Alle nieuws

How to get un-stuck by Architecting for Resilience 2

4 weeks ago

PQC Benchmarking test details

Context

From mid 2024 to mid 2025 the PCSI partners ran a PQC Benchmarking project. In the final phase of this cybersecurity innovation project, results and lessons learned are shared. For this project a series of 4 blogs are published on the PCSI website. The PQC Benchmarking test details you are currently reading are a technical addendum to the blog titled ‘How to get un-stuck by architecting for resilience’.

Migrating to quantum safe cryptography

It’s important to realize that it’s not simply a matter of drop-in replacing the current quantum-vulnerable algorithms by PQC algorithms. There are several PQC algorithm options available, each of them with different characteristics. Additionally, the PQC algorithms can be used either in a pure form or in combination with classical ones. We call the latter a ‘hybrid’ implementation and it comes in two flavours: ‘hybrid AND’ and ‘hybrid OR’ (for reference see the PQC Migration Handbook). ‘Hybrid AND’ means that both algorithms must be evaluated successfully. In the current phase of the PQC migration process, this hybrid implementation is often recommended.

Test setup

[PW1]Link naar de pagina invoegen

Group	Test case	Key exchange algorithm	Digital signature algorithm
No HTTPS (for reference)	0	-	-
Current algorithms DH & RSA (for reference)	1	ffdhe3072	rsassa-pss
Current algorithms ECC (=classical)	2	x25519	ed25519
Key exchange PQC, signatures classical	3 4	mlkem512 frodo640aes	ed25519 ed25519
Key exchange classical, signatures PQC	5 6 7 8 - 9	x25519 x25519 x25519 x25519 x25519 x25519	mldsa44 falcon512 slh-dsa-sha2-128s (sphincssha2128ssimple) slh-dsa-sha2-128f (sphincssha2128fsimple) slh-dsa-shake-128s slh-dsa-shake-128f (sphincsshake128fsimple)
Key exchange and digital signatures PQC	10 11 12 13 14 15 16 17	mlkem512 mlkem512 mlkem512 mlkem512 frodokem-640 (frodo640aes) frodokem-640 (frodo640aes) frodokem-640 (frodo640aes) frodokem-640 (frodo640aes)	mldsa44 falcon512 slh-dsa-sha2-128s (sphincssha2128ssimple) slh-dsa-sha2-128f (sphincssha2128fsimple) mldsa44 falcon512 slh-dsa-sha2-128s (sphincssha2128ssimple) slh-dsa-sha2-128f (sphincssha2128fsimple)
Key exchange hybrid, signatures classical	18 19	x25519_mlkem512 x25519_frodo640aes	ed25519 ed25519
Key exchange classical, signatures hybrid	20	x25519	mldsa44_ed25519
Key exchange and digital signatures hybrid	21 22	x25519_mlkem512 x25519_frodo640aes	mldsa44_ed25519 mldsa44_ed25519

Table 1: Test setup with 22 test configurations

Note: slh-dsa-shake-128s was not available for config in OpenSSL

Executed scenario

Some technical details of the executed scenario:

- All key exchange and digital signature algorithms we wanted to test could be tested, with the exception of the McEliece KEM. The combination of X25519 with slh-dsa-shake-128s could not be tested either. These choices were not supported by our versions of OpenQuantumSafe.
- We took X25519 key exchange with Ed25519 signatures as a baseline, which both use Elliptic Curve Cryptography (ECC). Another common classical choice is RSA signatures and finite field Diffie-Hellman (FFDH) key exchange. We used the combination of finite field Diffie-Hellman using 3072-bit keys (ffdhe3072) for key exchange and RSA signatures using 4096-bit keys (RSASSA-PSS). We include both choices for reference.
- We started testing with 22 configurations as listed in table 1.
- We measured the number of messages per second that we sent/received throughout the TLS connections for the test runs.
- For each message a separate TLS connection was set up and after delivery, it was torn down again. HTTP Keepalive was stopped explicitly.
- Test runs were executed with 1 thread and later re-run using 4 and 8 threads.
- 1 run = 2.500 messages, scaled linearly to 10.000 and 20.000 for the multi-thread runs.
- In all tests AES_256_GCM with SHA384 was used for symmetric encryption.
- CPU usage, memory and network information were observed, not measured.

- Message size was constant, but the TLS handshakes messages were differently sized depending on the used algorithms.

The software versions we used:

Software	Version
Linux	RHEL 9
TLS	1.3
OpenSSL	3.2.3
Liboqs	0.12.1
OQS Provider	0.8.1
cURL	8.12.1
HAProxy	3.1.0
JMeter	5.6.3

Table 2: Software versions

Raw test results

Group	Test case	Key exchange algorithm	Digital signature algorithm	msg/sec 1 thread	msg/sec 4 threads	msg/sec 4 threads
No HTTPS (for reference)	0	-	-	72,5	296,1	416,7
Current algorithms DH&RSA (for reference)	1	ffdhe3072	rsassa-pss	27,3	64,9	71,0
Current algorithms ECC (=classical)	2	x25519	ed25519	56,6	208,7	283,1
Key exchange PQC, signatures classical	3	mlkem512	ed25519	54,8	200,6	274,4
	4	frodo640aes	ed25519	48,8	171,1	255,7
Key exchange classical, signatures PQC	5	x25519	mldsa44	54,7	204,5	279,1
	6	x25519	falcon512	53,6	204,7	284,3
	7	x25519	slh-dsa-sha2-128s (sphincssha2128ssimple)	6,0	6,6	6,6
	8	x25519	slh-dsa-sha2-128f (sphincssha2128fsimple)	38,1	82,2	93,5
	-	x25519	slh-dsa-shake-128s	N/A*	N/A*	N/A*
	9	x25519	slh-dsa-shake-128f (sphincsshake128fsimple)	29,0	52,6	57,4
Key exchange and digital signatures PQC	10	mlkem512	mldsa44	55,3	208,3	271,5
	11	mlkem512	falcon512	53,8	194,0	278,7
	12	mlkem512	slh-dsa-sha2-128s (sphincssha2128ssimple)	6,0	6,5	6,7
	13	mlkem512	slh-dsa-sha2-128f (sphincssha2128fsimple)	37,3	79,2	93,4
	14	frodokem-640 (frodo640aes)	mldsa44	47,8	169,2	247,5
	15	frodokem-640 (frodo640aes)	falcon512	48,7	164,4	242,4
	16	frodokem-640 (frodo640aes)	slh-dsa-sha2-128s (sphincssha2128ssimple)	5,9	6,5	6,7
	17	frodokem-640 (frodo640aes)	slh-dsa-sha2-128f (sphincssha2128fsimple)	35,5	74,0	84,0
Key exchange hybrid, signatures classical	18	x25519_mlkem512	ed25519	49,1	193,1	261,5
	19	x25519_frodo640aes	ed25519	46,3	160,6	234,4
Key exchange classical, signatures hybrid	20	x25519	mldsa44_ed25519	50,3	189,3	249,7
Key exchange and digital signatures hybrid	21	x25519_mlkem512	mldsa44_ed25519	45,8	180,5	238,6
	22	x25519_frodo640aes	mldsa44_ed25519	38,5	159,1	211,5

Table 3: Raw test results

Detailed analysis

The results show the messages per second that we sent/received throughout the TLS connections using various combinations of Key Encapsulation Mechanisms and Digital Signature Schemes and multiple settings of numbers of threads used.

Here are some of the insights.

Baseline references

Currently, often finite field Diffie-Hellman (ffdhe3072) is used for key exchange, combined with RSA signatures using 4096-bit keys (RSASSA-PSS). However, the 'classical elliptic curve (ECC)' combination of X25519 with Ed25519 is significantly faster and also very common. We used both as baseline references. They serve as a good reference point for what classical cryptography has to offer at the moment.

The messages per second for classical ECC was two times higher than for classical FFDH/RSA at 1 thread and the performance increase got increasingly better for 4 threads and 8 threads (3 times higher and 4 times higher respectively).

This means that any results that are similar to the ECC baseline are also much better than the currently often used FFDH/RSA configurations.

Multithreading

Using four or eight threads seems to give a linear performance increase for almost all KEM and DSA configurations, except for size-optimized SLH-DSA. This means that CPU usage is generally not the bottleneck. It is likely that the network delays are large enough for the CPU to yield CPU time to parallelize processes while waiting for responses.

Key exchange

Pure PQC key exchange

Among the post-quantum KEMs, ML-KEM512 gives very similar performance results compared to X25519. Surprisingly, the results for Frodo640aes are also quite close to that of ML-KEM512.

The performance loss for ML-KEM512 compared to X25519 is around 3% for all numbers of threads. The performance loss for Frodo640aes compared to X25519 is 13%, 18% and 10% for 1, 4 and 8 threads respectively.

FrodoKEM has security benefits, because its security assumptions are more conservative than those of ML-KEM. Given the minor performance difference, it could be a relevant option for organizations that want to stay on the cautious side.

Hybrid key exchange

The extra performance loss of the hybrid KEMs versus the single KEMs is a lot less than expected, at most 10%. Specifically, hybrid ML-KEM + X25519 is 10%, 4%, 5% slower than ML-KEM alone, and hybrid Frodo640aes + X25519 is 5%, 6% and 8% slower than Frodo640aes alone, for 1,4 and 8 threads.

Digital signatures

Pure PQC signatures

The performance results differ quite a bit when combining X25519 key exchange with various different post-quantum signature schemes. ML-DSA44 results and Falcon512 results are very similar to Ed25519 (Falcon is sometimes even faster), whilst SLH-DSA results are significantly worse.

ML-DSA44 has a performance loss of at most 4% for all thread configurations. Falcon512 has a performance loss of at most 6% for all thread configurations, and even has a performance increase of 0.5% at 8 threads. SLH-DSA does not gain any benefits by adding threads, which means that it is very heavy for the CPU.

Speed-optimized SLH-DSA with SHA2-128 performs best out of all SLH-DSA configurations. At 1 thread, this gives a performance loss of about 32%, which gets worse with more threads, because the baseline (ECC) does gain performance benefits with more threads.

Compared to classical FFDH/RSA, X25519 key exchange with speed-optimized SLH-DSA using SHA-128 does perform better: 40%, 27% and 30% increase for 1 thread, 4 threads and 8 threads respectively.

For size-optimized SLH-DSA with SHA2-128, the performance loss is 90% at 1 thread and again gets worse with more threads. No results could be retrieved for size-optimized SLH-DSA with SHAKE.

Hybrid signatures

The only available hybrid signature scheme is ML-DSA44 combined with Ed25519. The performance loss is at most 12%.

The hybrid mode that was used is the composite mode, which means that both signatures need to be verified and both need to be valid.

If signature verification would take the majority of the performance overhead, we would expect about a 50% loss here, since ML-DSA44 and Ed25519 perform similarly. The results imply that this is not the case and that hybrid signatures come with a limited performance cost.

Pure PQC key exchange and digital signatures

From the previous results, we would expect that combining ML-KEM512 with ML-DSA44 or Falcon512 would not give a big performance loss, which is verified by the results: ML-KEM512 + ML-DSA44 has a performance loss of at most 4%, ML-KEM512 + Falcon512 has a performance loss of at most 5%.

Combining Frodo640aes with ML-DSA44 or Falcon512 gives a more significant performance loss: Frodo640aes + ML-DSA44 has a performance loss of at most 20%, Frodo640aes + Falcon512 has a performance loss of at most 21%.

Hybrid key exchange and digital signatures

Combining the fastest classical algorithms with the fastest post-quantum algorithms for KEMs and DSAs, we get about a 20% performance decrease. Using X25519 and ML-KEM512 for the key exchange and Ed25519 and ML-DSA44 for the signature schemes, we get a performance loss of 20%, 14% and 15% for 1 thread, 4 threads and 8 threads respectively.

TLS message size with PQC

Even though no packet drops were observed in the experiment, and the performance data did not indicate hinder from bigger packets on the network, it is interesting to see the actual size increase. This might become an issue in bigger or congested networks or with other applications. Since the key exchange algorithms and DSAs are used in different messages in TLS 1.3, we can inspect them separately.

With the KEMs there is a big difference between FrodoKEM and ML-KEM: FrodoKEM produces client hello messages that are almost 10x as big as the ones with ML-KEM, and also the server hello increases significantly.

		Baseline		KEM PQC		Kem hybrid	
	KEM	ffdhe3072	x25519	mlkem512	frodo640aes	x25519_mlkem512	x25519_frodo640aes
	Signature	RSASSA-PSS	ED25519	ED25519	ED25519	ED25519	ED25519
TLS 1.3 message	Client hello	778	512	1.194	10.010	1.226	10.042
	Server hello	474	122	858	9.810	890	9.842
	Certificate	1.310	803	803	803	803	803
	CERT verify	520	72	72	72	72	72

Table 4: Message size in bytes for the TLS handshake parts for ML-KEM, FrodoKEM and baselines.

Signatures influence the certificate and cert verify messages. Note that in this test the certificate contains the PQC signing public key (but has no PQC signature itself). The result of size

vs speed optimization in SLH-DSA (Sphincs+) is clearly visible, with a 9.2kB difference.

	KEM	Baseline		Signature PQC					Signature hybrid
		ffdhe3072	x25519	x25519	x25519	x25519	x25519	x25519	x25519
	Signature	RSASSA-PSS	ED25519	mldsa44	falcon512	sphincssha2 128ssimple	sphincssha2 128fsimple	sphincsshake 128fsimple	mldsa44_ed25519
TLS 1.3 message	Client hello	778	512	512	512	512	512	512	512
	Server hello	474	122	122	122	122	122	122	122
	Certificate	1.310	803	4.891	4.472	3.604	3.604	3.604	4.937
	CERT verify	520	72	2.428	664	7.864	17.096	17.096	2.504

Table 5: Message size in bytes for the TLS handshake parts for Digital Signatures and baselines