scientific reports

Check for updates

OPEN

In-line rate encrypted links using pre-shared post-quantum keys and DPUs

Abraham Cano Aguilera[1,2]✉, Carlos Rubio Garcia[1], Daniel Lawo[1,2], José Luis Imaña[3], Idelfonso Tafur Monroy[1] & Juan José Vegas Olmos[2]

The development of quantum computers poses a risk to the cryptographic algorithms utilized by current public key infrastructure (PKI). High performance computing (HPC), inter- and intra-datacenter communications, and governmental communications are particularly vulnerable to security attacks by quantum computers. In response, there are on-going efforts on implementing post-quantum cryptography (PQC) despite the considerable overhead involved in processing PQC encrypted data packets in comparison with their classical counterparts. This work aims to help with the transition to quantum resistant cryptographic schemes by means of providing a thorough experimental performance comparison of the four algorithms selected by the National Institute of Standards and Technology (NIST) for standardization when implemented in a data processing unit (DPU). The experiments are conducted at line-rate, demonstrating for the first time operation at almost 100 Gbit/s and hence suitability for high-capacity data center environments.

Post-quantum cryptography (PQC) is the field of study that focuses on developing cryptographic algorithms and protocols which are believed to be resistant against both classical and quantum attacks. Traditional cryptographic schemes such as Rivest Shamir Adleman (RSA)[1] or elliptic curve cryptography (ECC), e.g elliptic curve Diffie Hellman key exchange (ECDH)[2] or elliptic curve digital signature algorithm (ECDSA)[3], rely their security on the hardness on solving some specific mathematical problems such as the integer factorization or elliptic curve discrete logarithm problems.

However the expectation of powerful quantum computers in the foreseeable future, and their ability to break these computational problems[4,5], is compelling various entities including governmental bodies, cloud providers, and high performance data centers to migrate their public key infrastructure (PKI) to withstand quantum threats. Among the solutions providing quantum security, quantum key distribution (QKD)[6] and PQC are the ones receiving more attention by the research community. PQC holds a favorable position due to its seamless integration with existing infrastructure and its comprehensive cryptographic framework, including: key encapsulation mechanisms (KEM), digital signatures, encryption schemes, fully homomorphic encryption among many other functionalities, whereas QKD is limited to key exchanges (KEX). In August 2023, after extensive research, NIST finally provided four standard drafts featuring Kyber[7], Dilithium[8], Falcon and Sphincs+[9]. Kyber emerged as the only selected KEM chosen for standardization and it is designed to withstand quantum attacks through the hardness of solving the learning-with-errors (LWE)[10] problem. On the other hand, Dilithium, Falcon and Sphincs+ function as digital signatures, where the former two rely its security on the hardness of lattice problems and the latter bases its security on the hash hardness[11].

However, the integration of cryptography, including PQC, presents significant challenges due to its computational demands. Implementing PQC on conventional Central Processing Units (CPU) might be inefficient and consume a large amount of resources and energy. As an example of the emerging relevance, in February 2024, Apple introduced PQ3[12] in its iMessage application. PQ3 employs Kyber to encapsulate symmetric keys. The large fabric of iMessage users will create pressure on cloud-services, which will need to deal with the new PQ3 protocol at scale. Initial implementations of PQC often use Field-Programmable Gate Arrays (FPGAs) due to their flexibility and customizability. First implementations of Kyber and other PQ KEMs on FPGA[13,14], have shown that optimizing the scheduling of the binomial sampling, Number Theoretic Transform (NTT) operations and providing modifications of the Barrett reductions can lead to a speed up of x50 in comparison with other platforms at the cost of circuit size. There have also been works on implementing PQ signatures on FPGAs; Dilithium[15]

[1]Quantum & Terahertz Systems, Eindhoven University of Technology, 5600MB Eindhoven, The Netherlands. [2]NVIDIA Corporation, Yokneam Illit 2066730, Israel. [3]Department of Computer Architecture and Automation, Complutense University, 28040 Madrid, Spain. ✉email: a.c.a.cano.aguilera@tue.nl

, Dilithium+Falcon[16], and Sphincs+[17] where it has been manifested that when optimizing the sub-instructions of PQC like NTT, its inverse (INTT) or the Keccac hashing[18], it is possible to provide and improved throughput of 114 times in comparison with high-level synthesis languages. Nevertheless the implementation of PQC extends beyond high-performance architectures to microprocessors and constrained devices. The Neon-NTT project[19] provided an implementation of all Kyber, Dilithium and Saber optimized for ARM cortex M3 and A72 architectures providing speeds up of up to 2 times for the main subroutines of such algorithms, while providing new descriptions of Montgomery[20]/Barrett[21] reductions and NTT for the ARM instruction set. Similar works[22] have also focused on the implementation of new variants of NTT for Dilithium, obtaining performance boosts of around 20% in comparison with high-level synthesis languages. In addition to performance improvements, there has also been research on implementing masked versions of Kyber[23] with the aim of providing protection against side channel attacks. Although FPGAs and ASICs offer benefits in terms of performance, their utilization in practical communication systems is limited due to high production costs, power consumption concerns, and the CPU overhead incurred when offloading data to the processor in which they are integrated. Data processing units (DPUs) offer an alternative to such architectures by offloading specific tasks such as network processing and encryption, thereby enhancing system performance and reducing power consumption. In this work, we use DPUs to create our quantum safe communication link. This is achieved by employing specialized hardware like digital signal processors (DSP), semiconductor intellectual property (IP) blocks or graphics processing units (GPU). Both KEM and signatures are employed by various communication protocols, where Dilithium, Falcon and Sphincs+ are used initially for signing and verifying messages and Kyber facilitates key exchanges. These PQC keys exchanged by Kyber can then protect the data plane through the use of 256 bit-size key advanced encryption standard (AES)[24] or Rijndael[25] ciphers. It is of paramount importance the migration of protocols like internet protocol security (IPsec)[26,27] or transport layer security (TLS)[28] handshakes (Signature+Key exchange) to PQC. TLS has received much more attention in that regard with considerable works[29–31] highlighting the effects of PQ handshakes in terms of latency and the architectural changes needed for . IPsec, the focus of this work, has received less attention, yet there have been some efforts from Internet Engineering Task Force (IETF) providing drafts for standardization for using pre-shared PQ keys (RFC8784)[32] and allowing multiple key exchanges on IPsec (RFC9370)[33]. To the best of author's knowledge the contribution of this work presents:

- First characterization of all PQ algorithms considered by NIST for standardization on a single DPU.
- First PQ-IPsec link using SPHINCS+ for PQ authentication and using pre-shared keys exchanged by Kyber.

This paper is organized as follows: the main novelty of the paper is presented in the experiment section, where we define the methodology of PQC calls to be integrated in real communication protocols such as IPsec as well as its integration with DPU. Afterwards, experimental results are shown; these results include a comparison with classical cryptography in terms of signature sizes, CPU usage of KEMs and signatures, handshake throughput and finally encryption throughput. Finally, a summary of the main novelty of the paper is given in the discussion section.

Experiment

In order to evaluate the use of PQC in real world use cases we develop a PQ-testbed for running PQ communications between two independent data processing units. The setup, illustrated in Fig. 1, consists of two independent servers, each equipped with its own CPU, and two DPUs (model MBF2H516ACEEOT) capable of processing data at rate of 100 Gigabits (100G) per second. To connect the servers with the DPUs, Peripheral Component Interconnect Express (PCIe) bridges are employed. The DPUs are interconnected via optical pluggable that use Direct Attached Copper (DAC) cables. These DPUs feature ARMv8 A72 cores dedicated to specific operations. These 8 cores can operate at a clock rate between 550 and 2750 MHz. Both DPUs employ a Connect-X 6 smartNIC on-board with two dual QSFP56 optical pluggables which can operate with both DAC or optical fiber. Further details about the technical specification of the DPUs employed in this experiment can be found in[34]. With regard to Hardware Accelerators, DPUs employ non-programmable HW accelerators for classical public key cryptography, i.e ECDSA and ECDH[35], and for the authenticated encryption with associated data (AEAD) algorithm AES Galois Counter Mode (AES-GCM)[36]. The use of the HW accelerators is made through the deployment of an openssl engine, a low level interface to call the accelerated functions from openssl standard library code. In terms of software, we integrated a PQ pre-shared key scheme on top of the strongSwan Mellanox library which implements IPsec[37]. PKI methods, such as ECDSA, ECDH, Kyber, or Dilithium, are used primarily at the beginning of communication between parties, unlike symmetric cryptography. Therefore, achieving ultra-fast throughput is less critical for PKI compared to symmetric cryptography methods like AES, which are crucial for encrypting the data plane. Consequently, it's less important to accelerate PKI as it is for AES encryption. Based on this reasoning, we implemented all PQC core functionalities on ARM cores using intrinsic functions (Single Instruction, Multiple Data). These functions can optimize C code performance by more than 50 %, as demonstrated in Fig. 6.

Internet protocol security (IPsec) and its key exchange (IKE). IPsec protocol aims to provide encryption, authentication and integrity protection at the network layer. IPsec main use case is to secure virtual private networks (VPNs) and to secure communication between network interfaces through insecure channels as the internet. IKE is composed by two phases, in phase one the control plane is secured. Right after, the second phase is used to protect the data plane sending encrypted data with a symmetric cipher. Both phases use PKI stack and state of the art relies its security exclusively in classical security, thus, levering it weaker adversaries with quantum advantage.For this purpose, we utilized the reference code provided by CRYSTALS to establish a PQ link. In this link, we employ Kyber for encapsulating a PQ key for encryption, and either Dilithium, Falcon, or

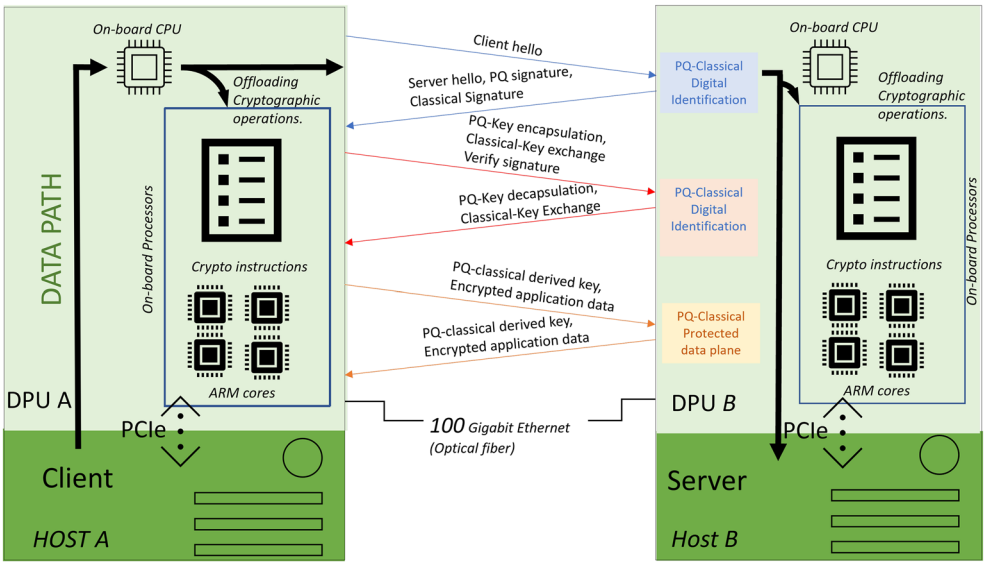


Fig. 1. Physical experimental set-up. Client initiates a quantum-resilient link using both PQ and classical algorithms to verify the certificates generated by the server. Upon identity confirmation, a key-exchange protocol ensues, establishing an AES-256 key for encrypted data transmission between client and server.

Sphincs+ for authentication. After the keys are exchanged, they can be blended with the original classical keys for encryption that were exchanged during the IKE protocol in IPsec, similar to RFC 8784[32]. The protocol (see Fig. 2) can be explained through the following steps:

1. Peers agree upon a security association, which includes the security levels and other relevant parameters.
2. An authenticated PQ KEX is executed between the peers to exchange a PQ pre-shared key (PPK) for encryption. This key is stored on both sides of the communication.
3. A classical KEX occurs between the peers to exchange a shared secret key (SSK) for encryption.
4. Classical authentication is performed between the peers to validate the server receiving the information.
5. Classical and PQ keys are combined to derive a new session key (SK). This mixed key is then used to encrypt the data plane with AES-256.

The PQ-IPsec protocol differs from the traditional protocol in two main aspects (see Fig. 3). The first, and most apparent, difference is the incorporation of PQ (post-quantum) algorithms for both authentication and key exchange. The PQ key exchange is conducted offline as a pre-shared key, while the classical key exchange

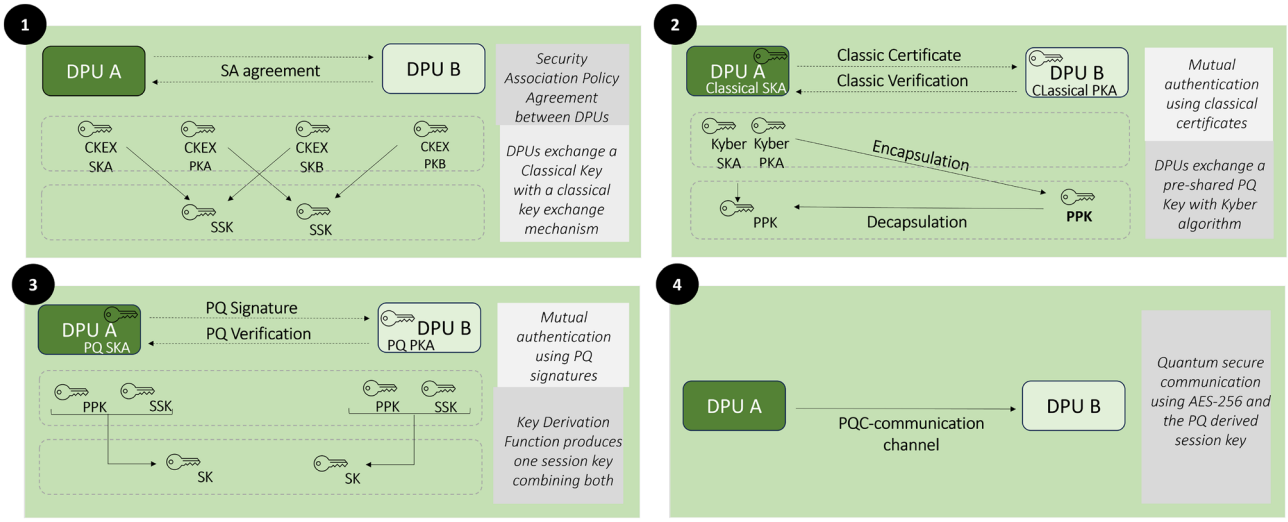
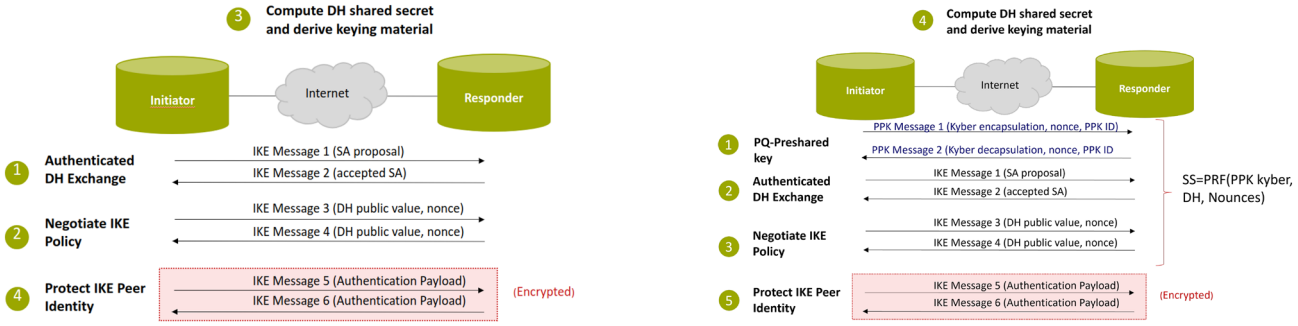


Fig. 2. Protocol for end-to-end PQ-IPsec establishment with pre-shared keys. Classical keys are exchanged within the IPsec handshake while PQ keys are exchanged offline.



(a) Classical IPsec protocol phase 1 **(b)** PQ IPsec protocol phase 1

Fig. 3. Comparison of classical and PQ IPsec protocol phases. Blue arrows show the part of the protocol where where a PQ key is exchanged offline and then integrated into the key mixing process of the IPsec protocol.

occurs within the IKE protocol. This approach ensures that the performance of the handshake remains unaffected by the PQC, although it does require the deployment of infrastructure for exchanging and storing the keys. These keys are subsequently utilized in the key expansion process within IPsec, in accordance with RFC 8784[32].The second difference is the integration of key materials for hybrid schemes. In this context, a Pseudo-Random Function (PRF)[38] is employed to establish a shared secret by deriving key material from the classical and PQ keys exchanged. PRFs are crucial in quantum-resistant communication because they ensure that the resulting key benefits from the security of at least one of the primitives, enabling the combination of various sources with minimal cost. In our experiments, we combined one PQ algorithm with one classical algorithm at a time. However, this approach can also be used to combine multiple PQ algorithms with classical cryptography or PQ keys with other technologies such as QKD[31]

The process for combining key material is straightforward[32]. First is computed the "SKEYSEED" by means of using a prf, the initiator's nonce (N_i) and the responder's nonce (N_r) and the classical material exchanged by ECDH (g^{ir}).

$$\text{SKEYSEED} = \text{prf}(N_i \mid N_r, g^{ir}) \tag{1}$$

Immediately following the use of SKEYSEED to calculate seven additional secrets: SK_d, which is used to derive new keys for the Child SAs established with the IKE SA; SK_{ai} and SK_{ar}, which serve as keys for the integrity protection algorithm to authenticate component messages in subsequent exchanges; SK_{ei} and SK_{er}, which are used for encrypting and decrypting all subsequent exchanges; and SK_{pi} and SK_{pr}, which are employed when generating an authentication payload.

$$\{SK_d \mid SK_{ai} \mid SK_{ar} \mid SK_{ei} \mid SK_{er} \mid SK_{pi} \mid SK_{pr}\} = \text{prf}^{+}(\text{SKEYSEED}, N_i \mid N_r \mid SPI_i \mid SPI_r) \tag{2}$$

Finally, SK_d, SK_pi, SK_pr are re expanded again through a prf+ using the PQ pre-shared material:

$$\begin{aligned} SK'_d &= prf + (PPK_{kyber}, SK_d) \\ SK'_{pi} &= prf + (PPK_{kyber}, SK_{pi}) \\ SK'_{pr} &= prf + (PPK_{kyber}, SK_{pr}) \end{aligned} \tag{3}$$

The most interesting property of this key mixing is that as long as one of the keys is secure for the PRF - The diffie hellman group or the pre-shared key with kyber the scheme will be secure, thus this construction is an hybrid cryptographic construction as per NIST recommnedations.

The other really important aspect of PQC schemes is the generation of random numbers. Even though there has been extensive research on the use of Quantum Random Number Generators (QRNG)[39] because their security is based in quantum physics and not mathematical relations, their most common methods - photon interaction with a beam splitter[40], and photon interference with random vacuum fluctuations[41] are still far from being deployed in real communication systems. Therefore, NIST[7,8] is enforcing the use of conventional Random Bit Generators that are secure against quantum and classical attacks and that must follow some security guarantees according to the recommendations for random bit generation NIST SP 800-90A[42] and NIST SP 800-90B[43]. In the case of the DPUs this source of randomness can be obtained by the arm cores by accessing getrandom system call interfaces behind /dev/random and /dev/urandom, which in all linux systems should be compliant with such standards.

Results

Running PQC algorithms as well as symmetric key cryptography for encrypting plaintext data might be computationally expensive and unaffordable for general purpose processors such as the CPUs involved in current data centers. In this section we show our experimental results when executing a PQ-IPsec channel using Kyber+ECDH for the Internet Key Exchange protocol as well as Dilithium, Falcon or Sphincs+ along with RSA or ECDSA for the authentication. First, we present an overview of the CPU overhead associated with executing each part of

Dilithium, Sphincs+ and Falcon. Subsequently, we present the throughput of executing all classical, PQ, and hybrid schemes for key exchange and authentication mechanisms. Finally, we demonstrate experimentally the feasibility of achieving line-rate encrypted communication in the dataplane within the IPsec protocol using DPUs. For each one of the later subsections we provide the metrics in terms of the security level. The security level of a cryptographic system is a metric that indicates the difficulty for an attacker to break a cryptographic primitive. Security levels are expressed in bits, where *n-bit* security means that an attacker would need 2^n operations to break the scheme. However, this definition can vary depending on the attacker model. In Table 1, we can see the security levels proposed by NIST for evaluating the post-quantum (PQ) algorithms chosen for standardization, with level 1 being the lowest security level and level 5 the highest. In the following sections, we provide all performance metrics in terms of these security levels to ensure fair comparisons of key and ciphertext sizes between classical and PQ cryptographic schemes. It is important to note that the security level of classical cryptography would be greatly diminished with the advent of quantum computers.

Signature sizes

One of the most important technical considerations when deploying cryptography is the size of public key sizes as well as digital signature sizes. The emission of large public keys can strain network bandwidth and the computational resources dedicated to it, thus making them less practical for real use-case scenarios. Similarly, signature sizes are also critical since they might affect the storage and transmission of digitally signed messages.

In Fig. 4, we can observe how the use of PQC introduces a large overhead with respect to the public key sizes actually employed by its classical counterparts. Specifically, Dilithium public key sizes are 20 times bigger than ECDSA in all its security level variants, while Falcon public key sizes are between 13 times (weakest security levels) to 19 times (hardest security levels) higher, exposing the main problem of lattice-based encryption, where larger public keys can stress the network when a client requests a server key to authenticate a peer in a connection.

In this regard, hash-based signatures offer a better memory trade-off since they outperform classical signatures by a factor of 33% to 50% between the highest security levels. However, the main disadvantage of hash-based signatures like SPHINCS+ is their huge signature sizes, which require sending between 356 and 375 more bytes through the handshake procedure compared to ECDSA. The effect of the signature size is weaker when using Falcon and Dilithium, with the former providing slightly better performance and having signatures between 10 and 14 times bigger than ECDSA.

CPU usage

Running cryptography in general and PQC in particular is computationally expensive and can overload general purpose processors such as servers handling connections in high performance computing data centers. Figure 5 illustrates the average cycles required for generating the keys, computing the signatures and computing the verification of each of the signature schemes considered for standardization by NIST, when running 10000 handshakes over our 100Gbps capacity point-to-point link. It is evident that Falcon imposes the heaviest computational

Level	Security description
I	At least as hard to break as AES128 (exhaustive key search)
II	At least as hard to break as SHA256 (collision search)
III	At least as hard to break as AES192 (exhaustive key search)
IV	At least as hard to break as SHA384 (collision search)
V	At least as hard to break as AES256 (exhaustive key search)

Table 1. Security levels.

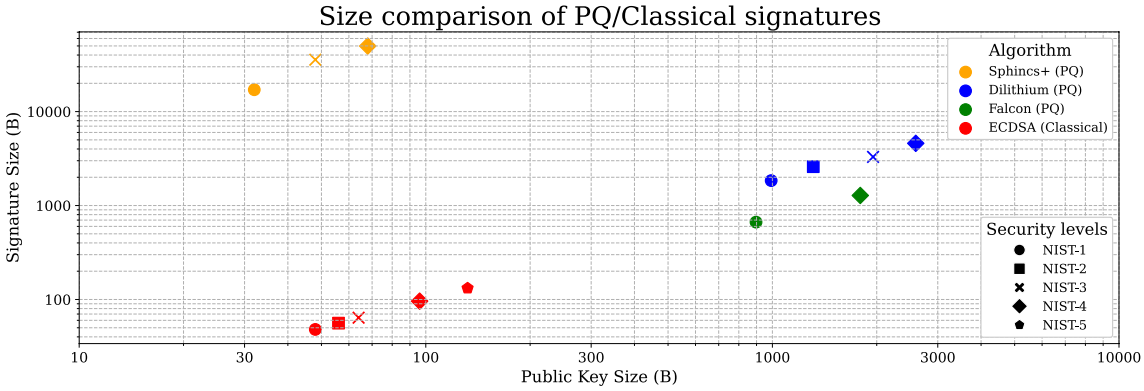


Fig. 4. Relationship between public key and digital signature sizes in bytes between classical cryptographic signatures (ECDSA), PQ-lattice-based cryptography (Dilithium and Falcon) and hash-based cryptography (Sphincs+).

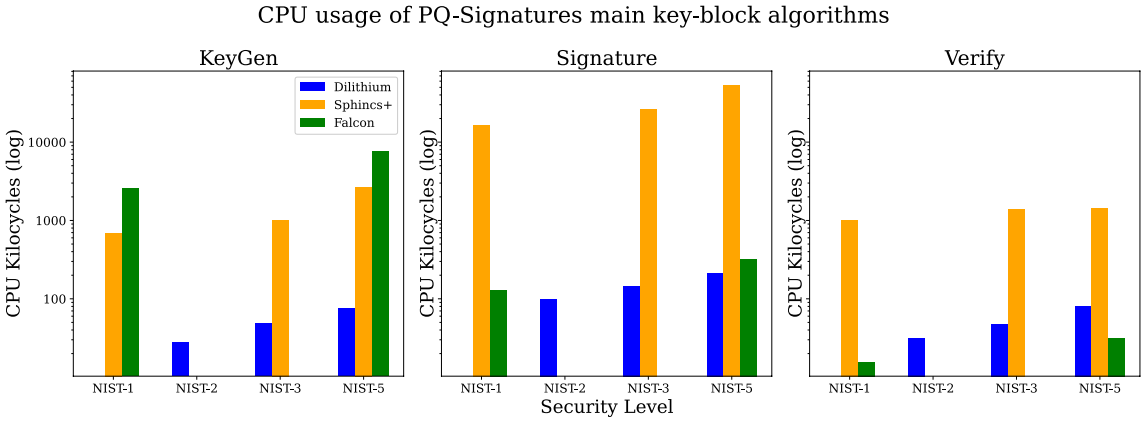


Fig. 5. CPU usage in Kylocycles of the main signature algorithms; KeyGen, signing and verifying from all the accepted PQ algorithms for standardization.

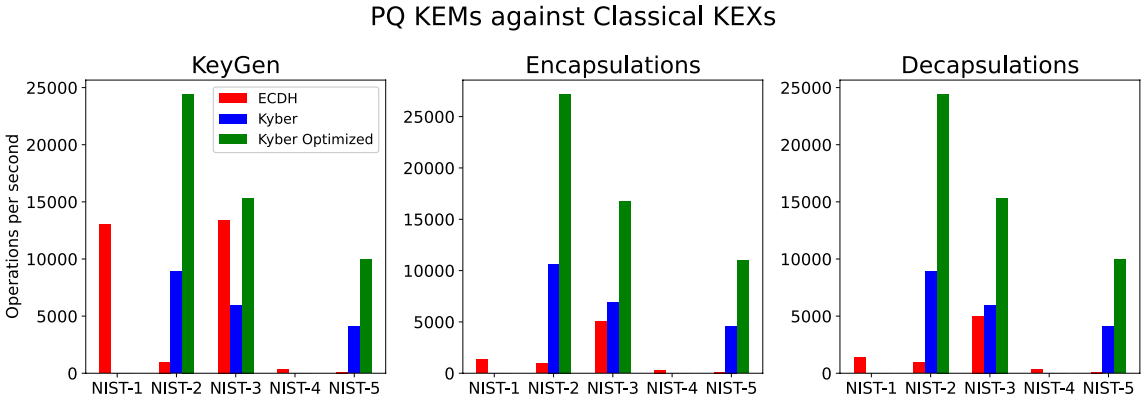


Fig. 6. Comparison of different classical and PQ key exchanges. We provide results for classical ECDH with the NIST recommended curves P-192 (NIST 1), P-224 (NIST 2), P-256 (NIST 3), P-384 (NIST 4) and P-512 (NIST 5) and the only PQ key accepted algorithm Kyber512(NIST-2), Kyber768(NIST-3) and Kyber1024 (NIST-5). For Kyber implementation we provide both the characterization for the standard (Kyber) and ARM optimized (Kyber_opt) libraries to show up the capabilities of DPUs to increase the performance in PQ handshakes.

efforts during the Key Generation process, exceeding Dilithium, the lightest PQ signature for generating keys, by two orders of magnitude. Computing Sphincs+ keys also involves significantly higher computational resources compared to Dilithium, yet it offers better performance than Falcon.

However, it is worth noting that the computation of keys for digital signatures is among the less critical steps steps in IPsec handshakes since they do not occur frequently; Certificate Authorities (CA) issue certificates with over one year of validity.

Regarding signing and verifying signatures, a notable trend emerges: Sphincs+ is the scheme that demands the most computational power for both functions by more than 3 orders of magnitude, making it unsuitable for resource-constrained environments or scenarios in which is necessary signing and verifying in real time. However, Sphincs+ might be suitable for scenarios with ample bandwidth and abundant computational resources while providing PQ security with one of the most research. The most suitable schemes for signing and verifying appear to be Falcon and Dilithium, exhibiting similar yet different performance characteristics. While Dilithium excels in signing CPU usage, it offers a slightly poorer performance in comparison with Falcon, which provides a lighter CPU usage during the verification step. This suggests that Dilithium is more suitable for applications like one-to-many communication in which the speed of the signer is more important than the speed of the parties verifying the signature, while Falcon might prove more useful in environments where fast verification is paramount, such as important transactions or user authentication systems.

Handshake throughput

In large-scale environments, such as high performance computing data centers, the management of a huge number of incoming requests to transmit data across the network per second is crucial. For each request to the network, an authenticated key exchange must happen as described in Fig. 1. In such environments it is of paramount importance provide the highest throughput as possible in order to handle as many requests as possible and do not saturate the network. Fig. 6 shows the results of our link for both classical and PQ key exchange. We offer the results for ECDH, standard Kyber and optimized Kyber. The optimized Kyber version[19] makes use of a combination of Barrett and Montgomery for computing the modular reduction of the polynomials

and interleaved multi-staged NTT for accelerating the multiplication of polynomials, the most cricital parallelizable step for lattice-based cryptography. Even though PQC has a worse memory footprint than its classical counterpart,lattice-based encryption provides a far more competitive performance than elliptic curves when deploying on the ARM cores of the DPU.

Notably, from all the curves proposed by NIST, only the curve P-256 has received enough consideration to be optimized by the cryptographic community. P-256 provides a good performance in terms of Key Generation, generating keys 2.2 times faster than the standard Kyber implementation. However, it still offers a worse performance in comparison with Kyber768 optimized. The results are even more pressing with regards to the encapsulation/decapsulation against the key exchange; the standard reference outperforms optimized ECDH by 15%. This improvement raises to 266% with ARM optimization on the DPU. Conversely, for other security levels, there has not been enough community effort in optimizing the curves, leading to a worse performance. Kyber, however, stands out, with Kyber512 optimized offering the most competitive performance, being capable of running up to 25,000 keys encapsulations/decapsulations per second, far exceeding the 5000 key exchanges performed by ECDH P-256. Kyber1024 optimized, the most secure key encapsulation mechanism, provides a throughput of 10000 encapsulations per second highlighting that PQ-cryptographic KEMs surpass even the most performance classic key exchanges in their maximum strength settings.

Figure 7 shows the relationship between the throughput achieved by the Classical/Hybrid/PQ signatures when deployed on the DPUs. Sphincs+ signatures provide the worst throughput due to inherent hash functions and substantial computational overload. This effect is more notorious in the hybrid setting, with Sphincs+ and RSA yielding the worst performance. Falcon, with its lowest security level, achieves the highest verification throughput among all signing algorithms while Dilithium variants offer the better trade-off between the signing/verifying throughput. Hybrid schemes, such as the combination of ECDSA and Dilithium or Falcon offer a competitive throughput, halfway of the most performance schemes, showing up that a slow migration to PQC can be done without introducing high latency to the network.

In-line encryption

Even though PQC is necessary for public key cryptography and thus is the key point of the communication handshakes which provide authentication and key exchange between peers, the most demanding function in every communication system is the encryption of the data plane. Figure 8 shows the throughput of our link under different configurations of the maximum transmision unit (MTU) sizes and threads used to send/receive data. For our experiment we consider two scenarios:

1. Transmission of data in clear. This is the fastest way of sending packets but does not provide any type of security to the transmission.
2. Transmission of data encrypted according to IPsec rules. This is the most secure way of sending packets but however it needs additional packet processing to make communications confidential.

The graph shows that there is a little penalty in terms of throughput when deploying the IPsec tunnel between the DPUs. This penalty is more evident as the number of cores and MTU sizes increase. This is due to the fact that while small packets can overload the network due to the processing of the headers, encryption might have an impact in longer chunks of data. We can see that DPUs can perform in exact line-rate in conditions in which MTUs are smaller, e.g noisy channels or without bit error connection and give excellent results in scenarios where large MTU sizes are allowed e.g when packet replay might not collapse the network with a 9% performance penalty in comparison with the data-on-clear approach and 15% of penalty with respect the capacity of the channel. In addition to this, it can be seen that DPUs cores can handle the traffic generated by the server since the performance of using more threads it does not affect significantly the performance of running the application with few threads.

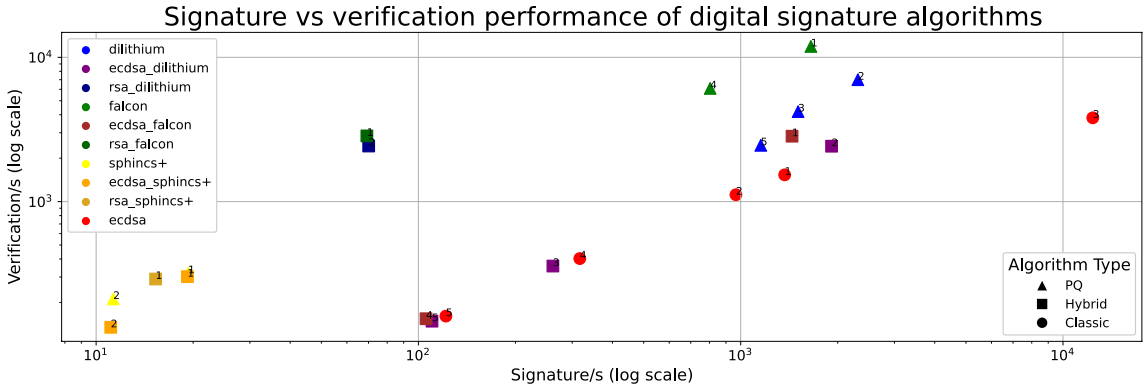


Fig. 7. Throughput of different signature algorithms in the PQ/hybrid setting and its associated security level. Shapes represent the type of algorithm; PQ, classical or hybrid, the colour the name of the algorithm/combination of algorithms and the number the security level of each combination.

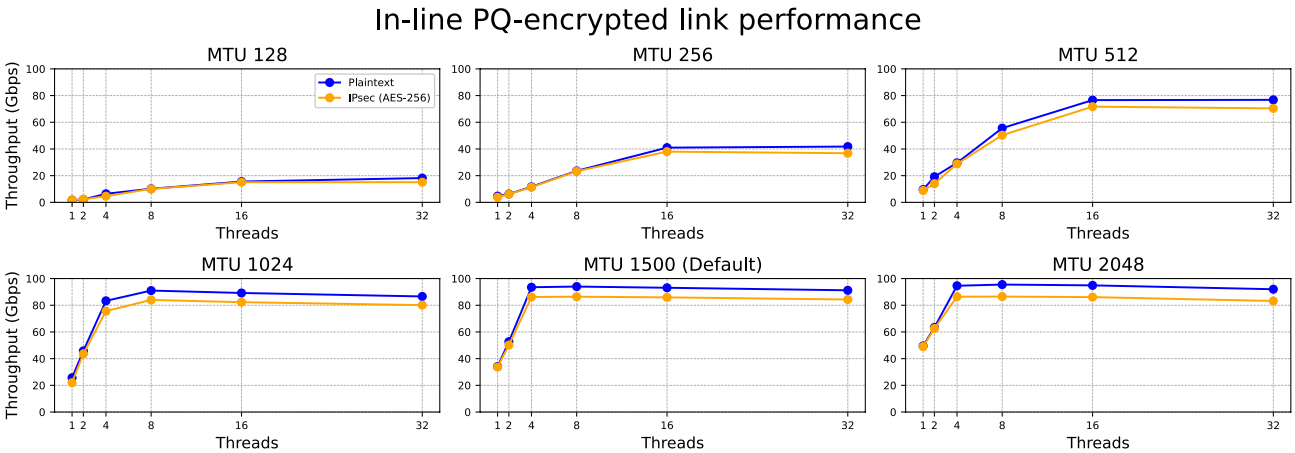


Fig. 8. AES-256 encrypted flow throughput based on different MTU sizes and threads. PQ KEM provide the 256 bits keys for encryption.

Conclusion

This work presented a groundbreaking advancement in inter-datacenter communication by providing a line-rate, end-to-end PQ encrypted link between two independent servers, evaluating for first time, all the PQ algorithms choosen for standardization by NIST on a DPU. DPUs are used for executing PQC operations offloaded by the server's CPU, thus, alleviating its computational resources. In our system architecture, DPUs are used for configuring an IPsec tunnel for confidential communication. This tunnel is initialized with hybrid cryptography, employing both classical and PQC. First, DPUs use classical certificates along with a classical key exchange. Subsequently, PQ authentication takes place, and then a pre-shared key is established in both sides through Kyber. Once the control plane is secured, both servers can encrypt the data plane by combining classical keys and the PQ pre-shared keys in accordance with RFC8784.

Through our experiments, we have identified suitable PQC algorithms for key encapsulation and signatures within various network scenarios. Sphincs+ is computationally heavy and its steps are slow in comparison with other signature algorithms, and thus its viability relies on environments with large bandwith and lots of computational resources available, and where the communications parties agree on relying on well established security assumptions (hashes) instead of newer security assumptions like lattice-based encryption. Dilithium and Falcon show comparable performance, with Dilithium being more adequate for fast signature applications and falcon being better when fast verification is needed.

Finally, our experiments have shown that DPUs can handle nearly perfect in-line rate traffic with small MTUs (<1024 bytes). Although increasing the MTU might slightly decrease the throughput by a factor of 10%, the offloading of computational tasks from the main server CPU makes this effect manageable.

Methods

All the methods required to reproduce our experiment and replicate results are explained in the experiment section. For further information we refer to the mail of the first author of this manuscript.

Data availibility

The datasets produced and analyzed in this work work for evaluating post-quantum cryptography algorithms on data processing units are available from the corresponding author on reasonable request.

Received: 6 March 2024; Accepted: 2 September 2024
Published online: 11 September 2024

References

1. Rivest, R. L., Shamir, A. & Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **21**, 120–126. https://doi.org/10.1145/359340.359342 (1978).
2. Barker, E., Chen, L., Roginsky, A., Vassilev, A. & Davis, R. *Recommendation for Pair-wise Key-Establishment Schemes Using Discrete Logarithm Cryptography*. https://doi.org/10.6028/NIST.SP.800-56Ar3 (2018).
3. Chen, L., Moody, D., Regenscheid, A. & Robinson, A. *Digital Signature Standard (DSS)*. https://doi.org/10.6028/NIST.FIPS.186-5 (2023).
4. Grover, L. K. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96. 212–219 https://doi.org/10.1145/237814.237866 (Association for Computing Machinery, 1996).
5. Shor, P. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*. 124–134 https://doi.org/10.1109/SFCS.1994.365700 (1994).
6. Bennett, C. H. & Brassard, G. Quantum cryptography: Public key distribution and coin tossing. *Theor. Comput. Sci.* **560**, 7–11 https://doi.org/10.1016/j.tcs.2014.05.025 (2014) (theoretical aspects of quantum cryptography—celebrating 30 years of BB84).
7. National Institute of Standards and Technology. Module-lattice-based key encapsulation mechanism standard. In *Federal Information Processing Standards Publication (FIPS) NIST FIPS 203 IPD* (Department of Commerce, 2023).

8. National Institute of Standards and Technology. Module-lattice-based digital signature standard. In *Federal Information Processing Standards Publication (FIPS) NIST FIPS 204 IPD* (Department of Commerce, 2023).
9. National Institute of Standards and Technology. Stateless hash-based digital signature standard. In *Federal Information Processing Standards Publication (FIPS) NIST FIPS 205 IPD* (Department of Commerce, 2023).
10. Lyubashevsky, V., Peikert, C. & Regev, O. On ideal lattices and learning with errors over rings. *J. ACM*[SPACE]https://doi.org/10.1145/2535925 (2013).
11. Lange, T. *Hash-Based Signatures* 540–542 (Springer, 2011).
12. imessage with pq3: The new state of the art in quantum-secure messaging at scale. https://security.apple.com/blog/imessage-pq3/. Accessed 01 Mar 2024 (2024).
13. Xing, Y. & Li, S. A compact hardware implementation of CCA-secure key exchange mechanism crystals-Kyber on FPGA. *IACR Trans. Cryptogr. Hardware Embedd. Syst.* **2021**, 328–356 https://doi.org/10.46586/tches.v2021.i2.328-356 (2021).
14. Dang, V. B., Mohajerani, K. & Gaj, K. High-speed hardware architectures and FPGA benchmarking of crystals-Kyber, NTRU, and saber. *IEEE Trans. Comput.* **72**, 306–320. https://doi.org/10.1109/TC.2022.3222954 (2023).
15. Ricci, S. *et al.* Implementing crystals-dilithium signature scheme on FPGAS. In *Proceedings of the 16th International Conference on Availability, Reliability and Security*, ARES '21 https://doi.org/10.1145/3465481.3465756 (Association for Computing Machinery, 2021).
16. Beckwith, L., Nguyen, D. T. & Gaj, K. High-performance hardware implementation of crystals-dilithium. In *2021 International Conference on Field-Programmable Technology (ICFPT)*. 1–10 https://doi.org/10.1109/ICFPT52863.2021.9609917 (2021).
17. Amiet, D., Leuenberger, L., Curiger, A. & Zbinden, P. FPGA-based Sphincs+ implementations: Mind the glitch. In *2020 23rd Euromicro Conference on Digital System Design (DSD)*. 229–237 https://doi.org/10.1109/DSD51259.2020.00046 (2020).
18. Dworkin, M. Sha-3 standard: Permutation-based hash and extendable-output functions. https://doi.org/10.6028/NIST.FIPS.202 (2015).
19. Becker, H., Hwang, V., Kannwischer, M. J., Yang, B.-Y. & Yang, S.-Y. Neon NTT: Faster dilithium, kyber, and saber on cortex-a72 and apple m1. *IACR Trans. Cryptogr. Hardware Embedd. Syst.* **2022**, 221–244 https://doi.org/10.46586/tches.v2022.i1.221-244 (2021).
20. Montgomery, P. L. Modular multiplication without trial division. *Math. Comput.* **44**, 519–521 (1985).
21. Barrett, P. Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor. In *Advances in Cryptology—CRYPTO' 86* (ed. Odlyzko, A. M.). 311–323 (Springer, 1987).
22. Greconici, D. O. C., Kannwischer, M. J. & Sprenkels, A. Compact dilithium implementations on cortex-m3 and cortex-m4. *IACR Trans. Cryptogr. Hardware Embedd. Syst.* **2021**, 1–24 https://doi.org/10.46586/tches.v2021.i1.1-24 (2020).
23. Heinz, D. et al. First-order masked kyber on arm cortex-m4. *Cryptology* ePrint archive, paper 2022/058. https://eprint.iacr.org/2022/058 (2022).
24. Dworkin, M. et al. *Advanced Encryption Standard (AES)*[SPACE]https://doi.org/10.6028/NIST.FIPS.197 (2001).
25. Daemen, J. & Rijmen, V. The block cipher Rijndael. In *Smart Card Research and Applications* (Quisquater, J.-J. & Schneier, B. eds.). 277–284 https://doi.org/10.1007/978-3-662-04722-4 (Springer, 2000).
26. Seo, K. & Kent, S. *Security Architecture for the Internet Protocol*. RFC 4301. https://doi.org/10.17487/RFC4301 (2005).
27. Housley, R. *Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)*. *RFC 4309*. https://doi.org/10.17487/RFC4309 (2005).
28. Rescorla, E. The transport layer security (TLS) protocol version 1.3. In *Request for Comments, Internet Engineering Task Force*. https://doi.org/10.17487/RFC8446 (2018).
29. Sosnowski, M. et al. The performance of post-quantum tls 1.3. In *Companion of the 19th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT 2023. 19–27. https://doi.org/10.1145/3624354.3630585 (Association for Computing Machinery, 2023).
30. Sikeridis, D., Kampanakis, P. & Devetsikiotis, M. Assessing the overhead of post-quantum cryptography in tls 1.3 and ssh. In *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '20. 149–156 https://doi.org/10.1145/3386367.3431305 (Association for Computing Machinery, 2020).
31. Rubio Garcia, C. *et al.* Quantum-resistant transport layer security. *Comput. Commun.* **213**, 345–358. https://doi.org/10.1016/j.comcom.2023.11.010 (2024).
32. Fluhrer, S., Kampanakis, P., McGrew, D. & Smyslov, V. *Mixing Preshared Keys in the Internet Key Exchange Protocol Version 2 (IKEv2) for Post-quantum Security*. RFC 8784 https://doi.org/10.17487/RFC8784 (2020).
33. Tjhai, C. et al. *Multiple Key Exchanges in the Internet Key Exchange Protocol Version 2 (IKEv2)*. RFC 9370 https://doi.org/10.17487/RFC9370 (2023).
34. *Nvidia Bluefield-2 ethernet DPU user guide*. https://docs.nvidia.com/networking/display/bluefield2dpuenug. Accessed 19 June 2024 (2024).
35. Rambus: High speed public key accelerator. https://www.rambus.com/security/protocol-engines/high-speed-public-key-accelerator/. Accessed 19 June 2024 (2024).
36. Rambus: Aes-ip-61 / eip-61 high speed low latency aes-gcm pipeline, 100gbps. https://www.rambus.com/security/crypto-accelerator-cores/aes-ip-61/.
37. Mellanox. *Mellanox Strongswan*. https://github.com/Mellanox/strongswan/tree/5.9.6bfRelease (2022).
38. Krawczyk, D. H. & Eronen, P. *HMAC-Based Extract-and-Expand Key Derivation Function* (HKDF). RFC 5869. https://doi.org/10.17487/RFC5869 (2010).
39. Herrero-Collantes, M. & Garcia-Escartin, J. C. Quantum random number generators. *Rev. Mod. Phys.*[SPACE]https://doi.org/10.1103/revmodphys.89.015004 (2017).
40. Kwon, O., Cho, Y.-W. & Kim, Y.-H. Quantum random number generator using photon-number path entanglement. *Appl. Opt.* **48**, 1774–1778. https://doi.org/10.1364/AO.48.001774 (2009).
41. Bruynsteen, C., Gehring, T., Lupo, C., Bauwelinck, J. & Yin, X. 100-gbit/s integrated quantum random number generator based on vacuum fluctuations. *PRX Quantum* **4**, 010330. https://doi.org/10.1103/PRXQuantum.4.010330 (2023).
42. Barker, E. B. & Kelsey, J. M. *Recommendation for Random Number Generation Using Deterministic Random Bit Generators* (Special Publication (NIST SP), National Institute of Standards and Technology, 2015).
43. Turan, M. S. *et al. Recommendation for the entropy sources used for random bit generation* (Special Publication (NIST SP), National Institute of Standards and Technology, 2018).

Acknowledgements

This work was partly funded by the QUARC project by the European Union Horizon Europe research and innovation program within the framework of Marie Sk?odowska-Curie Actions with grant number 101073355, and by the grant PID2021-123041OB-I00 funded by MCIN/AEI/10.13039/501100011033 and by "ERDF A way of making Europe".

Author contributions

A.C.A conceived the experiment, analysed the results and reviewed the manuscript. C.R.G analysed the results and reviewed the manuscript. D.L analysed the results and reviewed the manuscript. J.L.I analysed the results and reviewed the manuscript. I.T.M analysed the results and reviewed the manuscript. J.J.V.O analysed the results and reviewed the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to A.C.A.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.