Towards Quantum-Safe VPNs and Internet

Maran van Heesch* Niels van Adrichem* Thomas Attema*† Thijs Veugen*†

*TNO
the Hague, the Netherlands
Email: firstname.lastname@tno.nl

†CWI
Amsterdam, the Netherlands

***Abstract*—Estimating that in 10 years time quantum computers capable of breaking public-key cryptography currently considered safe could exist, this threat is already eminent for information that require secrecy for more than 10 years. Considering the time required to standardize, implement and update existing networks signifies the urgency of adopting quantum-safe cryptography.**

In this work, we investigate the trade-off between network and CPU overhead and the security levels defined by NIST. To do so, we integrate adapted OpenSSL libraries into OpenVPN, and perform experiments on a large variety of quantum-safe algorithms for respectively TLS versions 1.2 and 1.3 using OpenVPN and HTTPS independently. We describe the difficulties we encounter with the integration and we report the experimental performance results, comparing setting up the quantum-safe connection with setting up the connection without additional post-quantum cryptography.

I. Introduction

Research in the field of quantum computing is progressing, the first computational operations can already be performed on actual qubits and the first small gated quantum computer is commercially available [1]. For a specific computational task the first experimental realisation of quantum supremacy has been achieved [2]. This new paradigm for computing is expected to allow us to solve (computational) problems that are currently deemed infeasible to solve on traditional (super) computers. Unfortunately, some of the mathematical problems that become solvable with quantum computers form the basis of many cryptographic primitives that we use today to protect the confidentiality, authenticity and integrity of data transmitted over the Internet. More specifically, nearly all approaches for public-key cryptography currently in use (e.g., RSA and Elliptic Curve Cryptography) are vulnerable to a quantum-computer attack. This is due to Shor's algorithm [3], which can be used to solve the factoring and the Discrete Log problem used in public-key cryptography in polynomial time.

Experts say it is likely that a quantum computer capable of breaking 2048-bit RSA encryption in a matter of hours can be built by 2030 [4]. Even though the availability of a quantum computer for practical cryptographic attacks is still one or two decades in the future, we need to develop and adopt *quantum-safe* alternatives, being post-quantum cryptographic primitives. Malicious actors can already intercept and store large amounts of encrypted traffic, and wait for the availability of quantum computers to decrypt the traffic at a later time. Additionally, the standardization of post-quantum cryptographic protocols is a process that takes at least until late 2020 [5], and it will take even more time before hardware and software have been updated (or replaced) to comply with these new standards. Hence, the threat of quantum computers already needs to be taken into account when sharing sensitive information which needs to be kept secret beyond 2030, while the window for a smooth transition towards quantum-safe cryptographic protocols is narrowing [6].

One of the problems that is encountered in the transition from one cryptographic primitive to another is that primitives are not plug-and-play. A large number of devices need to be upgraded to achieve mass compatibility, often requiring replacement of hardware appliances. Furthermore, the mathematical problems used in the new post-quantum primitives are more complex than the factoring and Discrete Log problem. In general, the use of post-quantum primitives results in increased CPU usage and larger key sizes, leading to additional use of network resources. In order to prepare for the adoption of post-quantum cryptography, it is not only important to consider post-quantum cryptography as a stand alone primitive, but also to investigate the requirements that the use of it poses on our digital infrastructures and communication networks.

In this work, we investigate the impact of implementing post-quantum cryptographic protocols in OpenVPN [7] and OpenSSL [8], and study the performance and various trade-offs to be made when selecting one of many available post-quantum primitives. OpenVPN is a commonly used open-source software suite used to set up secure VPN connections. In this work OpenVPN is considered to be an end-to-end secure communication link, but OpenVPN can be used to construct large, advanced network topologies.

We describe how OpenVPN and OpenSSL can be configured to establish quantum-safe connections in section II, considering both post-quantum key exchange and post-quantum authentication. In section III a performance analysis on various (quantum-safe) cryptographic ciphers with TLS 1.2 [9] on OpenVPN and TLS 1.3 [10] independently is presented, and the effects of using post-quantum cryptography compared to current state-of-the-art cryptography are quantified. Related work is presented in Section IV.

[1]Using the "recommended" parameter set.
[2]Using the EES743EP1 parameter set.
[3]Newhope is an improved version of rlwe-bcns15 presented.

TABLE I
PQ-KEX AND PQ-SIGN SCHEMES SUPPORTED BY OQS-OPENSSL IN TLS 1.2 AND CURRENTLY CONSIDERED BY NIST

Cipher		NIST		Conventional or PQ-KEX		Hybrid KEX with ECDHE	
PQ-KEX	Type	Security Level	Contestant	Bytes transmitted	Nb. of instructions	Bytes transmitted	Nb. of instructions
ECDHE [11]				7289	26710360	Not applicable	Not applicable
frodo[1] [12]	Lattice	130 bits (L1)	Yes	30326	35931344	30509	36802746
ntru[2] [13]	Lattice	159 bits (L1)	Merged	9195	39295256	9378	40152096
rlwe-bcns15 [14]	Lattice	78 bits	Renewed[3]	15646	35996687	15829	36872509
rlwe-msrln16 [15]	Lattice	128 bits (L1)	Merged[4]	11054	27006717	11237	27886168
newhope [16]	Lattice	128 bits (L1)	Yes	11054	26963921	11237	27861721
sidh [17]	Isogeny	128 bits (L1)	Sike only	7866	568167669	8049	569050654
PQ-SIGN							
picnic [18]	Hash	128 bits (L1)	Yes				
frodo+picnic	Combined			23218	150150465	232352	151022835
rlwe-msrln16+picnic	Combined			212952	141301962	213172	142235675
sidh+picnic	Combined			209748	682411927	209911	683303939

II. IMPLEMENTATION

In this paper, we consider 2 different test cases and a variety of (quantum-safe) ciphers. Our scenario comprises a 1-to-1 setup, considering two distinct networks or hosts that are connected via a single encrypted connection. Although a limited setup, our results can be extrapolated to larger networks for larger quantitative analysis. Both test cases consider the Transport Layer Security (TLS) protocol as communication protocol to set up and maintain the encrypted communication channel. TLS is well-known and commonly used when protecting web traffic, e-mail services and VPN services such as HTTPS and OpenVPN. In the first test case in Section III, we use OpenVPN [7] on top of TLS version 1.2 [9] to set up encrypted connections, since it is a commonly used VPN solution. In the second test case we use TLS 1.3 [10] with HTTPS.

A. Post-quantum cryptography implementation

The Open Quantum Safe (OQS) project [19] provides open-source prototypes implementing many quantum-safe ciphers, among which candidate ciphers for NIST[5][5]. Additionally, OQS offers a customized version of OpenSSL [20] (*OQS-OpenSSL*), an open-source TLS library, offering an experimental SSL library with quantum-safe ciphers. We have used two branches of the OQS library *liboqs*, being *master* and *nist-branch*. For use with TLS 1.3 independently, we used these two branches with OQS-OpenSSL branch *OQS-OpenSSL_1_1_1-stable*, while we compiled OpenVPN with TLS 1.2 against OQS-OpenSSL from branch *OpenSSL_1_0_2*. We have evaluated both quantum-safe key exchange protocols (*PQ-KEX*) and signature schemes (*PQ-SIGN*), as well as combined (PQ-SIGN and PQ-KEX) and hybrid solutions.

Hybrid solutions imply the use of both quantum-safe and quantum-unsafe protocols to further elevate the security level. Many cryptographic vulnerabilities lie in their implementations. Conventional quantum-unsafe schemes have historic track records of implementation vulnerabilities and their solutions. Hence, combining a conventional quantum-unsafe cryptographic scheme with a novel quantum-safe scheme assures that when unexpected implementation vulnerabilities are found in the quantum-safe scheme one still enjoys the security of the conventional quantum-unsafe scheme.

When a hybrid key exchange protocol is tested, two cryptographic schemes are executed in parallel, and two TLS pre-master secrets are determined. The resulting TLS pre-master secrets are concatenated as input to the key derivation function, determining the TLS master secret used to derive the keys for the symmetric cipher and Message Authentication Code.

a) Security level: Additional to testing various PQ-KEX and PQ-SIGN schemes, we have tested the schemes for various security levels. NIST has defined five levels of security [21] for which the cryptographic protocols can have a parameter set, being: any attack that breaks the relevant security definition must require computational resources comparable to, or greater than, those required for:

- L1 key search on a block cipher with a 128-bit key (e.g. AES128 on a conventional computer),
- L2 collision search on a 256-bit hash function (e.g. SHA256/ SHA3-256 on a conventional computer),
- L3 key search on a block cipher with a 192-bit key (e.g. AES192 on a conventional computer),
- L4 collision search on a 384-bit hash function (e.g. SHA384/ SHA3-384 on a conventional computer),
- L5 key search on a block cipher with a 256-bit key (e.g. AES 256 on a conventional computer).

These five levels are reconsidered in a *quantum world*, where an adversary has access to a quantum computer. For each cryptographic cipher, multiple parameter sets have been selected in order to satisfy various security levels. We have indicated the security level per cryptographic protocol considered in the experiments in tables I and II.

b) Algorithms used in OpenVPN with TLS 1.2: The implementations of the cryptographic ciphers that we consider in our experiment using OpenVPN with TLS 1.2 are listed in Table I. Additionally, we mention the class of assumptions on which the cryptographic protocols are based. Additionally, we

[4]The scheme rlwe-msrln16 contains improvements to the number theoretic transform used in newhope. These changes were later added to newhope, making the two schemes very similar.

[5]We did not validate the correctness and security of these cryptographic implementations.

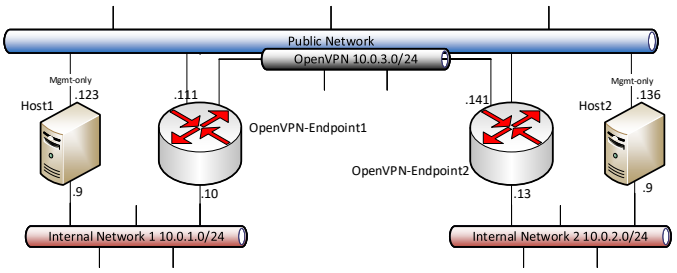

Fig. 1. Experiment topology

TABLE II
PQ-KEX AND PQ-SIGN SCHEMES USING TLS 1.3 SUPPORTED BY OQS-OPENSSL AND CURRENTLY CONSIDERED BY NIST

PQ-KEX	Type	Security levels and parameters		
bike [22]	Codes	L1	L3	L5
frodo [12]	Lattice	L1 (640)	L3 (976)	
kyber [23]	Lattice	L1 (512)	L3 (768)	L5 (1024)
ledakem [24]	Codes	L1 (C1)	L3 (C3)	L5 (C5)
newhope [16]	Lattice	L1 (512)		L5 (1024)
saber [25]	Lattice	L1 (Light)	L3	L5 (Fire)
sike [26]	Isogeny	L1 (p503)	L3 (p751)	
PQ-SIGN				
Picnic [18]	Hash	L1	L3	L5
Qtesla [27]	Lattice	L1 (I)	L3 (III)	

mention whether the cipher reached the second round of NIST standardization, or has been merged into another cipher.

Key exchanges are by default executed with an RSA authentication of 2048 bits or with picnic authentication when denoted. Picnic authentication is only supported in combination with frodo, rlwe-msrln16 and sidh key exchanges. All key exchanges, including those with picnic, have been executed in both native and hybrid mode with ECDHE. After key exchange and authentication all connections use a symmetric cipher with AES 256-bit encryption, which is considered quantum-safe on its own offering 128 bits of security in a quantum world, in Galois/Counter Mode of operation with a SHA-384 hash for the key expansion function and message authentication.

c) Algorithms in TLS 1.3 independently: Instead of using OpenVPN with TLS 1.2, for TLS 1.3 we have used the OpenSSL [8] *s_server* and *s_client* applications to set up connections and measure CPU and network overhead for HTTPS-alike connections. The OQS branches for TLS 1.3 support more candidate ciphers considered by NIST. In Table II we list all schemes implemented by OQS that are still considered by NIST and evaluated in our work. For each algorithm, we mention the class of assumptions on which the cryptographic protocols are based, their respective NIST security levels, and denote their parameter sets or names specific to that security level.

We note that there are more cryptographic schemes still under consideration for standardisation by NIST than currently available in OQS, those are not considered in this work.

B. Hardware

The experimental setup is realized on our private cloud environment based on OpenStack. On our cloud, a quantum-safe OpenVPN and OpenSSL testbed has been created by reserving resources and creating internal and public networks representing trusted and untrusted networks, respective to red-black architectures where red networks represent trusted plaintext (unencrypted) networks and black networks represent untrusted networks over which solely encrypted information is transmitted.

Figure 1 represents the topology in the cloud environment. The internal networks 1 and 2 are networks internal to respectively hosts 1 and 2 and routers OpenVPN-Endpoints 1 and 2. The routers OpenVPN-Endpoint 1 and 2 connect to each other through the public network, which is considered unsafe for data distribution, and set up a secure communication channel represented as network OpenVPN. Through routers OpenVPN-Endpoint 1 and 2, hosts 1 and 2 can communicate securely. Additionally, we connected hosts 1 and 2 directly to the public network for management purposes of our experiment. In a production network these management interfaces require additional security measures.

Both hosts and routers are running on virtual machines with two virtually assigned CPUs of the x86-64 processor architecture, 2 GB RAM and 20 GB of disk space, hypervisors are interconnected through 2 aggregated network links of 10 Gbps each.

C. Implementation challenges

Although OQS already offers a customized version of OpenSSL that works with most of the algorithms out of the box (OQS-OpenSSL), we had to implement a few changes to both OpenSSL and OpenVPN to acquire a functioning setup.

For TLS 1.2 with OpenVPN we adapted OQS-OpenSSL branch *OpenSSL_1_0_2-stable* in the following ways. First, we compiled OpenSSL with the shared-libraries option enabled to obtain the Shared Object libraries *libpicnic*, *liboqs*, *libcrypto* and *libssl*. Additionally, we increased the internal message buffer to 4 times the size of its initial maximum length, to prevent the connection set-up from failing as the larger picnic certificates did originally not fit within the internal message buffer. Finally, we adapted OpenVPN to include the new Shared Object libraries liboqs and libpicnic and replaced libssl and libcrypto with the adapted versions and compiled OpenVPN accordingly. This resulted in a version of OpenVPN integrating OQS-OpenSSL and enabled us to use quantum-safe cryptography in OpenVPN. Although we solved these practical burdens to set up the encryption channels, we remained experiencing problems running specifically McBits [28] key exchange and sidh key exchange using its initial reference implementation, the sidh implementation by Microsoft Research Labs did function. Hence, we have not been able to evaluate McBits and the reference implementation of sidh for OpenVPN with TLS 1.2.

For TLS 1.3 with HTTPS, we were able to acquire a functioning setup by following the readme instructions of OQS-OpenSSL branch OQS-OpenSSL_1_1_1-stable. The biggest

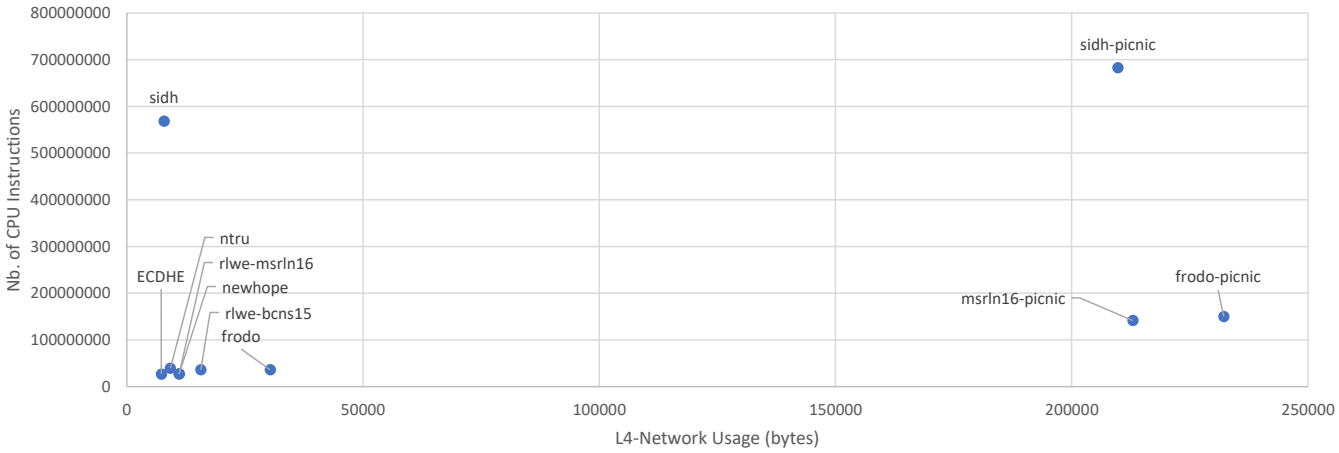

Fig. 2. Performance evaluation of PQ-KEX and PQ-SIGN using TLS 1.2 with default parameters of OQS-OpenSSL branch OpenSSL_1_0_2-stable.

difference is that this branch requires to compile liboqs manually for either its *master* or *nist-branch* branches, depending on the algorithms we needed to evaluate, whereas with TLS 1.2 compiling liboqs was included in the process.

Finally, we set up independent Public Key Infrastructures (PKI) for individual iteration in the experiments in Section III and cipher. For TLS 1.2 we set up PKIs based on 2048-bit RSA certificates and PKIs based on picnic authentication through "openssl x509" commands. For each PKI we created a self-signed Certificate Authority (CA), which signed two certificates to be used by the endpoints, all signed using either RSA or picnic. Picnic signatures exist of a SHA512 hash of the certificate input, which is private-key encrypted by picnic. For TLS 1.3 we also set up PKIs for each authentication algorithm, albeit that this version supports more authentication algorithms.

III. Experiment and Results

Quantifying the impact of post-quantum cryptography in (large) communication networks can be done in various ways. Aspects of network traffic that can be measured are network usage (packet and byte counters), throughput, delay, jitter, packet loss and out-of-order packet delivery. On the client/server side, CPU load and memory usage can be observed.

In this section, we describe how we determined the impact of post-quantum cryptography on our experiment setup and experimental results are discussed.

A. Experiment scenario

As far as we currently know the main threat of the quantum computer lies in the use of public-key cryptography. We determined that the start-up phase of an encrypted communication channel is hence the most relevant to evaluate. To do so, we have performed the following actions. For OpenVPN with TLS 1.2, we automatically set up an OpenVPN connection from endpoints 1 to 2 (see Figure 1), submit four ICMP echo requests [29], log the resulting ICMP echo replies to confirm VPN connectivity, and break down the connection again. During the life-span of the iteration, we capture all OpenVPN traffic between the VPN endpoints using *tcpdump* and gather performance profiling statistics using *perf*. This process is repeated 100 times for each cipher.

For TLS 1.3 we use a similar approach, except that we use TLS 1.3 independent from OpenVPN and respectively run OpenSSL's *s_server* and *s_client* applications on endpoints 1 and 2 (see Figure 1) to set up an encrypted HTTPS channel, and send a 24-byte message for each iteration of each cipher.

The scenario is executed on our private OpenStack-based cloud, described in section II-B. Due to the shared nature of a virtualized environment, potential spatiotemporal inaccuracies are more likely to occur. Besides repeating the procedure 100 times for every described cipher, we execute each cipher once at every iteration instead of running the ciphers 100 times sequentially to enable us to detect such cross-talk with other users of the cloud.

The experiments provide ample metrics to evaluate. Where *perf* already provides aggregated statistics for each iteration and cipher, we used *tshark* to analyze the traces from *tcpdump* and generate statistics such as OSI-Layer 2 to 4 frame and payload lengths and packet counters. We used the number of issued CPU instructions as an indication for the computational complexity, and the number of bytes sent and transmitted by OpenVPN or OpenSSL (OSI Layer 4 payload) as a measure for the network overhead.

We decided to use the more abstract metric of CPU instructions instead of a more detailed metric such as CPU cycles, since the actual amount of consumed CPU cycles is influenced by which optimizations are implemented in the used processor. Instruction execution optimizations, parallelization, pipelining and branch predictioning influence the efficiency of execution by the processor and hence the actual number CPU cycles used. Hardware appliances tend to have even greater improvements due to the use of application-specific processors instead of general-purpose processors. Taking into account that we aim to give an overview that is useful for devices ranging from simple embedded processors, to general-purpose

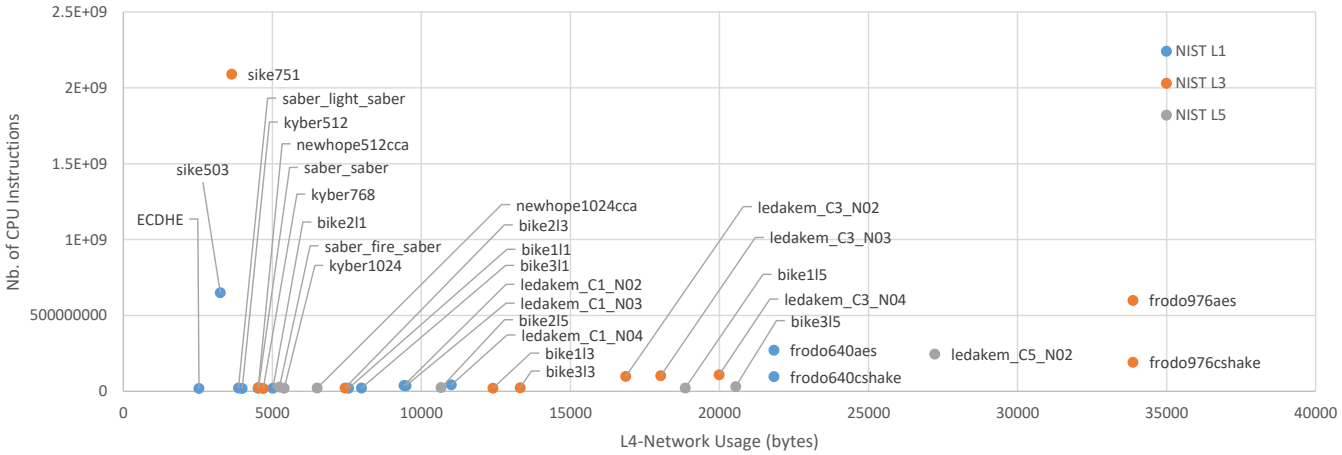

Fig. 3. Performance evaluation of native and hybrid PQ-KEX using TLS 1.3

computing and application-specific designs and the fact that there is a high correlation between the number of issued CPU instructions and consumed cycles, we decided to use the more abstract number of CPU instructions instead.

B. Results

We split the performance results based on whether they were achieved through OpenVPN with TLS 1.2, or through OQS-OpenSSL independently with TLS 1.3. Both results contain measurements regarding post-quantum key exchange and post-quantum authentication. Both results also contain hybrid approaches where a traditional and quantum-safe algorithm are combined. However, the exact algorithms evaluated depend on whether they were implemented in OQS-OpenSSL and liboqs.

a) Results with OpenVPN on TLS 1.2: We have measured the performance results of the start-up phase of OpenVPN using quantum-safe key exchange ciphers with RSA 2048-bit authentication, as well as quantum-safe authentication through the picnic algorithm combined with three compatible key exchanges. Figure 2 illustrates the network overhead on the x-axis, compared to the CPU overhead illustrated on the y-axis.

Additionally, we have repeated measurements for all ciphers with hybrid key exchange, combining PQ-KEX with ECDHE for additional security. The experimental results of the hybrid PQ-KEX are included in Table I. Due to the minimal increase in overhead hybrid PQ-KEX is omitted in Figure 2 for readability purposes. The data-point labelled ECDHE is used as benchmark, as these results come from setting-up the VPN connection with traditional cryptography.

From Figure 2 it is clear that both the PQ-KEX and PQ-SIGN perform differently considering their CPU or network overhead. Sidh has the smallest network overhead, roughly 600 bytes compared to ECDHE, whereas it has by far the largest CPU overhead, over 21 times more than setting up the VPN connection without a quantum-safe key exchange. Newhope has the smallest CPU overhead, with rlwe-msrln16 as a close second. Both have a network overhead of approximately 1.5 times more than setting up the VPN connection with ECDHE. The most expensive in terms of network overhead, frodo, has an overhead increase of approximately factor 4.

Figure 2 illustrates that adding the picnic signature scheme has a big impact on the set-up time of the quantum-safe VPN. The network overhead increases up to 26.6 times when using sidh with or without picnic authentication, while the CPU overhead increases up to 5.2 times when using rlwe-msrln16 with or without picnic authentication.

b) Results with OpenSSL independently on TLS 1.3: Instead of working with OpenVPN, this version of TLS has been evaluated through using HTTPS directly on OpenSSL.

Figure 3 illustrates the overhead of various PQ-KEX algorithms compared to the benchmark of ECDHE. The name of the implementation describes the parameter option that is chosen for the instantiation, which is linked to the NIST security level that is aimed for in Table II. Note that when the security level increases, the performance of the cryptographic ciphers decreases. Some ciphers show multiple data points, this is due to the fact that there are multiple parameter instances that can be used. Depending on the required security and allowed infrastructure overhead, a parameter set should be selected.

Comparing L1 security to the baseline ECDHE implementation, we observe that the least network overhead is measured using *sike503*, as was the case in the earlier experiment where the (also isogeny-based) sidh (which has a different protocol structure than sike503) has the smallest network overhead. The network overhead is 1.3 times larger than using the benchmark, but in the TLS 1.2 experiment it is only 1.08 times larger. Similar, the CPU overhead is the largest, 34 times higher. For L1 security, *kyber512* [23] and *saber_light_saber* [25] appear to be the most suitable candidates to protect the TLS connection, as they provide both low CPU and network overhead.

Comparing L3 security to the baseline ECDHE implementation, considering only the implementations of the Open Quantum Safe project, *bike* and *kyber* appear to be the most

suitable PQ-KEX for the VPN test case using TLS1.3.

In Figure 4 we illustrate the effect of using quantum-safe authentication[6]. There are two parameter sets tested for *qtesla* at NIST L3, one optimizing for speed, and one for reducing the signature size. The qtesla and *picnicL1FS* schemes have been tested independently and to show the effect of hybrid connections adding a second, quantum-unsafe, signature protocol. The quantum-unsafe protocols used in hybrid mode are ECC-P256, ECC-P384 and RSA-3072. We observed that adding the elliptic curve signatures does not affect the performance much, whereas adding RSA-3072 signatures adds a CPU overhead of 61% in the case of picnic and 53% in the case of qteslaI.

IV. Related Work

The need for post-quantum cryptographic primitives has become apparent and research efforts have grown significantly. Moreover, applicability has become an important and challenging research direction. The vast amount and diversity of new cryptographic schemes and their parameter selection clearly indicate the various performance and security trade-offs that can be made. Post-quantum cryptographic primitives will most-likely impact current communication infrastructures, for example by introducing larger keys, failure probabilities, or an increased computational burden. Important questions about the implications of real-world implementations thus arise.

In 2016 Google, operating both the client-side (Chrome browser) and the server-side (various web services), conducted an experiment in which they secured the communication channel with the lattice-based scheme *newhope* [30]. They reported only a minor increase in latency and concluded that, in this specific scenario, they "did not find any unexpected impediment to deploying" [31]. Cloudflare on the other hand reported a significant increase in computational costs when integrating an *sidh* key-exchange protocol in TLS 1.3 [32], building on the work of Microsoft Research [33]. Yet another approach was followed by De Vries, who applied the Niederreiter cryptosystem [34] to achieve post-quantum security in OpenVPN [35]. Since the public keys of the Niederreiter cryptosystem are too large for standard TLS messages, De Vries chose to run the the post-quantum key exchange once a conventional VPN tunnel was set up. Others that have worked on post-quantum VPN solutions are Microsoft Research [36], Mullvad [37], Post-Quantum [38] and InfoSec Global [39].

Crockett et al. [40] provide very extensive results on integrating NIST candidates into OpenSSL and OpenSSH, design considerations and case studies, reporting on work integrating the liboqs [19] library from the Open Quantum Safe project into OpenSSL and OpenSSH. Providing extensive test results whether combinations of cryptographic algorithms and protocols successfully work or not, no quantitative measurements such as in our experiments are provided.

The above experiments indicate that there does not seem to be a single post-quantum cryptosystem that is optimal in all scenarios. This is further reinforced by NIST's *Post-Quantum Cryptography Standardization* initiative, in which 82 proposals were originally submitted and of which currently 26 (15 PQ-KE and 9 PQ-SIGN) proposals are still under consideration [5]. In the case that there will only be two rounds, the first standards will be available by the end of 2020. Software libraries such as liboqs [19] and libpqcrypto [41] enable the performance and security evaluation of these and other cryptographic schemes. Moreover, they play an important role in the adaptation of post-quantum cryptography.

[6]The results are obtained using ECDHE to establish a key between client and server, and therefore the setup is not fully quantum-safe.

V. Conclusion

We have implemented and evaluated post-quantum cryptography in OpenVPN and over HTTPS using an adapted version of OpenSSL. Both OpenVPN and HTTPS use TLS to set up encrypted channels, hence, this evaluation can be considered to be similar to other software solutions using TLS, such as secure email transmissions. We experienced that post-quantum cryptography is not yet plug-and-play, since there are schemes that do not work without a custom integration due to large key or signature sizes, such as McBits.

The experimental results show that, based on the chosen cipher and parameters, there is additional overhead when using post-quantum cryptography. Basing our statement on the implementations provided by the Open Quantum Safe project, *saber_light_saber* and *kyber512* appear to be the most suitable candidates to set-up quantum-safe OpenVPN connections, as they provide both acceptable CPU and network overhead, and claim NIST L1 security. The schemes *saber_saber* and *kyber768* appear to be the most suitable candidates claiming NIST L3 security, for similar reasons.

The security of the schemes *saber* and *kyber* is based on lattice assumptions, the security of *saber* is based on the hardness of module learning with rounding (MLWR) problem and the security of *kyber* is based on the hardness of module learning with errors (MLWE) problem. Both of these problems are variations or adaptations of the LWE problem. These variations are introduced to reduce key-sizes and improve efficiency, but the additional structure in these problems potentially increases the attack surface. To date, no attacks exploiting this structure and breaking these schemes has been found. However, MLWE has only been introduced in 2012 [42] and further cryptanalysis might be required to strengthen our confidence in these schemes. This holds for more of the cryptographic protocols considered by NIST. Therefore it is currently not advised to only use a post-quantum scheme but to combine it with one of the current standards such as RSA and ECDHE. We show that, in general, hybrid operation of quantum-safe and conventional cryptography is efficient in terms of network and CPU overhead and thus provides additional security at low cost.

The results presented in this work are subjected to change when cryptographic implementations are updated. Performance could become better in terms of CPU usage once the code is written more efficiently, and network load can change due to different parameter selection.

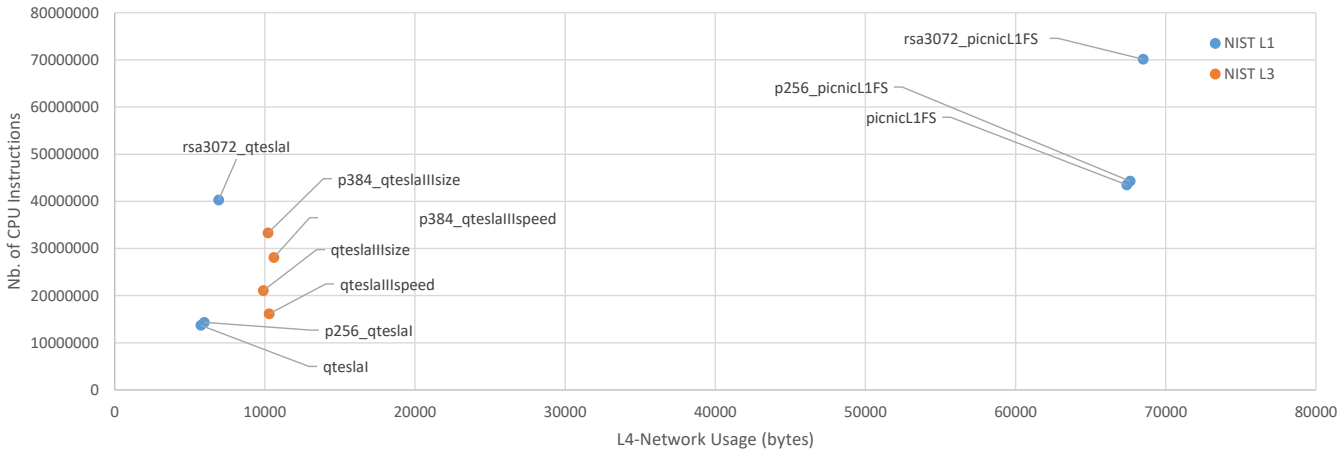

Fig. 4. Performance evaluation of native and hybrid PQ-SIGN using TLS 1.3

ACKNOWLEDGMENT

The authors would like to thank Jacco van Buuren for insightful discussions and technical assistance.

REFERENCES

[1] "IBM Unveils World's First Integrated Quantum Computing System for Commercial Use." [Online]. Available: https://newsroom.ibm.com/2019-01-08-IBM-Unveils-Worlds-First-Integrated-Quantum-Computing-System-for-Commercial-Use#assets_all

[2] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019. [Online]. Available: https://doi.org/10.1038/s41586-019-1666-5

[3] P. W. Shor, "Polynominal time algorithms for discrete logarithms and factoring on a quantum computer," in *Algorithmic Number Theory, First International Symposium, ANTS-I*, 1994.

[4] L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone, "Report on post-quantum cryptography," NIST, Tech. Rep., April 2016, NIST Interagency/Internal Report (NISTIR) - 8105.

[5] G. Alagic, J. Alperin-Sheriff, D. Apon, D. Cooper, Q. Dang, Y.-K. Liu, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, and D. Smith-Tone, "Status report on the first round of the nist post-quantum cryptography standardization process," NIST, Tech. Rep., January 2019, NIST Interagency/Internal Report (NISTIR) - 8240 .

[6] M. Campagne et al., "Quantum Safe Cryptography and Security," ETSI, Tech. Rep., June 2015. [Online]. Available: http://www.etsi.org/images/files/ETSIWhitePapers/QuantumSafeWhitepaper.pdf

[7] "OpenVPN." [Online]. Available: https://openvpn.net/

[8] "OpenSSL." [Online]. Available: https://www.openssl.org/

[9] E. Rescorla and T. Dierks, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246, Aug. 2008. [Online]. Available: https://rfc-editor.org/rfc/rfc5246.txt

[10] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, Aug. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8446.txt

[11] B. Moeller, N. Bolyard, V. Gupta, S. Blake-Wilson, and C. Hawk, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)," RFC 4492, May 2006. [Online]. Available: https://rfc-editor.org/rfc/rfc4492.txt

[12] J. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila, "Frodo: Take off the ring! practical, quantum-secure key exchange from lwe," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1006–1018.

[13] J. Hoffstein, J. Pipher, and J. H. Silverman, "Ntru: A ring-based public key cryptosystem," in *International Algorithmic Number Theory Symposium*. Springer, 1998, pp. 267–288.

[14] J. W. Bos, C. Costello, M. Naehrig, and D. Stebila, "Post-quantum key exchange for the tls protocol from the ring learning with errors problem," in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 553–570.

[15] P. Longa and M. Naehrig, "Speeding up the number theoretic transform for faster ideal lattice-based cryptography," in *International Conference on Cryptology and Network Security*. Springer, 2016, pp. 124–139.

[16] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange—a new hope," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 327–343.

[17] C. Costello, P. Longa, and M. Naehrig, "Efficient algorithms for supersingular isogeny diffie-hellman," in *Annual International Cryptology Conference*. Springer, 2016, pp. 572–601.

[18] M. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, and G. Zaverucha, "Post-quantum zero-knowledge and signatures from symmetric-key primitives," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 1825–1842.

[19] "Open Quantum Safe - liboqs." [Online]. Available: https://github.com/open-quantum-safe/liboqs

[20] "Open Quantum Safe - OpenSSL." [Online]. Available: https://github.com/open-quantum-safe/openssl

[21] "Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process," 2016, NIST. [Online]. Available: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf

[22] N. Aragon, P. Barreto, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, S. Gueron, T. Guneysu, C. A. Melchor *et al.*, "Bike: bit flipping key encapsulation," *Proposal to NIST Standardization Competition.*, 2017.

[23] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Kyber: a CCA-secure module-lattice-based KEM," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2018.

[24] M. Baldi, A. Barenghi, F. Chiaraluce, G. Pelosi, and P. Santini, "Ledakem: a post-quantum key encapsulation mechanism based on qc-ldpc codes," in *International Conference on Post-Quantum Cryptography*. Springer, 2018, pp. 3–24.

[25] J.-P. D'Anvers, A. Karmakar, S. S. Roy, and F. Vercauteren, "Saber: Module-LWR based key exchange,CPA-secure encryption and CCA-secure KEM," in *International Conference on Cryptology in Africa*. Springer, 2018.

[26] R. Azarderakhsh, M. Campagna, C. Costello, L. Feo, B. Hess, A. Jalali, D. Jao, B. Koziel, B. LaMacchia, P. Longa *et al.*, "Supersingular isogeny key encapsulation," *Submission to the NIST Post-Quantum Standardization project*, 2017.

[27] E. Alkim, P. S. Barreto, N. Bindel, P. Longa, and J. E. Ricardini, "The lattice-based digital signature scheme qtesla." *IACR Cryptology ePrint Archive*, vol. 2019, p. 85, 2019.

[28] N. Sendrier, "Code-based cryptography: State of the art and perspectives," *IEEE Security & Privacy*, vol. 15, no. 4, pp. 44–50, 2017.

[29] J. Postel, "Internet Control Message Protocol," RFC 792, Sep. 1981. [Online]. Available: https://rfc-editor.org/rfc/rfc792.txt

[30] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum Key Exchange - A New Hope," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016.

[31] A. Langley, "CECPQ1 results," Nov. 2016. [Online]. Available: https://www.imperialviolet.org/2016/11/28/cecpq1.html

[32] H. de Valence, "SIDH in Go for Quantum-Resistant TLS 1.3," Sep. 2017. [Online]. Available: http://blog.cloudflare.com/sidh-go/

[33] C. Costello, P. Longa, and M. Naehrig, "Efficient algorithms for supersingular isogeny diffie-hellman," in *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference*. Springer, 2016.

[34] H. Niederreiter, "Knapsack-type cryptosystems and algebraic coding theory," *Problems of Control and Information Theory*, vol. 15, no. 2, pp. 159–166, 1986.

[35] S. de Vries, "Achieving 128-bit Security against Quantum Attacks in OpenVPN," Master's thesis, University of Twente, August 2016. [Online]. Available: http://essay.utwente.nl/70677/

[36] H. Easterbrook, K. Kane, B. LaMacchia, D. Shumow, and G. Zaverucha, "Post-quantum Cryptography VPN," 2018. [Online]. Available: https://www.microsoft.com/en-us/research/project/post-quantum-crypto-vpn/

[37] Mullvad, "Introducing a post-quantum VPN, Mullvad's strategy for a future problem," Dec. 2017. [Online]. Available: https://mullvad.net/nl/blog/2017/12/8/introducing-post-quantum-vpn-mullvads-strategy-future-problem/

[38] Post-Quantum, "VPN," 2018. [Online]. Available: https://www.post-quantum.com/vpn/

[39] Infosec Global, "VPN Solution Empowered with Agile Cryptography," 2018. [Online]. Available: https://www.infosecglobal.com/solutions/network-protection/agilesec-vpn

[40] E. Crockett, C. Paquin, and D. Stebila, "Prototyping post-quantum and hybrid key exchange and authentication in tls and ssh," Cryptology ePrint Archive, Report 2019/858, 2019, https://eprint.iacr.org/2019/858.

[41] PQCRYPTO project, "Libpqcrypto," 2018. [Online]. Available: http://libpqcrypto.org/index.html

[42] A. Langlois and D. Stehlé, "Worst-case to average-case reductions for module lattices," *Designs, Codes and Cryptography*, vol. 75, no. 3, pp. 565–599, 2015.