Post-Quantum Cryptography in WireGuard VPN

Quentin M. Kniep(✉), Wolf Müller, and Jens-Peter Redlich

Institut für Informatik, Humboldt-Universität zu Berlin, 10099 Berlin, Germany
{kniepque,wolfm,jpr}@informatik.hu-berlin.de
https://sar.informatik.hu-berlin.de

Abstract. WireGuard is a new and promising VPN software. It relies on ECDH for the key agreement and server authentication. This makes the tunnel vulnerable to future attacks with quantum computers.

Three incremental improvements to WireGuard's handshake protocol are proposed, giving differently enhanced levels of post-quantum security. Performance impacts of these are shown to be moderate.

Keywords: Post-quantum cryptography · VPN · Key exchange

1 Introduction

Security features of VPN software can hide your identity, the actual data, and its destination. Well-known implementations are OpenVPN, IPsec, and WireGuard, which was recently integrated into the Linux kernel. We focus on WireGuard (WG) because it is promising for the future. WG's handshake is currently based on ECDH. However, Rötteler et al. [17] have shown that the quantum computer algorithm by Shor can break it. This allows for retroactive attacks. Therefore, we need to use post-quantum (PQ) primitives for the key agreement now. WG allows to use a 256-bit pre-shared key (PSK) [6], which can come from a PQ-secure handshake. While this ensures basic PQ confidentiality, it has neither PFS nor identity hiding. Symmetric encryption and hashes are only somewhat threatened by Grover's algorithm. Also, there are very reasonable doubts raised about its impact [2,18], mainly because of bad parallelizability.

2 Related Work

Mullvad is a VPN provider that allows to use WG with a PQ PSK. Therefore, they also provide neither PFS nor identity hiding. Researchers at Microsoft use OpenVPN with a full PQ key exchange, with all the advantages and disadvantages of OpenVPN compared to WG. Hülsing et al. also propose an adapted WG handshake protocol [10]. The biggest difference to our adaptions is that they aim

N. Park et al. (Eds.): SecureComm 2020, LNICST 336, pp. 261–267, 2020.
https://doi.org/10.1007/978-3-030-63095-9_16

to replace ECDH with PQ primitives. They further prove security under extensions of the models used on WG [7,8]. The proposed solution is very promising for the future. In contrast, we propose a more conservative integration of a full PQ key exchange in WG, which should also guarantee the same security as WG against classical adversaries.

3 Protocol Design

First we define four incremental **levels of security** in a PQ setting:

Level 0 (L0) basic confidentiality (WG specification),
Level 1 (L1) perfect forward secrecy,
Level 2 (L2) identity hiding (same passive security in PQ setting as WG),
Level 3 (L3) active attacks (same security in PQ as WG in classical setting).

Whereas the security levels defined by NIST [12] concern specific costs for breaking primitives, these are about general security properties of handshake protocols. We use *Roman uppercase* numbers **I**, **III**, and **V** for **NIST security levels** and *Arabic* numbers **0**, **1**, **2**, and **3** for **handshake security levels**.

In the following we present three proposed handshake protocols which are extensions of the WG handshake and satisfy L1–L3 respectively. By the Noise construction they are at least as strong as the WG handshake. [15] Note that WG's handshake is an L0 handshake. The protocols are written in Noise notation [15], with the hybrid forward secrecy extension for KEMs [14]. This is further extended with tokens `sX,skemX` for static public keys and their ciphertexts.

Level 1 Handshake (PFS)

```
<- s
...
-> e, e1, es, s, ss
<- e, ekem1, ee, se, psk
```

PFS is achieved by establishing a shared secret using the PQ KEM with an ephemeral key pair generated by the initiator. For this we need to add the public key to the initiation message and a ciphertext to the response.

Level 2 Handshake (identity hiding)

```
<- s, s1
...
-> e, e1, skem1, es, s, ss
<- e, ekem1, ee, se, psk
```

Identity hiding is achieved by adding a PQ static key pair, in addition to the responder's classical static key pair. The initiator's identity is encrypted with a secret protected under both.

Level 3 Handshake (active attacks)

```
<- s, s1
...
-> e, e1, skem1, es, s, s2, ss
<- e, ekem1, skem2 ee, se, psk
```

For security against active attackers another static PQ key pair is added, this time on the initiator's side. The PQ parts now closely resemble the Noise IK handshake.

4 Performance Analysis

The proof-of-concept implementation is based on BoringTun [5], Cloudflare's Rust implementation of WG, and available on GitHub under the BSD 3-Clause License: https://github.com/qkniep/pqwg-rust. This was originally developed and explained in more detail in my Bachelor's thesis. [11]

Message Sizes. Ciphertexts and public keys of PQ KEMs are hundreds of Bytes large. Since WG is based on UDP and IP packet fragmentation is considered fragile [3], we need to split datagrams on the application layer. In WG there is a 5 s timeout if the handshake fails, e.g. because a datagram was lost. Combined with a study on packet loss [16] this gives about **30 ms per additional datagram**. Choosing cryptographic primitives with small key sizes is thus essential.

Amplification Attacks. To prevent DoS attacks based on amplification, initiation messages should be larger than the response. In the following we always accommodate for the necessary padding in the initiation message.

Memory Exhaustion Attacks. Another problem when splitting messages is the possibility of memory exhaustion attacks. [3] Malicious initiators may send incomplete messages until the responder runs out of memory. WG already has a system against CPU-exhaustion attacks [6], which could be expanded to also prevent memory exhaustion.

4.1 Benchmark Setup

All benchmark results that follow come from a workstation with the following specifications: **CPU** Intel Xeon E3–1230 v3 (4 × 3.3 GHz, 8 MB Cache, AVX2, AES-NI), **RAM** 8 GB DDR3–1600, **OS** Arch Linux x86_64, and **Kernel** 5.3.7–arch1–2–ARCH.

The benchmarks are based on the BoringTun handshake test [5], adapted in the number of packets sent over the WG tunnel, and run through Criterion.rs [9], with *sample size* of 100 and *measurement time* of one minute. Results will be presented only for some of the most interesting cryptographic primitives, based on results from Sect. 4 and the `speed_kem` benchmark in liboqs [13].

4.2 Use Cases

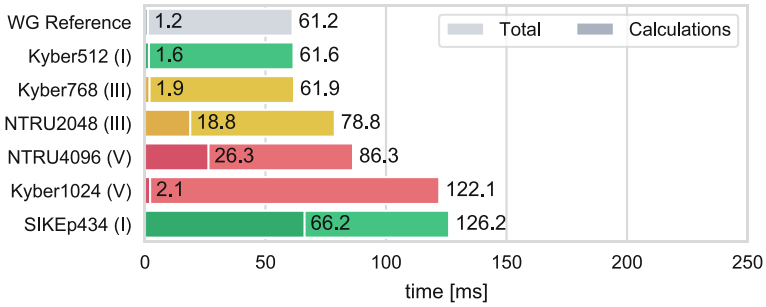

Fig. 1. Time the L1 handshake takes with different PQ cryptographic primitives.

Home VPN Server (L1). Here, identity hiding is not a priority or its impossible anyways. **Kyber-768** seems almost perfect, as it is almost as fast as WG (see Fig. 1), while fitting into one datagram per message. For NIST level V there is a trade-off between computation time and additional datagrams, with **NTRU4096 and Kyber-1024**.

Trusted Network Access (L2). If we want to protect the data and the clients' identities, achieving NIST level V is the most interesting case: It gives the same security guarantees as WG, while also considering the possibility of future quantum-capable attackers. While it could be reasoned to go for level III with Kyber-768, **NTRU4096 or Kyber-1024** both have reasonable cost.

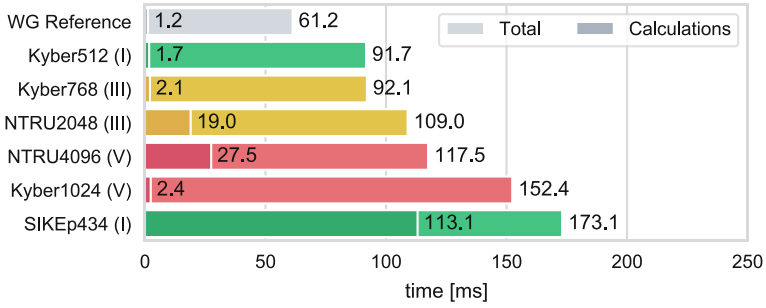

Fig. 2. Time the L2 handshake takes with different PQ cryptographic primitives.

Future Proof System (L3). In cases where a rigid system is built, that can not be adapted once capable quantum computers arrive, it needs to implement an L3 handshake. Such a future-oriented system should reasonably target NIST level V. Then, the handshake needs at least two datagrams more than L2. That is a lot for preventing attacks that are not possible until strong quantum computers are in active use. Only **NTRU4096 and Kyber-1024** seem suitable.

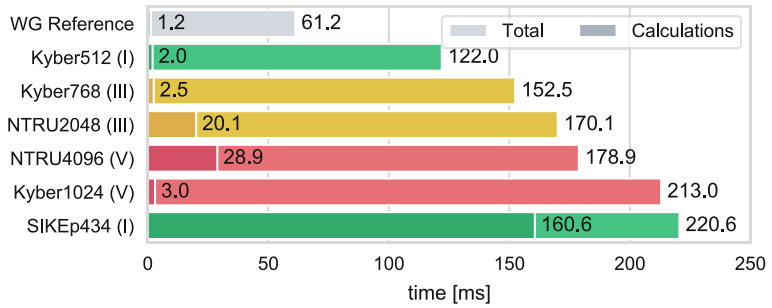

Fig. 3. Time the L3 handshake takes with different PQ cryptographic primitives.

4.3 Throughput, Ping, Reliability

Notably, *nothing* about the symmetric cryptography was changed. Therefore, we expect no noticeable difference in throughput and average packet ping. Especially when measured over the course of a longer networking session.

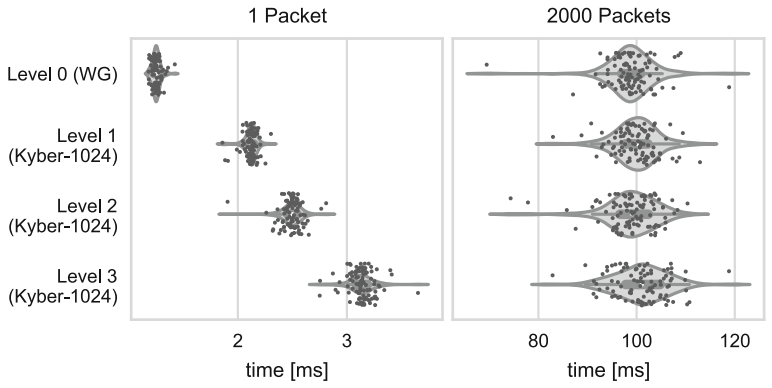

Fig. 4. Runtime of the handshakes with different PQ security levels, each performing the handshake and sending 1 or 2000 packets over the tunnel.

Sending only 2000 packets, it is apparent that the key exchange has little impact in practice. This can be seen in Fig. 4. In these examples packets are sent to *localhost*, latency and its variance link is thus minimal.

WG also has a feature to ensure a failing handshake will, in most cases, not harm transmission throughput over the tunnel: After starting a key exchange there is a sixty second grace period, during which the old key can still be used. [6]

5 Summary

Our analysis has shown that it is already feasible to implement basic PQ security measures, especially when key exchanges do not happen too frequently. At the moment, the cost in performance is still relatively high though. Also, threat models may not even include attacks that far into the future. For almost any use case, it is probably too expensive to preemptively implement security against active quantum adversaries.

While our hybrid approach does not fully adhere to WG's notion of simplicity, we did try to achieve a practical solution, and for now that means using PQ primitives in hybrid. This is also explicitly recommended by the BSI (German Department for IT Security). [4] Once this is no longer necessary, the approach in [10] seems very reasonable.

5.1 Future Work

Results from this work can be used for estimating the cost of including PQ measures into key exchanges. From this point one can decide, whether the gain in security against future attacks is worth the cost in computation and transmission times today. The work by Hülsing et al. [10], which adapts proofs for WG [7,8] to their PQ adaption of WG, could be used as strong foundation to make similar proofs for our hybrid construction. In a future version of this work we would redefine L1 to achieve identity hiding but not PFS, by using the Tiny WireGuard Tweak [1], and then, add the same steps now added for L1 in L2 instead.

References

1. Appelbaum, J., Martindale, C., Wu, P.: Tiny wireguard tweak. In: Buchmann, J., Nitaj, A., Rachidi, T. (eds.) AFRICACRYPT 2019. LNCS, vol. 11627, pp. 3–20. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-23696-0_1
2. Bernstein, D.J.: Cost analysis of hash collisions: will quantum computers make SHARCS obsolete. SHARCS **9**, 105 (2009)
3. Bonica, R., Baker, F., Huston, G., Hinden, R., Troan, O., Gont, F.: IP fragmentation considered fragile. Internet-Draft draft-ietf-intarea-frag-fragile-17, IETF Secretariat, September 2019. http://www.ietf.org/internet-drafts/draft-ietf-intarea-frag-fragile-17.txt
4. Bundesamt für Sicherheit in der Informationstechnik: Migration zu post-quanten-kryptografie. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Krypto/Post-Quanten-Kryptografie.pdf (2020). Accessed 25 June 2020

5. Cloudflare: Boringtun, March 2019. https://github.com/cloudflare/boringtun. Accessed 25 June 2020
6. Donenfeld, J.A.: WireGuard: next generation kernel network tunnel. In: NDSS (2017)
7. Donenfeld, J.A., Milner, K.: Formal verification of the WireGuard protocol (2017). https://www.wireguard.com/papers/wireguard-formal-verification.pdf. Accessed 25 June 25 2020
8. Dowling, B., Paterson, K.G.: A cryptographic analysis of the wireguard protocol. In: Preneel, B., Vercauteren, F. (eds.) ACNS 2018. LNCS, vol. 10892, pp. 3–21. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93387-0_1
9. Heisler, B.: Criterion.rs. March 2014. https://github.com/bheisler/criterion.rs. Accessed 25 June 2020
10. Hülsing, A., Ning, K.C., Schwabe, P., Weber, F., Zimmermann, P.R.: Post-quantum WireGuard. Technical report, Cryptology ePrint Archive, Report 2020/379. http://eprint.iacr.org/2020/379 (2020)
11. Kniep, Q.M.: Post-quantum cryptography in WireGuard VPN (2019). https://hu.berlin/PQWireGuard
12. National Institute of Standards and Technology: Submission requirements and evaluation criteria for the post-quantum cryptography standardization process, December 2016. https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf. Accessed 25 June 2020
13. Open Quantum Safe: liboqs, August 2016. https://github.com/open-quantum-safe/liboqs/. Accessed 25 June 2020
14. Perrin, T.: KEM-based hybrid forward secrecy for noise (2018). https://github.com/noiseprotocol/noise_hfs_spec/blob/master/output/noise_hfs.pdf. Accessed 25 June 2020
15. Perrin, T.: The Noise protocol framework, July 2018. https://noiseprotocol.org/noise.pdf. Accessed 25 June 2020
16. Raghavendra, R., Belding, E.M.: Characterizing high-bandwidth real-time video traffic in residential broadband networks. In: 8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, pp. 597–602. IEEE (2010)
17. Roetteler, M., Naehrig, M., Svore, K.M., Lauter, K.: Quantum resource estimates for computing elliptic curve discrete logarithms. arXiv preprint arXiv:1706.06752 (2017)
18. Zalka, C.: Grover's quantum searching algorithm is optimal. Phys. Rev. A **60**(4), 2746 (1999)