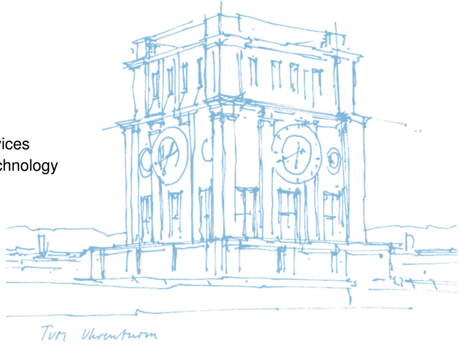
The Performance of Secure Future Networks: Post-Quantum TLS

Markus Sosnowski

Chair of Network Architectures and Services
School of Computation, Information, and Technology
Technical University of Munich



This presentation is based on a collaboration with Nokia Bell Labs and our CoNEXT ’23 paper[1]

"The Performance of Post-Quantum TLS 1.3"

CoNEXT 2023

Markus Sosnowski[2], Florian Wiedner[2], Eric Hauser[2], Lion Steger[2], Dimitrios Schoinianakis[3], Sebastian Gallenmüller[2], and Georg Carle[2]

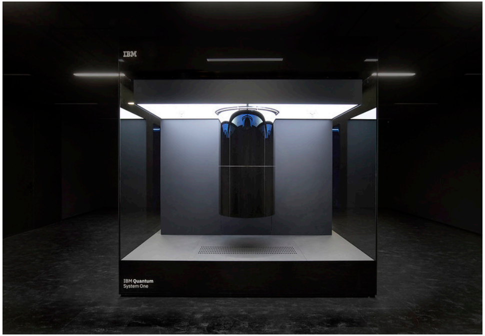
[2] Technical University of Munich, Germany
[3] Nokia Bell Labs, Athens, Greece

[1] M. Sosnowski, F. Wiedner, E. Hauser, *et al.*, “The Performance of Post-Quantum TLS 1.3,” in *Proc. of the International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, Paris, France, Dec. 2023. DOI: 10.1145/3624354.3630585

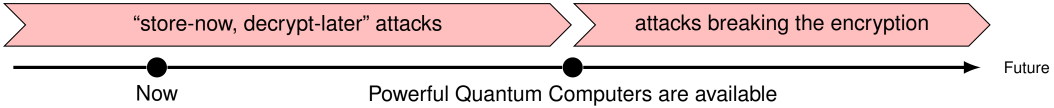
Quantum Computing in a Nutshell

Changing the Rules of the game

1. Traditional asymmetric cryptography will be broken with the availability of powerful quantum computers
2. Several new Post-Quantum Cryptographic (PQC) algorithms are proposed that claim to be resilient against quantum computers
3. Currently, the National Institute of Standards and Technology (NIST) runs a competition where the winners will be standardized



Motivation



→ We have to make our communication (esp., via TLS) post-quantum-safe **as soon as possible**!

→ What are the performance implications of using such PQ-safe TLS in our (6G-)networks **now**?

PQC with TLS:

- changes the CPU costs
- increases amount of exchanged data

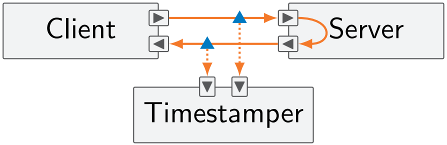
$\Rightarrow$ **?** $\Leftarrow$

Real-world systems are complex

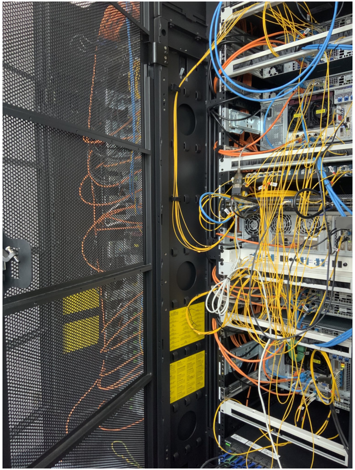
- 6G, 5G, LTE-M, Fiberglass, ...
- hardware
- libraries
- TCP
- TLS
- etc.

Background

How to measure TLS performance?



- Client and Server run the PQ-safe OpenSSL[2]
- The Timestamper captures traffic with an optical splitter, preventing potential measuring bias
- We emulated different constrained network scenarios (low bandwidth, high delay, etc.) with `netem`[3]
- Setup:
  - CPU: Intel Xeon D-1518 (4 $\times$ 2.20 GHz)
  - NIC: Intel X552 (2 $\times$ 10 Gbit/s)
  - OS: Debian Bullseye (Kernel 5.10)



[2] The Open Quantum Safe Project, *OQS-OpenSSL 1.1.1*, [Online]. Available: `https://github.com/open-quantum-safe/openssl`

[3] linux network emulation tool

Background

What should we measure?

TLS is designed independently of the underlying cryptographic algorithms

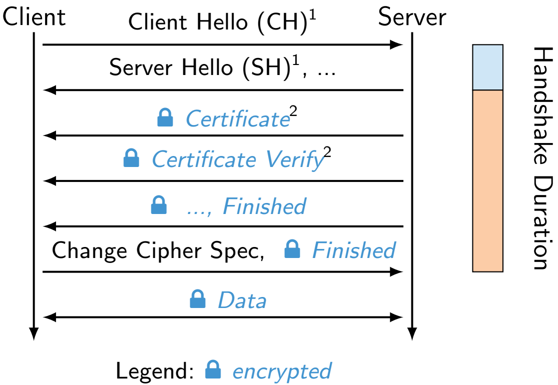
$\rightarrow$ No new PQ TLS protocol is necessary, only different algorithms need to be negotiated

Asymmetric cryptography is only used in the TLS handshake

$\rightarrow$ We only have to analyze the initial handshake

Background

What should we measure?



- [1] Affected by PQ Key Agreements (KAs)
- [2] Affected by PQ Signature Algorithms (SAs)
- → We can measure the handshake latency without decryption

Client Hello → Server Hello | Server Hello → Client Finished

Background

Relevant Post-Quantum Algorithms for TLS

Key Agreement (KA)	Signature Algorithm (SA)
CRYSTALS-KYBER[4]	**CRYSTALS-Dilithium**[2]
Bike	**FALCON**
HQC	**(SPHINCS+)**[2]
State-of-the-Art pre-quantum:	
Elliptic Curves	RSA

- NIST announced four PQ candidates to be **standardized**[5]
- SPHINCS+ is very resource expensive
- Additional algorithms are still in consideration

4 Recent drafts change the names to ML-KEM, ML-DSA, and SLH-DSA

5 NIST PQC Team, "PQC Standardization Process: Announcing Four Candidates to be Standardized, Plus Fourth Round Candidates," (2022), [Online]. Available: https://csrc.nist.gov/News/2022/pqc-candidates-to-be-standardized-and-round-4.

Key Insights

1. **TLS Latency is influenced by the OpenSSL message handling**

2. **Large PQ key sizes are a bottleneck in low-bandwidth environments**

3. **1-RTT PQ TLS 1.3 can take several RTTs**

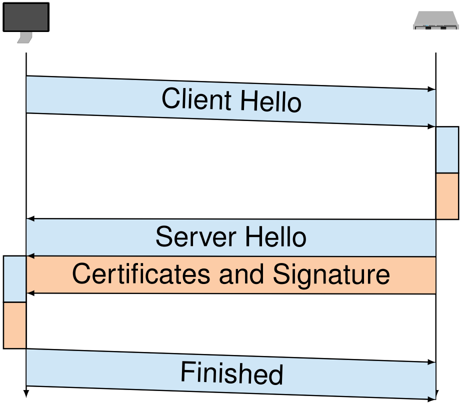
4. **In the right conditions, PQ TLS can be fast!**

TLS Latency is influenced by the OpenSSL message handling

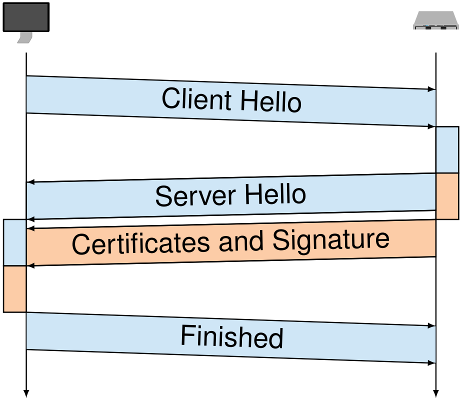
1. **TLS Latency is influenced by the OpenSSL message handling**
2. **Large PQ key sizes are a bottleneck in low-bandwidth environments**
3. **1-RTT PQ TLS 1.3 can take several RTTs**
4. **In the right conditions, PQ TLS can be fast!**

TLS Latency is influenced by the OpenSSL message handling

Some combinations of PQ KA and SA were faster than expected! Why?



Default OpenSSL Behavior



Sometimes: Early Server Hello

TLS Latency is influenced by the OpenSSL message handling

Explanation

- OpenSSL used an internal TLS message buffer of 4096 Bytes
- If the messages exceeded the buffer, the content was flushed early (improving the latency)
- The key sizes of PQ KAs/SAs are significantly larger, triggering an early Server Hello more likely

$\Rightarrow$ To get consistent results, we modified OpenSSL to flush the Server Hello directly after computation

Large PQ key sizes are a bottleneck in low-bandwidth environments

1. TLS Latency is influenced by the OpenSSL message handling

2. **Large PQ key sizes are a bottleneck in low-bandwidth environments**

3. 1-RTT PQ TLS 1.3 can take several RTTs

4. In the right conditions, PQ TLS can be fast!

Large PQ key sizes are a bottleneck in low-bandwidth environments

Results (Excerpt): Emulated Scenarios

NIST Security	KA	SA	No Emulation	Low Bandwidth (1 Mbit/s)	Data Exchanged
Level ≤ 2	X25519	rsa:2048	1.77	14.11	2 kB
	X25519	**falcon512**	1.44	25.93	4 kB
	X25519	**dilithium2**	1.22	58.66	8 kB
Level 5	X25519	**falcon1024**	2.23	43.73	6 kB
	X25519	**dilithium5**	1.46	106.53	14 kB

Legend: **post-quantum** Full Handshake Latency (ms) **Note**: different bar scales per emulation!

- The Dilithium latency increases considerably more than the rest
- The larger key sizes of PQC are a bottleneck in low low-bandwidth environments

1-RTT PQ TLS 1.3 can take several RTTs

1. TLS Latency is influenced by the OpenSSL message handling

2. Large PQ key sizes are a bottleneck in low-bandwidth environments

3. **1-RTT PQ TLS 1.3 can take several RTTs**

4. In the right conditions, PQ TLS can be fast!

1-RTT PQ TLS 1.3 can take several RTTs

Results (Excerpt): Emulated Scenarios

NIST Security	KA	SA	5G (4% loss, 44 ms RTT, and 880 Mbit/s)	Data Sent by Server
Level 1	**kyber512**	rsa:2048	46.44	2 kB
	hqc128	rsa:2048	46.31	6 kB
Level 5	**kyber1024**	rsa:2048	46.51	3 kB
	hqc256	rsa:2048	92.77	16 kB

Legend: **post-quantum** ☐ Full Handshake Latency (ms)

- TLS handshake are (usually) directly after the TCP handshake ($\Rightarrow$ still at slow start minimum)
- Initial TCP Congestion Window (usually): $10 \times MSS = 10 \times 1460\,\text{B} \approx 15\,\text{kB}$
- Can be tuned on servers, especially if using both PQ KA and SA!

1-RTT PQ TLS 1.3 can take several RTTs

Results (Excerpt): Emulated Scenarios

NIST Security	KA	SA	LTE-M (10% loss, 200 ms, RTT 1 Mbit/s)	Data Sent by Server
Level 1	**kyber512**	rsa:2048	220.02	2 kB
	hqc128	rsa:2048	251.31	6 kB
Level 5	**kyber1024**	rsa:2048	226.95	3 kB
	hqc256	rsa:2048	706.85	16 kB

Legend: **post-quantum** Full Handshake Latency (ms)

- High packet loss amplifies the effect of the additional RTTs

In the right conditions, PQ TLS can be fast!

1. TLS Latency is influenced by the OpenSSL message handling

2. Large PQ key sizes are a bottleneck in low-bandwidth environments

3. 1-RTT PQ TLS 1.3 can take several RTTs

4. **In the right conditions, PQ TLS can be fast!**

In the right conditions, PQ TLS can be fast!

Measurement Results (Excerpt)

NIST Security	KA	SA	Handshake Latency Median (ms)
Level 1,2	X25519	rsa:2048	0.25, 1.48
	bikel1	rsa:2048	0.24, 2.79
	hqc128	rsa:2048	0.27, 1.48
	kyber512	rsa:2048	0.20, 1.78
	X25519	**falcon512**	0.36, 1.02
	X25519	**dilithium2**	0.39, 0.84

Legend: **post-quantum** ■ Client Hello → Server Hello ■ Server Hello → Client Finished

- PQ KAs offer competitive performance compared to X25519
- PQ SAs are faster compared to traditional RSA_{2048} signatures

In the right conditions, PQ TLS can be fast!

Measurement Results (Excerpt)

NIST Security	KA	SA	Handshake Latency Median (ms)
Level 5	p521	rsa:2048	6.97 5.30
	hqc256	rsa:2048	0.72, 1.92
	kyber1024	rsa:2048	0.25, 1.78
	X25519	**dilithium5**	0.36, 1.10
	X25519	**falcon1024**	0.38, 1.89

Legend: **post-quantum** Client Hello → Server Hello Server Hello → Client Finished

- PQ KAs offer competitive performance compared to X25519
- PQ SAs are faster compared to traditional RSA_{2048} signatures
- On higher levels, they are significantly faster than p521/RSA_{2048}

Conclusion

- Fast PQ-safe TLS 1.3 is possible!
- Sometimes, the algorithms are a trade–off between CPU and bandwidth
- Performance tuning factors arise: initial TCP CWND and the TLS message handling
- The effect of packet loss is amplified

Read our paper[6] for additional findings:

- More algorithms and variants. We examined 15 KA, 14 SA, and 14 hybrid algorithm variants
- There is no performance drawback in using hybrid algorithms (with the pre-quantum as bottleneck)
- Whitebox measurements revealing library usages
- PQC algorithm recommendations
- Open-sourced experiment scripts, measurement data, and evaluation code

tumi8.github.io/pqs-tls-measurements

[6] M. Sosnowski, F. Wiedner, E. Hauser, *et al.*, "The Performance of Post-Quantum TLS 1.3," in *Proc. of the International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, Paris, France, Dec. 2023. DOI: 10.1145/3624354.3630585

Appendix: Key Agreements (KAs)

Lvl	KA	SA	Handshake Latency Median (ms)
1	X25519	rsa:2048	0.25, 1.48
	bikel1	rsa:2048	0.24, 2.79
	hqc128	rsa:2048	0.27, 1.48
	kyber512	rsa:2048	0.20, 1.78
	p256_bikel1	rsa:2048	0.42, 2.58
	p256_hqc128	rsa:2048	0.52, 1.31
	p256_kyber512	rsa:2048	0.51, 1.81
3	p384	rsa:2048	3.09, 2.63
	bikel3	rsa:2048	0.42, 6.07
	hqc192	rsa:2048	0.53, 1.40
	kyber768	rsa:2048	0.25, 1.82
	p384_bikel3	rsa:2048	3.23, 9.00
	p384_hqc192	rsa:2048	3.39, 3.30
	p384_kyber768	rsa:2048	3.17, 2.72
5	p521	rsa:2048	6.97, 5.30
	hqc256	rsa:2048	0.72, 1.92
	kyber1024	rsa:2048	0.25, 1.78
	p521_hqc256	rsa:2048	7.52, 9.38
	p521_kyber1024	rsa:2048	7.06, 5.41

Legend: **Pre-Quantum** Hybrid

Appendix: Signature Algorithms (SAs)

Lvl	KA	SA	Handshake Latency Median (ms)
	X25519	rsa:2048	0.25, 1.48
1	X25519	**falcon512**	0.36, 1.02
	X25519	rsa:3072	0.26, 3.41
	X25519	rsa:4096	0.25, 6.88
	X25519	**sphincs128**	0.28, 15.02
	X25519	**p256_falcon512**	0.39, 1.35
	X25519	**p256_sphincs128**	0.28, 15.48
2	X25519	**dilithium2**	0.39, 0.84
	X25519	**p256_dilithium2**	0.39, 1.27
3	X25519	**dilithium3**	0.36, 0.94
	X25519	**sphincs192**	0.27, 23.83
	X25519	**p384_dilithium3**	0.31, 3.86
	X25519	**p384_sphincs192**	0.27, 28.75
5	X25519	**dilithium5**	0.36, 1.10
	X25519	**falcon1024**	0.38, 1.89
	X25519	**sphincs256**	0.27, 49.52
	X25519	**p521_dilithium5**	0.29, 7.55
	X25519	**p521_falcon1024**	0.35, 8.72
	X25519	**p521_sphincs256**	0.27, 60.78

Legend: **Pre-Quantum** Hybrid