deprecated and removed from wolfSSL

wolfSSL Products



wolfSSL and Post-Quantum Cryptography

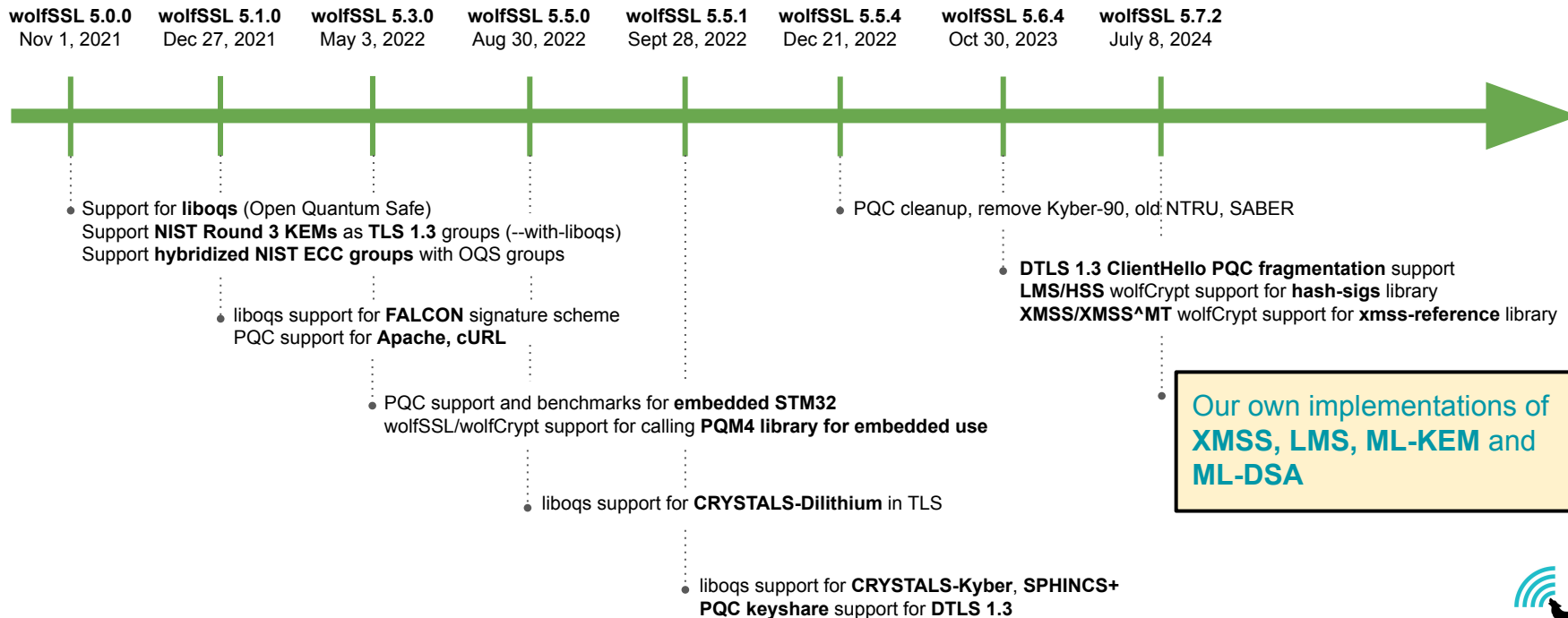
- **Current status and roadmap** is following a **step-wise approach**



- Native wolfCrypt implementations will be optimized for **footprint**, **memory usage**, and **performance**
- Falcon and SLH-DSA are the only ones left

wolfSSL and Post-Quantum Cryptography

Flagship product. Cryptographic algorithm and TLS/DTLS protocol implementations.



Our own implementations of
**XMSS, LMS, ML-KEM and
ML-DSA**

wolfSSH and Post-Quantum Cryptography

SSHv2 protocol implementation

wolfSSH 1.4.12
Dec 28, 2022

wolfSSH 1.4.14
Jul 7, 2023

----->
More interop, algorithm testing!

- Add Hybrid ECDH-P256 Kyber-Level1

- ecc_p256-kyber-level1 hybrid interop with and AWS Transfer Family (Panos K. of AWS Security is technical lead of the feature)

MQTTv5 publish and subscribe protocol implementation

wolfMQTT 1.14.0

Jul 25, 2022

----->
More interop, algorithm testing!

- **PQC** support for **KYBER_LEVEL_1**, **P256_KYBER_LEVEL1**, **FALCON_LEVEL1**
Uses **liboqs** with **wolfSSL**
Connects to **Mosquitto MQTT broker** integrated with OpenQuantumSafe project
PQC with TLS 1.3 support for MQTT!

wolfBoot and Post-Quantum Cryptography

Fully featured low level boot-loader for firmware on embedded systems

wolfBoot 2.0.0

Nov 11, 2023

More algorithms and testing!

- PQC Stateful Hash-Based Signature Scheme support with:
 - + LMS / HSS
 - + XMSS / XMSS^{MT}

cURL and Post-Quantum Cryptography

Command-line tool and library, non-interactive client for many protocols

curl 7.80.0
Nov 10, 2021

----->
More algorithms and testing!

- Support **TLS 1.3 with KEM** using **wolfSSL** with **liboqs**
Support **KEM** or **Hybrid KEM** with:
 - + **KYBER** and **NTRU**
 - + **Hybrid**: ECC P256, P384, P521

NIKE vs KEM

Some Definitions

- **NIKE**
 - Non-Interactive Key Exchange
 - Examples: DH, ECDH
- **KEM**
 - Key Encapsulation Mechanism
 - Examples: RSA, ML-KEM

NIKE APIs and Protocol

- **API**

- KeyGen(Out my_public_key, Out my_private_key);
- Derive(In my_private_key, In peer_public_key, Out shared_secret);

- **Protocol:**

Client does KeyGen()

-----> client_public_key ----->

Server does KeyGen()

<----- server_public_key <-----

Client does Derive()

Server does Derive()

Client has shared_secret

Server has shared_secret

KEM APIs and Protocol

- **API**

- KeyGen(Out my_public_key, Out my_private_key);
- Encapsulate(In peer_public_key, Out ciphertext, Out shared_secret);
- Decapsulate(In my_private_key, In ciphertext, Out shared_secret);

- **Protocol:**

Client does KeyGen()

-----> client_public_key ----->

Server does Encapsulate()

<----- server_ciphertext <-----

Client does Decapsulate()

Client has shared_secret

Server has shared_secret

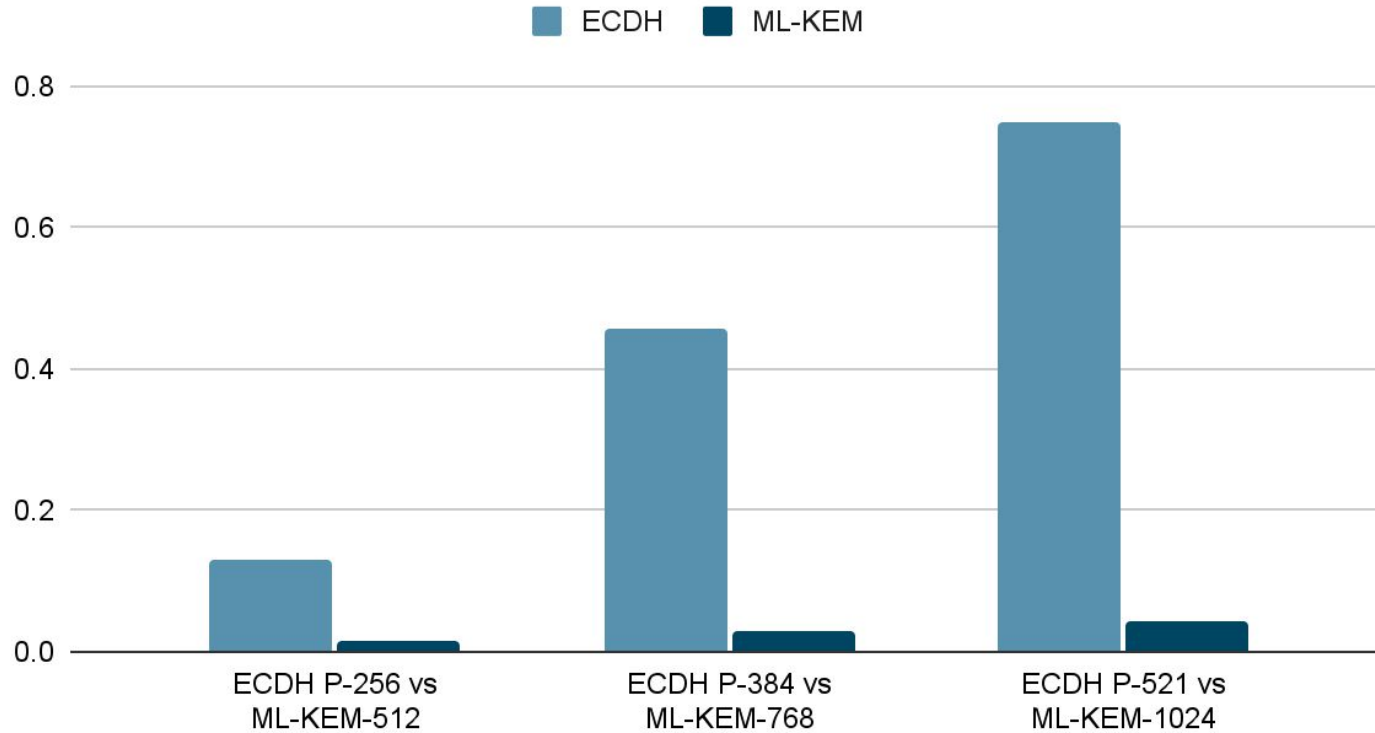
So What?

- In a NIKE, Keygen() and Derive() are each called **twice**
- In a KEM, Keygen(), Encapsulate() and Decapsulate() are each called **once**
- Therefore, when comparing the benchmarks for NIKEs and KEMs, you must **double** the time for the operations of the NIKEs for a fair comparisons!

Benchmarking ECC vs ML-DSA and ML-KEM

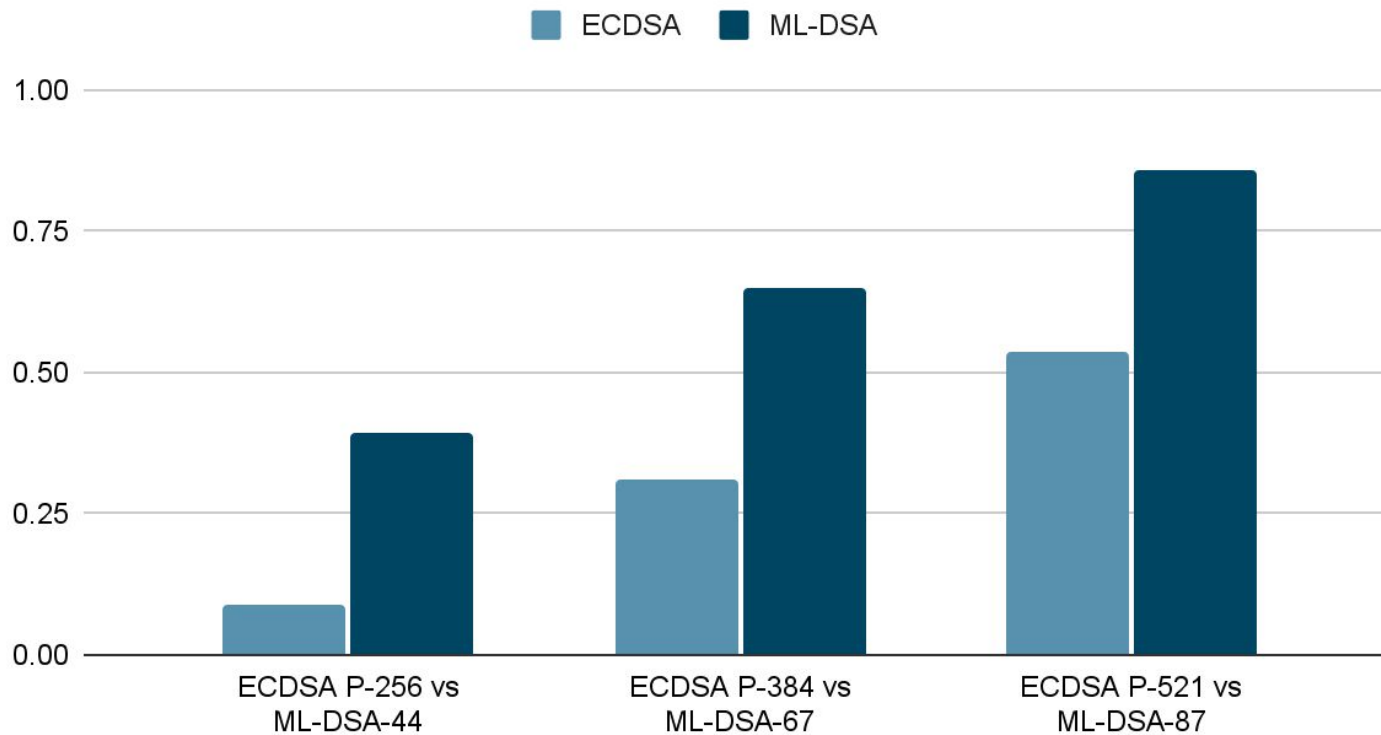
ECDH vs ML-KEM (DOUBLED!)

Average Operation Time in Milliseconds (Smaller is Better)



ECDSA vs ML-DSA

Average Operation Time in Milliseconds (Smaller is Better)



P-256 ECC vs ML-DSA-44 and ML-KEM-512

ECC	SECP256R1	key gen	74600 ops took 1.001 sec, avg 0.013 ms, 74507.858 ops/sec
ECDHE	SECP256R1	agree	19500 ops took 1.004 sec, avg 0.052 ms, 19415.926 ops/sec
ECDSA	SECP256R1	sign	48500 ops took 1.001 sec, avg 0.021 ms, 48460.012 ops/sec
ECDSA	SECP256R1	verify	18400 ops took 1.003 sec, avg 0.055 ms, 18338.526 ops/sec
ML-DSA	44	key gen	15900 ops took 1.006 sec, avg 0.063 ms, 15807.566 ops/sec
ML-DSA	44	sign	3900 ops took 1.012 sec, avg 0.260 ms, 3853.441 ops/sec
ML-DSA	44	verify	14100 ops took 1.004 sec, avg 0.071 ms, 14041.845 ops/sec
ML-KEM	512	key gen	226500 ops took 1.000 sec, avg 0.004 ms, 226493.466 ops/sec
ML-KEM	512	encap	214800 ops took 1.000 sec, avg 0.005 ms, 214740.610 ops/sec
ML-KEM	512	decap	127400 ops took 1.001 sec, avg 0.008 ms, 127305.302 ops/sec

P-384 ECC vs ML-DSA-65 and ML-KEM-768

ECC	SECP384R1	key gen	22900 ops took 1.002 sec, avg 0.044 ms, 22847.325 ops/sec
ECDHE	SECP384R1	agree	5500 ops took 1.016 sec, avg 0.185 ms, 5414.530 ops/sec
ECDSA	SECP384R1	sign	15700 ops took 1.004 sec, avg 0.064 ms, 15636.482 ops/sec
ECDSA	SECP384R1	verify	5000 ops took 1.005 sec, avg 0.201 ms, 4975.211 ops/sec
ML-DSA	65	key gen	8500 ops took 1.003 sec, avg 0.118 ms, 8471.859 ops/sec
ML-DSA	65	sign	2400 ops took 1.001 sec, avg 0.417 ms, 2398.061 ops/sec
ML-DSA	65	verify	8800 ops took 1.002 sec, avg 0.114 ms, 8778.814 ops/sec
ML-KEM	768	key gen	131500 ops took 1.000 sec, avg 0.008 ms, 131447.224 ops/sec
ML-KEM	768	encap	128700 ops took 1.001 sec, avg 0.008 ms, 128624.806 ops/sec
ML-KEM	768	decap	79600 ops took 1.002 sec, avg 0.013 ms, 79457.937 ops/sec

P-521 ECC vs ML-DSA-87 and ML-KEM-1024

ECC	SECP521R1	key gen	13100 ops took 1.004 sec, avg 0.077 ms, 13048.826 ops/sec
ECDHE	SECP521R1	agree	3400 ops took 1.012 sec, avg 0.298 ms, 3360.219 ops/sec
ECDSA	SECP521R1	sign	7400 ops took 1.000 sec, avg 0.135 ms, 7398.915 ops/sec
ECDSA	SECP521R1	verify	3100 ops took 1.012 sec, avg 0.326 ms, 3064.356 ops/sec
ML-DSA	87	key gen	5700 ops took 1.006 sec, avg 0.176 ms, 5668.430 ops/sec
ML-DSA	87	sign	2100 ops took 1.040 sec, avg 0.495 ms, 2019.394 ops/sec
ML-DSA	87	verify	5500 ops took 1.016 sec, avg 0.185 ms, 5415.962 ops/sec
ML-KEM	1024	key gen	87900 ops took 1.001 sec, avg 0.011 ms, 87800.567 ops/sec
ML-KEM	1024	encap	82700 ops took 1.000 sec, avg 0.012 ms, 82683.185 ops/sec
ML-KEM	1024	decap	53300 ops took 1.001 sec, avg 0.019 ms, 53226.549 ops/sec

wolfSSL PQC Readiness and Migration Efforts

NSA CNSA 2.0 Stance on Hybrids

Q: What is NSA's position on the use of hybrid solutions?

A: NSA has confidence in CNSA 2.0 algorithms and will not require NSS developers to use hybrid certified products for security purposes. Product availability and interoperability requirements may lead to adopting hybrid solutions.

NSA recognizes that some standards may require using hybrid-like constructions to accommodate the larger sizes of CRQC algorithms and will work with industry on the best options for implementation.

Source: https://media.defense.gov/2022/Sep/07/2003071836/-1/-1/0/CSI_CNSA_2.0_FAQ_.PDF

NIST Stance on Hybrids in Key Establishment

Q: Is it possible for a hybrid key-establishment mode (i.e: ECC and ML-KEM concatenation) to be performed in a FIPS 140 approved mode of operation?

A: ... In any of the key derivation methods specified in SP 800 - 56C, the revision would permit a concatenation of Z and T, e.g., $Z||T$, to serve as the shared secret instead of Z. This would require the insertion of T into the coding for the scheme and the FIPS 140 validation code may need to be modified.

NIST Stance on Hybrids in Authentication

Q: Is it possible for dual signature generation or verification to be performed in a FIPS 140 approved mode of operation?

A: ... Like hybrid key establishment schemes, dual signatures can be accommodated by current standards in “FIPS mode,” as defined in FIPS 140, provided at least one of the component methods is a properly implemented, NIST-approved signature algorithm.

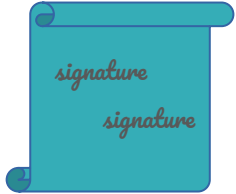
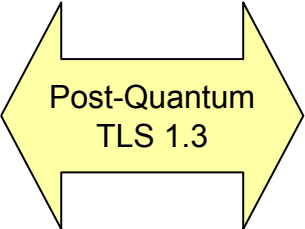
We support them!

- D(TLS) 1.3
 - P256_KYBER_LEVEL1
 - P384_KYBER_LEVEL3
 - P521_KYBER_LEVEL5
 - Dilithium hybridized with ECC via dual algorithm certificates as specified in the 2019 edition of the X.509 standard
 - X9.146 (Banking standards body) TLS 1.3 extensions for dual algorithm verification
- SSH
 - Ecc_p256-kyber-level1 Key exchange

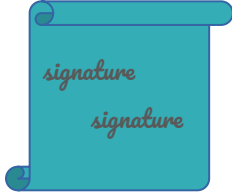
Example Hybrid Certificate Usage in TLS 1.3

Client

Server



X.509 Root CA
Certificate



X.509 Server
Certificate



What Others are Doing Together with wolfSSL

Open Source Project Integrations

- **Web Servers**

- Apache

https://github.com/wolfSSL/osp/blob/master/apache-httpd/README_post_quantum.md

- Nginx

<https://github.com/wolfSSL/wolfssl-nginx?tab=readme-ov-file#post-quantum-algorithms>

- Lighttpd

<https://github.com/wolfSSL/osp/tree/master/lighttpd/lighttpd-1.4.50/wolfStartUp>

- **Secure Tunneling**

- Stunnel

https://github.com/wolfSSL/osp/blob/master/stunnel/5.57/README_UNIX.md#stunnel-with-experimental-post-quantum-algorithms

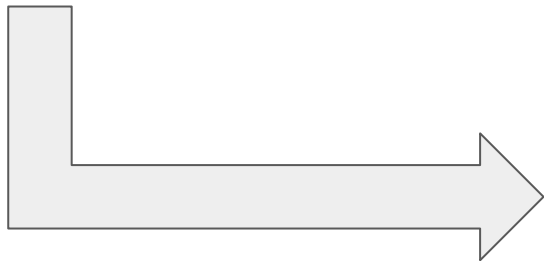
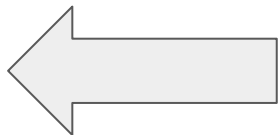
Migration to Post-Quantum Cryptography

The advent of quantum computing technology will compromise many of the current cryptographic algorithms, especially public-key cryptography, which is widely used to protect digital information. Most algorithms on which we depend are used worldwide in components of many different communications, processing, and storage systems. Once access to practical quantum computers becomes available, all public-key algorithms and associated protocols will be vulnerable to criminals, competitors, and other adversaries. It is critical to begin planning for the replacement of hardware, software, and services that use public-key algorithms now so that information is protected from future attacks.

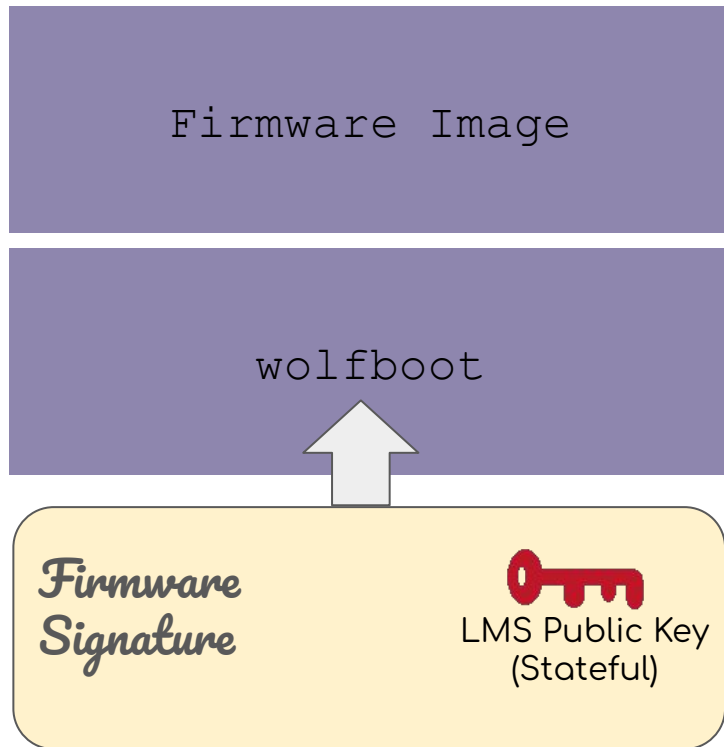


Crypto4A and wolfSSL Interoperability Demo

CRYPTO4A

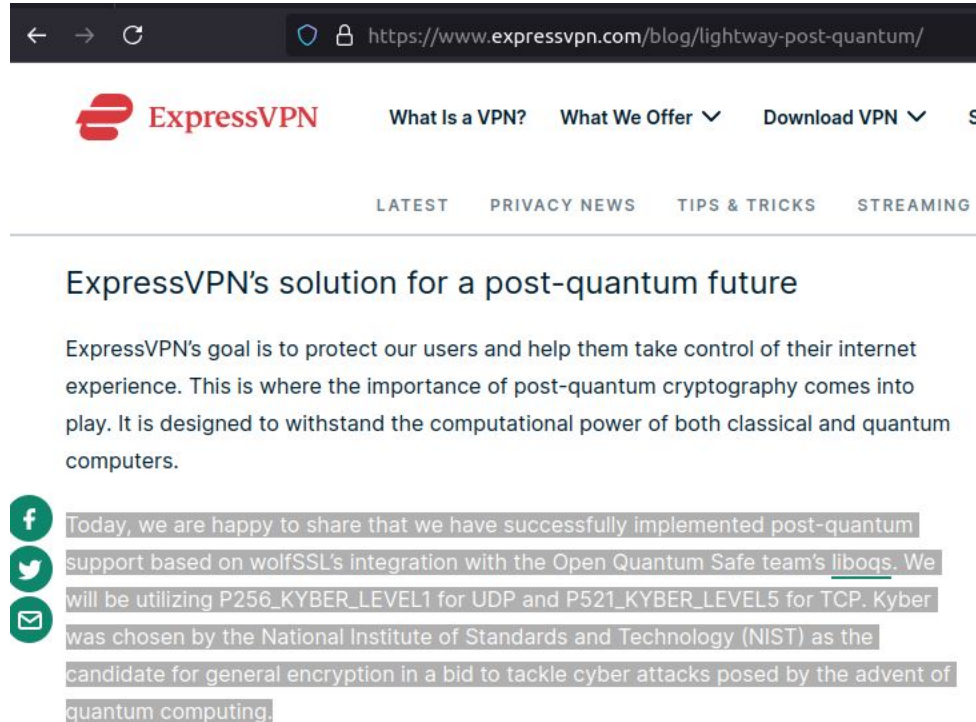


Micro Controller



ExpressVPN Using wolfSSL PQC in Production

<https://www.expressvpn.com/blog/lightway-post-quantum/>



The screenshot shows a web browser displaying the ExpressVPN website. The address bar shows the URL <https://www.expressvpn.com/blog/lightway-post-quantum/>. The ExpressVPN logo is visible on the left, and navigation links for 'What Is a VPN?', 'What We Offer', and 'Download VPN' are on the right. Below the navigation, there are links for 'LATEST', 'PRIVACY NEWS', 'TIPS & TRICKS', and 'STREAMING'. The main content area features the title 'ExpressVPN's solution for a post-quantum future' and a paragraph of text. Below the text are three social media sharing icons: Facebook, Twitter, and Email. The text of the post is highlighted in a light gray box.

ExpressVPN's solution for a post-quantum future

ExpressVPN's goal is to protect our users and help them take control of their internet experience. This is where the importance of post-quantum cryptography comes into play. It is designed to withstand the computational power of both classical and quantum computers.

Today, we are happy to share that we have successfully implemented post-quantum support based on wolfSSL's integration with the Open Quantum Safe team's [liboqs](#). We will be utilizing P256_KYBER_LEVEL1 for UDP and P521_KYBER_LEVEL5 for TCP. Kyber was chosen by the National Institute of Standards and Technology (NIST) as the candidate for general encryption in a bid to tackle cyber attacks posed by the advent of quantum computing.

Barriers to Migration

- **Inertia and a lack of prioritization**
- **Lack of awareness**
- **(Historical) lack of standards**
- **(Historical) Lack of requirements (Whitehouse memos, NSM-10)**
- **Lack of enforcement (Who is enforcing CNSA 2.0?)**
- **Faulty reasoning about quantum computers**
 - **The question is NOT when will quantum computers break cryptography.**
 - **The question is when do I need to comply with upcoming requirements?**



Thanks!

facts@wolfssl.com