UR-089 Performance Analysis of PQC KEM Algorithms

Abstract

Within the next twenty years or so experts predict that we will have quantum computers which will make certain kinds of encryption that we rely on ineffective and vulnerable to malicious entities. Post quantum computing (PQC) algorithms fill in that security gap that classical encryption algorithms can not. A particular category of PQC algorithms are key exchange mechanism (KEM) algorithm. The goal of these algorithms is to securely generate a shared symmetric key which can be used for encrypting future communication between the hosts. An important use case for these algorithms is in securing the Transport Layer Security protocol (TLS) against quantum adversaries. Due to the widespread use of TLS, it is critical that any new standard use PQC algorithms which are both efficient and secure. To this end we test each of the PQC KEM algorithms provided by oqs-provider library to compare their performance impact on the TLS handshake.

PQC Algorithms

Post-Quantum Cryptographic algorithms are algorithms that are safe from attacks by quantum computers. These PQC algorithms are tested in rounds by NIST with the current round being the 4th. In figure 1, the security levels presented by NIST can be observed. The algorithms that are round 4 candidates are presented in figure 2. By implementing these algorithms, the goal is to find the most optimized and efficient PQC algorithm.

Classical and quantum security for NIST's levels

NIST level		Classical	Quantum
AES-128	(L1)	128	64
SHA3-256	(L2)	128	85
AES-192	(L3)	192	96
SHA3-384	(L4)	192	128
AES-256	(L5)	256	128

Figure 1. NIST level security

	HQC	BIKE	Kyber	ML	Frodo
• Public key size(bytes)	HQC-128: 2249 HQC-192: 4522 HQC-256: 7245	BIKE-L1: 1541 BIKE-L3: 3083 BIKE-L5: 5122	Kyber-512: 800 Kyber-768: 1184 Kyber-1024: 1568	ML-512: 800 ML-768: 1184 ML-1024: 1568	Frodo-640: 9616 Frodo-976: 15632 Frodo-1344: 21520
• Secret Key (bytes)	HQC-128: 2289 HQC-192: 4562 HQC-256: 7285	BIKE-L1: 5223 BIKE-L3: 10105 BIKE-L5: 16494	Kyber-512: 1632 Kyber-768: 2400 Kyber-1024: 3168	ML-512: 1632 ML-768: 2400 ML-1024: 3168	Frodo-640: 19888 Frodo-976: 31296 Frodo-1344: 43088
• Ciphertext size (bytes)	HQC-128: 4481 HQC-192: 9026 HQC-256: 14469	BIKE-L1: 1573 BIKE-L3: 3115 BIKE-L5: 5154	Kyber-512: 768 Kyber-768: 1088 Kyber-1024: 1568	ML-512: 768 ML-768: 1088 ML-1024: 1568	Frodo-640: 9720 Frodo-976: 15744 Frodo-1344: 21632

Figure 2. NIST level 4 candidates

The algorithms from figure 2 that are being implemented include: HQC a Hamming Quasi-Cyclic approach, BIKE a Quasi-Cyclic Moderate Density Parity-Check, Kyber a LWE problem over modular lattices, ML a LWE problem over modular lattice, and Frodo a LWE problem. Each of these in the figure are associated with a public key, secret key, and ciphertext size for each version.

Key Words:
NIST (National Institute of Standards and Technology)
LWE (Learning with Errors)

System design

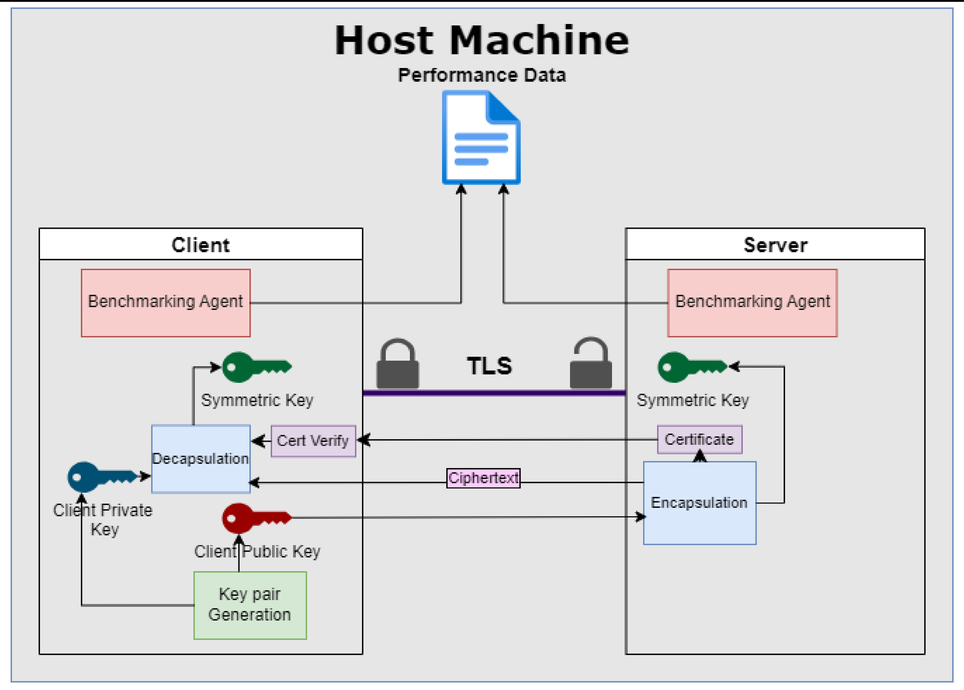


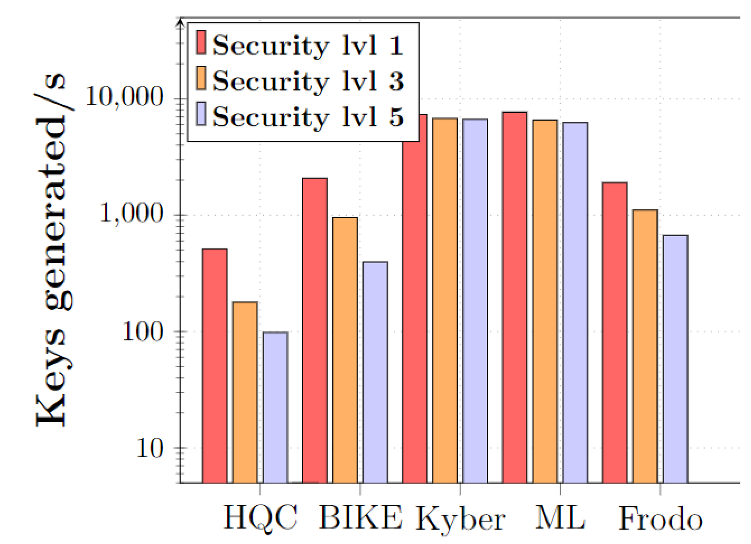
Figure 2. Overview of TLS1.3 handshake experiment

System Design Continued

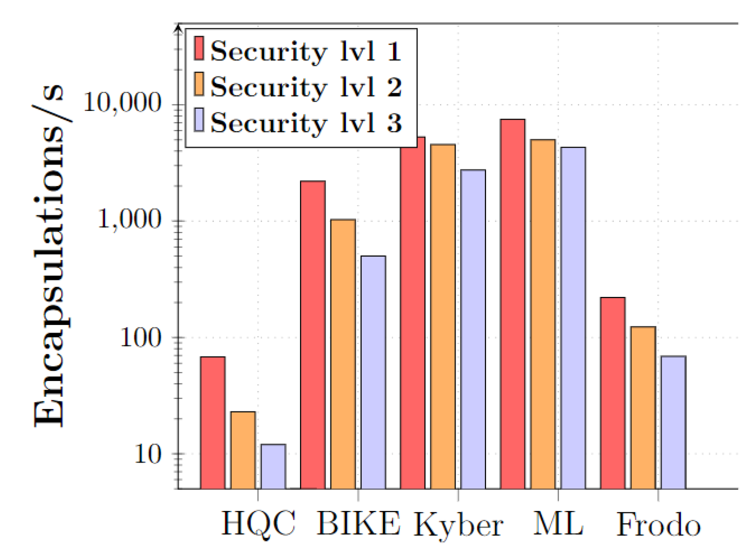
We implement our performance benchmarking using the Openssl and liboqs libraries. Openssl is a C library which provides an implementation of TLS while liboqs allows us to integrate the PQC algorithms into OpenSSL. The design of this protocol can be seen in Figure 2, the TLS handshake uses KEM algorithms as part of three operations. The first is key generation, this is done by the client which generates the KEM key pair and sends the public key to the server which uses as part of the next operation, encapsulation. The server uses the public key to encrypt the AES key data and send it back to the client. The client performs decapsulation which decrypts the ciphertext to obtain the shared symmetric key.

Our experiment runs two programs, a client and a server which conduct a TLS 1.3 handshake and gather data on the key generation, encapsulation, and decapsulation. We also gather data on the overall performance of the TLS handshake under each algorithm. We test each algorithm at all 3 of its security levels

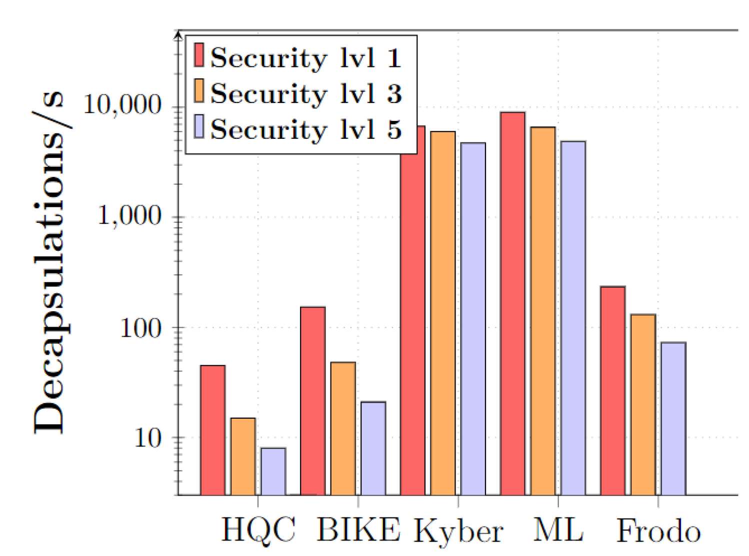
Results



A. Key Generation.

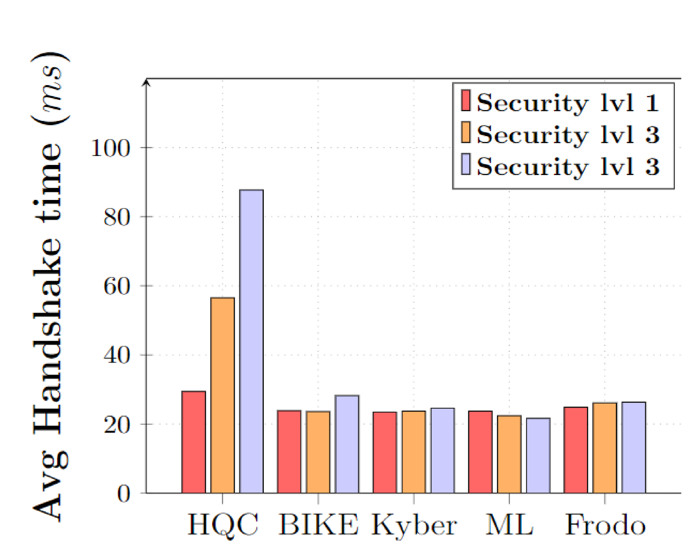


B. Encapsulation.

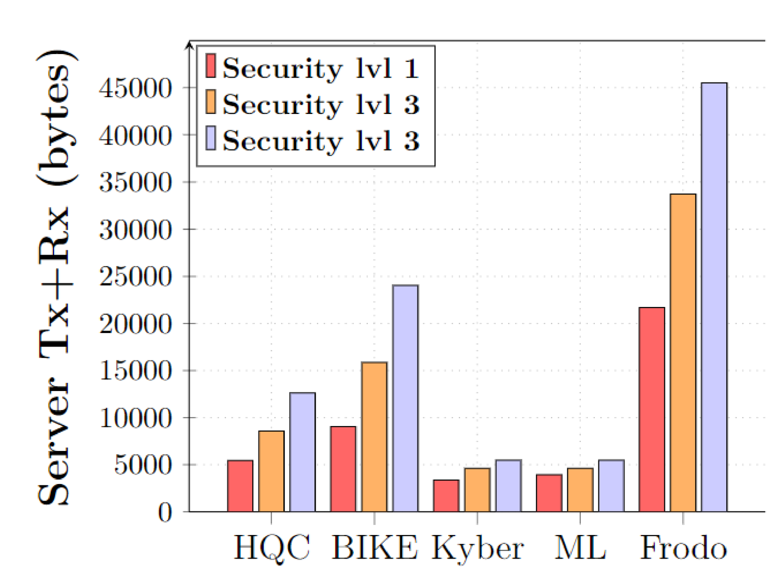


C. Decapsulation.

Figure 4. the number of cryptographic operations per second each tested PQC algorithm can achieve at different levels of security



A. Handshake.



B. Bandwidth Usage.

Figure 5. Performance of TLS Handshake under different PQC algorithms

Analysis

Figure 4 shows the individual performance of each tested PQC KEM algorithm's cryptographic operations on a logarithmic scale. Overall, Kyber and MLKEM see the best performance with the fastest key generation, encapsulation, and decapsulation. HQC performed worse in all categories.

Figure 5 shows the performance of the TLS handshake with 5a measuring the handshake duration and 5b measuring the amount of data that needed to be sent and received for the handshake. Each of the algorithms saw relatively comparable performance for handshake duration, besides HQC which saw performance degradation at higher security levels. As for bandwidth usage, the Kyber and MLKEM algorithms once again saw the best performance while FrodoKEM saw the worst. While each of these handshake sizes is small (5-50kb), The large number of handshakes a service like a web server performs could affect network traffic levels.

Conclusions

Our work measures the performance of the round 4 PQC KEM algorithm submissions, as well as the impact each algorithm had on the performance of the TLS handshake. Overall, the Kyber and MLKEM algorithms seem like the best choice performance wise for TLS. Each of the other algorithms had some area of weakness. Particularly, the more data-intensive algorithms will likely see worse performance under poor network conditions. Future work could include testing hybrid KEM algorithms which try to use the different strengths of KEM algorithms to make the process more efficient.

Acknowledgments

We'd like to thank our faculty advisor, Dr. Manohar Raavi for all his help on this project. We'd also like to thank our Senior Project Professor Sharon Perry for helping us push across the finish line.

Contact Information

- Dillon Horton - dhorto23@students.kennesaw.edu
- Gage Standard - tstanda2@students.kennesaw.edu
- Jose Gutierrez – jgutie14@students.kennesaw.edu

References

- D. Herman, C. Googin, X. Liu, Y. Sun, A. Galda, I. Safro, M. Pistoia, and Y. Alexeev, "Quantum computing for finance," Nature Reviews Physics, vol. 5, no. 8, p. 450-465, Jul. 2023.|Online|. Available: http://dx.doi.org/10/1038/s42254-023-00603-1
- OpenSSL Foundation, Inc. "OpenSSL.", www.openssl.org/.
- Open-Quantum-Safe. "Open-Quantum-Safe/OQS-Provider: Openssl 3 Provider Containing Post-Quantum Algorithms." *GitHub*, github.com/open-quantum-safe/oqs-provider.
- P. Schwabe D. Stebila, and T. Wiggers, "Post-quantum tls without handshake signatures," Cryptology ePrint Archive, Paper 2020/534, 2020, https://eprint.iacr.org/2020/534. [Online]. Avaliable: https://eprint.iacr.org/2020/534
- R. P. Feynman, "Quantum mechanical computers," Found. Phys. Vol. 16, no. 6, pp. 507-531,Jun 1986.

KENNESAW STATE UNIVERSITY
COLLEGE OF COMPUTING AND SOFTWARE ENGINEERING

Dillon Horton, Gage Standard, Jose Gutierrez
Prof. Manohar Raavi