Post-Quantum Cryptography in Use: Empirical Analysis of the TLS Handshake Performance

Ronny Döring
Deutsche Telekom AG, T-Labs
Berlin, Germany
ronny.doering@telekom.de

Marc Geitz
Deutsche Telekom AG, T-Labs
Berlin, Germany
marc.geitz@telekom.de

NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium | 978-1-6654-0601-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/NOMS54207.2022.9789913

***Abstract*—With the increasing speed in the development of quantum computers, secure communication is at risk. A promising candidate to replace existing cryptographic patterns with quantum-secure variants is Post-Quantum Cryptography (PQC). This paper presents an empirical analysis of how Post-Quantum Cryptography affects real-world performance compared to classical cryptography. Therefore we analyze the Transport Layer Security (TLS) handshake performance showing that Post-Quantum Cryptography can be as fast or even faster than classical cryptography, depending on the specific encryption and signature algorithms used.**

***Index Terms*—Post-Quantum Cryptography (PQC), Asymmetric Cryptography, TLS handshake, Key Encapsulation Mechanism (KEM), Digital Signatures (SIG), Authentication, Performance of PQC**

I. Introduction

While more and more powerful quantum computers are being developed, we need to think about how they will affect internet security in the near future. It is already known that Shor's algorithm [1] with its ability to split a number into its prime factors will become a threat to some cryptography systems used today. Although there are no quantum computers that are powerful enough to execute Shor's algorithm on a reasonably sized asymmetric key today [2], there will eventually be a quantum computer that can do so in the future.

As this will mainly affect asymmetric cryptography like Rivest-Shamir-Adleman (RSA) and Elliptic Curve Cryptography (ECC), the National Institute of Standards and Technology (NIST) is in the process of standardizing so-called Post-Quantum Cryptography (PQC), which is believed to be secure against future quantum computers [3].
While NIST is considering factors like performance, security, and others to choose algorithms for standardization, we can already explore how PQC might affect real-world performance.

For this purpose, we use a simple client and server setup, perhaps the most widely used application of asymmetric cryptography, to test the performance of different PQC algorithms and compare them to classical ones, like RSA and ECC. The test scenario is, therefore, similar to what happens when browsing the web. Particular interest lies in the performance of the TLS handshake.
The TLS handshake consists of three phases [4]:

- Key Exchange for establishing shared keys
- Server Parameters for establishing other parameters
- Authentication to verify the server (and optionally the client) using digital signatures

In contrast to related work which often only compares specific algorithms (e.g. [5], [6], [7]) or a limited set of algorithms (e.g. [8], [9]), we measure real-world performance using different combinations of chosen algorithms. This not only allows us to make conclusions about how PQC will affect day-to-day usage but also to compare the performance of PQC against classical cryptography.

II. Considerations

A. Choosing algorithms to analyze

The process of establishing a shared key between two parties and verifying their authenticities requires both digital signature and key exchange algorithms. Therefore we examine several combinations of preselected algorithms within the two groups.

We will focus on classical algorithms, like RSA and ECC, as these are the most widely used algorithms today. Both can be used either for digital signatures or key exchange. PQC algorithms, on the other hand, can be split into two groups. There are Key Encapsulation Algorithms (KEM) which are used for the key exchange and signature algorithms (SIG) which are used for the sender authentication. Both groups of algorithms are shown in Tab. I and Tab. II. Tab. I depicts the Key Encapsulation Mechanisms and Tab. II the signature algorithms which made it to the final round of the NIST standardization competition, including the mathematical foundation they are built on.

TABLE I
Key Encapsulation Mechanisms - NIST PQC Third-Round Finalists and Alternate Candidates [10]

Key Encapsulation Algorithm	Mathematical Foundation
Classic McEliece	code-based
CRYSTALS-KYBER	lattice-based
NTRU	lattice-based
Saber	lattice-based
BIKE	code-based
FrodoKEM	lattice-based
HQC	code-based
NTRU Prime	lattice-based
SIKE	isogeny-based

Final and alternate candidates for signature algorithms and their underlying mathematical problems are shown in Tab. II.

TABLE II
SIGNATURE ALGORITHMS - NIST PQC THIRD-ROUND FINALISTS AND ALTERNATE CANDIDATES [10]

Signature Algorithm	Mathematical Foundation
CRYSTALS-DILITHIUM	lattice-based
FALCON	lattice-based
Rainbow	multivariate-polynomial-based
GeMSS	multivariate-polynomial-based
Picnic	hash-based
SPHINCS+	hash-based

Not all combinations of classical and PQC signature and key exchange algorithms could be taken into account for this analysis. The following set of criteria has therefore been defined to choose the most relevant algorithms:

- For the classical algorithms, some popular web pages (e.g. facebook.com, google.com, icloud.com, amazon.com) have been investigated and it has been found that RSA and ECC are widely used
- For the PQC algorithms, we defined:
  - At least one finalist and one alternate candidate of each group should be chosen
  - A variety of underlying mathematical problems should be considered
  - The algorithm should be supported by the testing environment (for more details regarding the testing setup see section III)

The selected KEMs are CRYSTALS-KYBER, Saber, HQC, and SIKE. They are used in combination with the signature algorithms CRYSTALS-DILITHIUM, FALCON, Rainbow, and SPHINCS+ to measure TLS handshake duration.

B. Choosing which variants of algorithms to take into account

Not only PQC but also RSA and ECC have a number of variants that target specific security levels.

The five security strength categories (levels) defined by NIST [11] range from security level 1 which is the lowest to level 5 which is the highest. The security levels are compared to the difficulty to break classical encryption or hashing schemes with appropriate key lengths as seen in Tab. III. Comparing variants across different security levels would not be appropriate, so we only focus on variants that target the same security level.
Tab. IV lists the variants of the selected algorithms and their targeted security levels. The focus is on levels 1,3 and 5 as these are the most widely used security levels. Note that not all algorithms have variants for all the levels.

III. PERFORMANCE MEASUREMENTS

A. Measurement Procedure

In order to measure TLS handshake performance, three core components are required:

- An asymmetric key pair (public and private key)

TABLE III
NIST SECURITY STRENGTH CATEGORIES [11]

Level	Security description
1	At least as hard to break as AES128 (exhaustive key search)
2	At least as hard to break as SHA256 (collision search)
3	At least as hard to break as AES192 (exhaustive key search)
4	At least as hard to break as SHA384 (collision search)
5	At least as hard to break as AES256 (exhaustive key search)

TABLE IV
SELECTED ASYMMETRIC CRYPTOGRAPHY AND THEIR SECURITY STRENGTH CATEGORIES [12] [13]

Group	Variant	Level
RSA^k	RSA-3072	1
	RSA-7680	3
	RSA-15360	5
ECC^k	P-256	1
	P-384	3
	P-521	5
	Curve25519	1
	Curve448	3
Saber	LightSaber	1
	Saber	3
	FireSaber	5
CRYSTALS-KYBER	Kyber512	1
	Kyber768	3
	Kyber1024	5
HQC	HQC-128	1
	HQC-192	3
	HQC-256	5
SIKE	SIDH-p434	1
	SIDH-p610	3
	SIDH-p751	5
CRYSTALS-DILITHIUM	Dilithium2	2
	Dilithium3	3
	Dilithium5	5
FALCON	Falcon-512	1
	Falcon-1024	5
Rainbow	Rainbow-I-Classic	1
	Rainbow-III-Classic	3
	Rainbow-V-Classic	5
SPHINCS+	SPHINCS+-SHAKE256-128f-Robust	1
	SPHINCS+-SHAKE256-192f-Robust	3
	SPHINCS+-SHAKE256-256f-Robust	5

^{k} classical, non Post-Quantum Cryptography

- A webserver using the key pair
- A web client

The key pair is generated using the signature algorithm which can be used during the TLS handshake.
The webserver is started with the generated key pair and accepts a TLS connection with the selected key exchange method. The web client must also use the specified key exchange method to successfully perform the TLS handshake. It is ensured that only full handshakes are performed and after a handshake completes, the connection is dropped. The TLS version used on both server and client is TLS 1.3 [4].

A more detailed description of the test procedure can be found at the corresponding Gitlab [14].

B. Hardware and Software

The following performance measurements were performed on an Intel NUC7PJYH with a Pentium Silver J5005 1,5 GHz Quad-Core processor and 16GB RAM with 2666MHz. The installed operating system is Ubuntu 20.04.3.
Especially the software libraries liboqs (v0.7.1) and OpenSSL (v2021-12rc1) provided by [15] were used for performing the measurements. The libraries were compiled from source using the default configuration with some additional algorithmic variants enabled to match Tab. IV.

It should be noted, that the webserver and client were started on the same machine to minimize network impact.

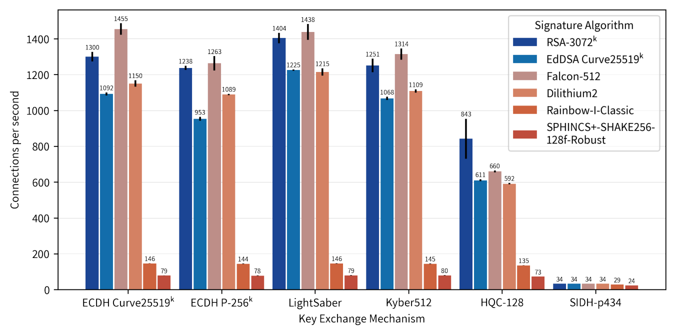
IV. Results

A. TLS handshake Performance

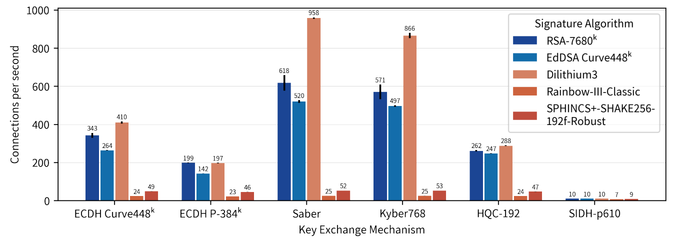
Fig. 1 shows the mean TLS handshake performance for the combinations of selected algorithms and variants for different security levels. The performance is measured in connections per second. The measurements were conducted five times for each security level with irregular time intervals between them. Additionally, the standard deviation of the measurements was calculated.
From the measurements the following results can be seen:

- For security level 1, the TLS handshake performance varies mostly between 1000 and 1400 connections per second. A significant difference between classical and PQC algorithms is not visible. The Saber KEM scheme delivers the best results, up to 1400 connections per second with FALCON signatures. It is significant, that the Rainbow and SPHINCS+ signature schemes reduce the overall performance to around 100 connections per second. Signature schemes in combination with SIKE have overall the lowest performance.
- For security level 3, the TLS handshake performance drops significantly. While the best measurement of a classical key exchange scheme reaches only about 400 connections per second, the PQC KEMs CRYSTALS-KYBER and Saber deliver much better results. Here, CRYSTALS-DILITHIUM seems to be the best choice for SIG yielding results of up to 950 connections per seconds.
- For security level 5, the TLS performance drops even further, except for the Saber and CRYSTALS-KYBER variants. Best results have been measured for the CRYSTALS-KYBER / FALCON combination yielding 890 connections per second or Saber / FALCON yielding even 920 connections per second.

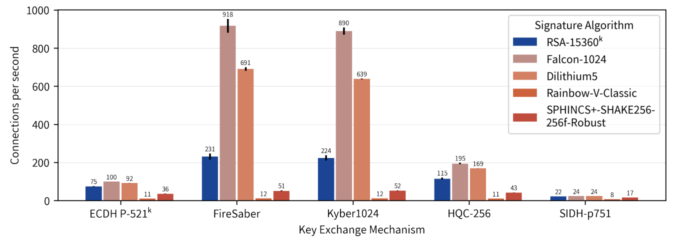
The reason for the great difference in performance is the mathematical foundation of the algorithms but also how efficient their implementation is. Since this is a practical analysis of algorithms available today as they are realized in the liboqs [13] library, we conclude that Saber or CRYSTALS-KYBER for key exchange together with the FALCON signature scheme seems to be a performant choice for further prototyping with Post-Quantum Cryptography.



(a) Security level 1



(b) Security level 3



(c) Security level 5

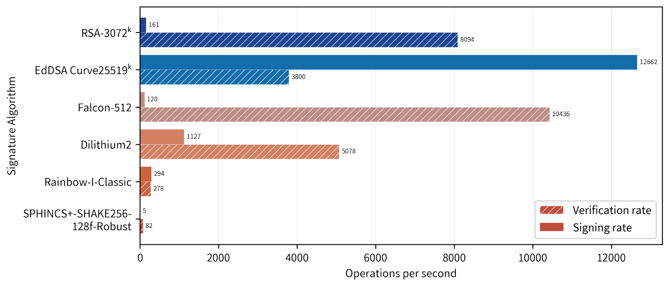
Fig. 1. TLS-1.3 handshake performance

B. Signature Algorithm Performance

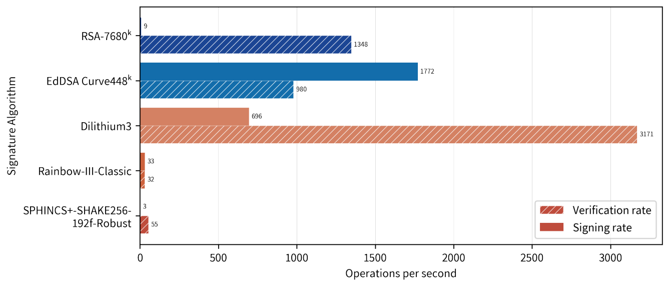
In contrast to section IV-A, the following measurements only focus on signature algorithms to make them comparable. Fig. 2 shows the raw algorithmic performance of signature algorithms for different security levels measured with the tools OpenSSL provides (openssl speed). The performance is measured in operations per second. The signature algorithm performance measurements show the following results:

- For security level 1, FALCON has the highest verification rate with up to 10000 verifications per second, followed by RSA and CRYSTALS-DILITHIUM. Rainbow and SPHINCS+ are the least performant with only 280 and 80 verifications per second respectively.
- For security level 3, the verification rate decreases drastically. As there is no FALCON variant for level 3, the best results are achieved by CRYSTALS-DILITHIUM with about 3000 verifications per second followed by RSA and EdDSA. Rainbow and SPHINCS+ perform the worst.
- For security level 5, the decrease in overall performance is not as noticeable for PQC but RSA suffers from an increased security level. Once again, FALCON

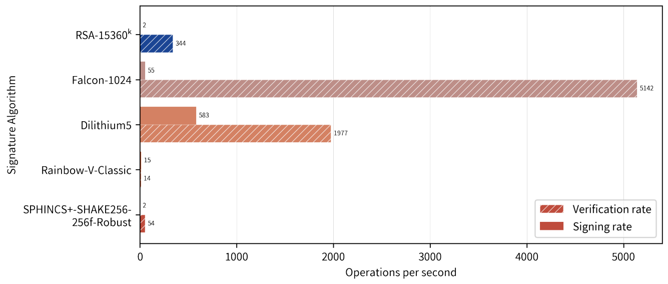
is the winner outperforming even the level 3 variant of CRYSTALS-DILITHIUM. Although Rainbow and SPHINCS+ do not suffer as much from increasing the security level, the performance is still far worse.



(a) Security level 1



(b) Security level 3



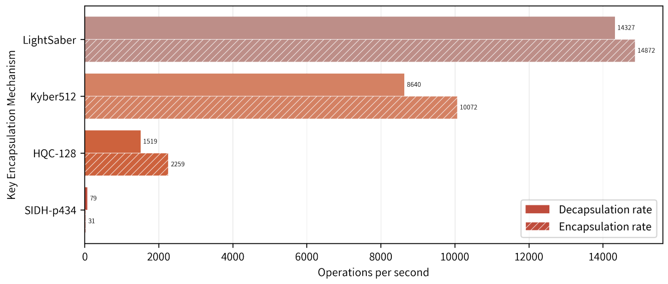
(c) Security level 5

Fig. 2. Signature Algorithm performance

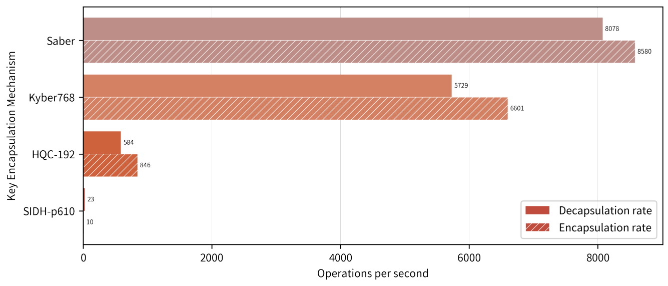
C. Key Encapsulation Mechanism Performance

Like section IV-B, the subsequent measurements focus only on KEMs. Fig. 3 shows the raw algorithmic performance of Key Encapsulation Mechanisms for different security levels measured with openssl speed. The performance is measured in operations per second. The raw performance of KEMs tested in this scenario cannot be directly compared to RSA and ECC-based key exchange schemes because they derive the shared keys differently.
The KEM performance measurements show the following results:

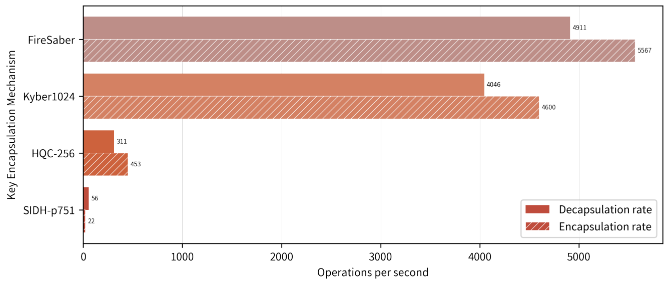
- Across all security levels, Saber has the highest decapsulation and encapsulation rate followed by CRYSTALS-KYBER and HQC. The slowest KEM is SIKE which only achieves a fraction of the rate of the best performing KEMs.



(a) Security level 1



(b) Security level 3



(c) Security level 5

Fig. 3. Key Encapsulation Mechanism performance

V. Conclusion

The results show that especially lattice-based cryptography can be compared to RSA and ECC, outperforming these classical schemes at higher security levels. Cryptography which is based on other mathematical problems seems to be significantly slower for the tested scenario.

This leads us to the conclusion that Post-Quantum Cryptography can replace classical asymmetric mechanisms used for web browsing, e-mail communication, and other applications without reducing the speed significantly. It must be said, however, that the measurements shown have been gathered with the implementation of the liboqs library (v0.7.1). We cannot judge if any other more performant implementations are available that contradict the results shown in this analysis. It is advised to repeat this study regularly.

For future analysis, one could consider other factors, like power consumption. A mapping of the obtained results to the underlying mathematical theory by analyzing computational algorithm complexity could also be very interesting.

References

[1] P. W. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,” *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[2] C. Gidney and M. Ekerå, “How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits,” *Quantum*, vol. 5, p. 433, 2021.

[3] D. J. Bernstein and T. Lange, “Post-quantum cryptography,” *Nature*, vol. 549, no. 7671, pp. 188–194, 2017.

[4] E. Rescorla and T. Dierks, “The transport layer security (tls) protocol version 1.3,” 2018.

[5] M. Raavi, S. Wuthier, P. Chandramouli, Y. Balytskyi, X. Zhou, and S.-Y. Chang, “Security comparisons and performance analyses of post-quantum signature algorithms,” in *International Conference on Applied Cryptography and Network Security*. Springer, 2021, pp. 424–447.

[6] V. Valyukh, “Performance and comparison of post-quantum cryptographic algorithms,” 2017.

[7] K. Basu, D. Soni, M. Nabeel, and R. Karri, “Nist post-quantum cryptography-a hardware evaluation study.” *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 47, 2019.

[8] D. Sikeridis, P. Kampanakis, and M. Devetsikiotis, “Post-quantum authentication in tls 1.3: A performance study,” Cryptology ePrint Archive, Report 2020/071, 2020. [Online]. Available: https://ia.cr/2020/071

[9] T. George, J. Li, A. P. Fournaris, R. K. Zhao, A. Sakzad, and R. Steinfeld, “Performance evaluation of post-quantum tls 1.3 on embedded systems,” Cryptology ePrint Archive, Report 2021/1553, 2021, https://ia.cr/2021/1553.

[10] G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, J. Kelsey, Y.-K. Liu, C. Miller, D. Moody, R. Peralta *et al.*, “Status report on the second round of the nist post-quantum cryptography standardization process,” *US Department of Commerce, NIST*, 2020.

[11] D. Moody, “Let’s get ready to rumble. the nist pqc competition,” in *Proc. of First PQC Standardization Conference*, 2018, pp. 11–13.

[12] T. Fernández-Caramés, P. Fraga-Lamas, and M. Suárez-Albela, “A practical evaluation on rsa and ecc-based cipher suites for iot high-security energy-efficient fog and mist computing devices,” *Sensors*, vol. 18, p. 3868, 11 2018.

[13] The Open Quantum Safe Project, “liboqs - an open source c library for quantum-safe cryptographic algorithms,” 2022. [Online]. Available: https://github.com/open-quantum-safe/liboqs

[14] R. Döring, “Testing setup and measurement procedure,” 2022. [Online]. Available: https://gitlab.com/3v3ryb0dy/praxisprojekt-notebook

[15] The Open Quantum Safe Project, “Software for prototyping quantum-resistant cryptography,” 2022. [Online]. Available: https://openquantumsafe.org/