Quantum secure connection to your institution

Nick Aquina, Nick.Aquina@os3.nl

University of Amsterdam
SURF

15 February 2023

Abstract

Once a powerful enough quantum computer is built, it will be able to decrypt communications from the past. eduVPN is an open-source VPN service used to securely connect to an institution's network. In this paper, we will explore how we can protect eduVPN traffic against future decryption by a quantum computer that is able to decrypt currently used cryptography. Two contributions are described:

1. A protocol that can be used to protect against an eavesdropper with a quantum computer. This protocol uses TLS-PSK and has been implemented in an Android application and in an Apache web server.
2. A modified version of the eduVPN API that protects the VPN connection against an eavesdropper with a quantum computer. This protocol uses its own symmetric encryption and has been implemented in an Android application and into the eduVPN web application.

1 Introduction

eduVPN is a service offered by GEANT based on open source software. It is used by more than 140 organizations world-wide. Organizations using eduVPN mainly consist of NRENs, universities and research institutions[1]. Using eduVPN, students and employees can access the organization's network without having to be physically present at the organization. eduVPN also offers a VPN connection which can be used to access the internet. Using this VPN connection will prevent against man-in-the-middle attacks and eavesdropping in the network of the user.

The cryptography used by eduVPN is not quantum resistant. When a quantum computer is built that is powerful enough to run Shor's algorithm[2], VPN traffic could be decrypted and the privacy of users violated.

The goal of the project is to research how to prevent eduVPN traffic that is stored now from being decrypted later when a quantum computer becomes available, and to build a Proof of Concept demonstrating a quantum-resistant eduVPN.

1.1 Scope / Attack model

We assume that, right now, no sufficient quantum computer exists that can break cryptography. Therefore, we will only protect against an attacker which can not do an active MiTM attack with a quantum computer right now.

eduVPN supports 2 VPN protocols: OpenVPN and WireGuard. Because OpenVPN is planned to be removed in the future[3], we will only consider attacks against WireGuard.

1.2 Outline

First, we will explain how a post-quantum key exchange works in section 2. Section 3 presents the main research question and sub-questions. Section 4 discusses related work. Section 5, 6 and 7 all answer a different sub-question. Section 8 presents the conclusions. Finally, in section 9 future work is discussed.

2 Preliminary: Post-quantum key exchange

A post-quantum key exchange works in a different way than Diffie-Hellman. To understand a post-quantum key exchange, we compare it to a Diffie-Hellman key exchange and public-key encryption.

Abbreviations

pk public key
sk secret key (also private key)
c ciphertext
ss shared secret
DH Diffie-Hellman function
KDF Key Derivation Function

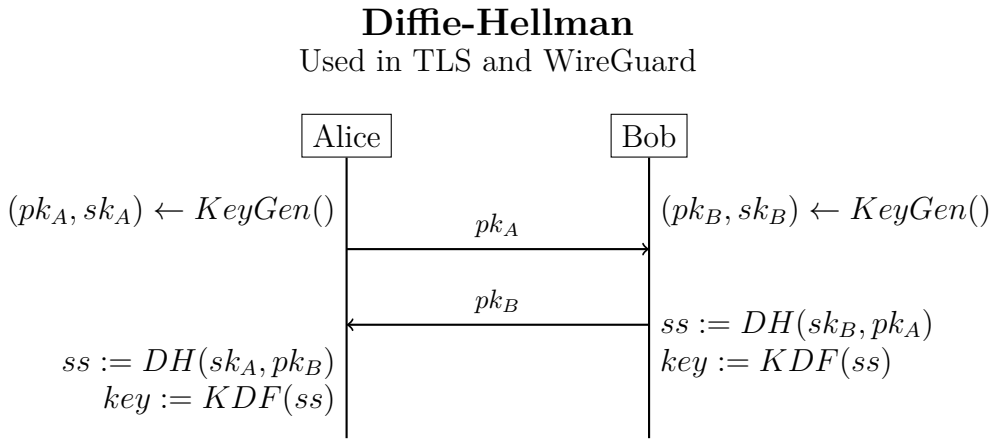

Figure 1: Diffie-Hellman key exchange

In Diffie-Hellman, Alice and Bob both generate a keypair. They both send the public key to the other person and then generate the shared secret using their own private key and the other person's public key.

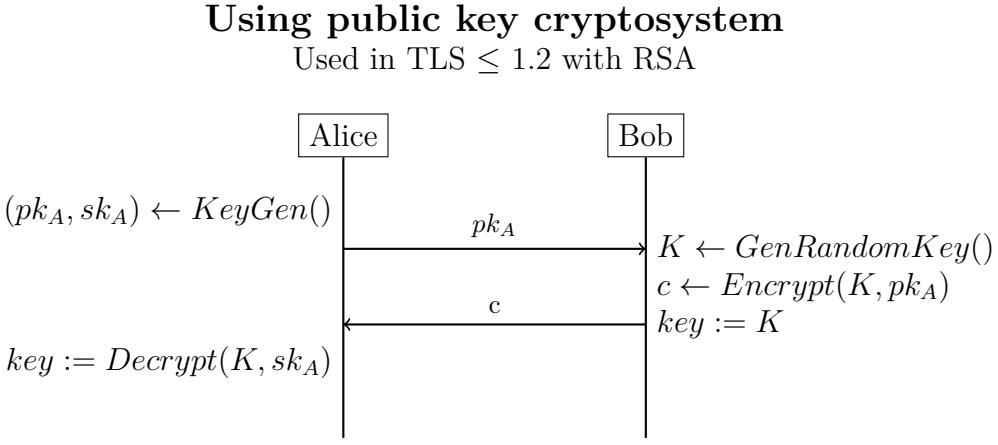

Figure 2: Key exchange using public key cryptosystem

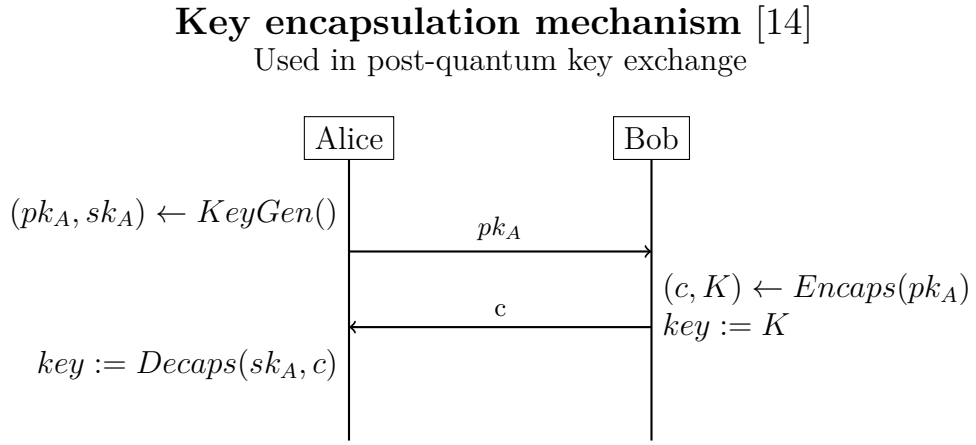

Figure 3: Key exchange using key encapsulation mechanism

With a public key cryptosystem and a key encapsulation mechanism (KEM) Alice generates a keypair and sends this to Bob. Bob generates a ciphertext which Alice decrypts or decapsulates.

As can be seen in Figure 1, 2 and 3, a KEM is more similair to using a public key cryptosystem than to Diffie-Hellman. This is because KEMs are built on top of public-key encryption schemes[4]. An important difference between Diffie-Hellman and the 2 other systems, is that in Diffie-Hellman, both parties can publish their public keys and derive a shared secret without having to interact with each other[5]. With the other 2 systems, one party has to encapsulate (Encaps) the public key and send the encapsulation to the other party, which decapsulates (Decaps) the ciphertext. This requires interaction. In TLS, Diffie-Hellman can be replaced by a KEM because the interaction is already present in the protocol. This is however not the case in WireGuard.

There is one post-quantum key exchange algorithm in the NIST post-quantum competition that is more similair to a Diffie-Hellman key exchange: SIKE[6]. However, SIKE has been shown to be insecure[7].

After using Diffie-Hellman (DH function), the key should be generated using key derivation function (KDF). In the case of post-quantum KEMs, this is not necessary because the KEM internally already uses a KDF or hashing algorihtm.

Note that all these key exchanges do not authenticate the other participant and thus only protects against a passive man in the middle.

3 Research question

The main research question that will be answered is: How can eduVPN traffic be protected against decryption by quantum computers?

To answer this research question, the following sub-questions will be answered:

- In section 5: What data is sent between the eduVPN client and the server?
- In section 6: Which data should be protected against future decryption?
- In section 7: How can confidential data be protected against decryption by quantum computers?

4 Related work

This paper is not the first paper that secures existing software against the threat of quantum computers. Prior work has been done on securing both TLS and WireGuard against this threat. Both protocols are used in eduVPN to protect data. We discuss how we protect against this threat in section 7.

4.1 Post-quantum TLS

In `Prototyping Post-Quantum and Hybrid Key Exchange and Authentication in TLS and SSH`[8] E. Crockett et al. implement several post-quantum key exchanges and authentication in TLS 1.2, TLS 1.3 and SSHv2. The TLS implementation is done in forks of OpenSSL, the same library is used on eduVPN servers to provide TLS.

In `Post-Quantum TLS Without Handshake Signatures`[9], P. Schwabe et al. present a version of TLS that uses a post-quantum key exchange and also uses a key exchange for authentication. The implementation is done using a fork of OpenSSL, the same library is used on eduVPN servers to provide TLS. The performance of their TLS variant (KEMTLS) is compared with TLS 1.3.

4.2 Post-quantum WireGuard

In `Tiny WireGuard Tweak`[10] J. Appelbaum et al. present a modified version of WireGuard that protects against future decryption by a quantum computer in case the adversary does not have access to the used public keys.

In `Post-quantum WireGuard` [5] A. Hülsing et al. present a modified version of WireGuard that uses a post-quantum key exchange to protect against future decryption by quantum computers. This modified variant also provides post-quantum authentication.

5 What data is sent between the eduVPN client and server?

eduVPN offers a way to securely connect to the network of the institution. This has been visualized in Figure 4.

To investigate the communication between the server and client, a test eduVPN server was used and a rooted Android phone with a custom certificate authority was used to connect to this server. This custom certificate authority in combination with mitmproxy

and Wireshark allows to inspect the traffic of the Android phone. The investigation was done using eduVPN server version 3.2.2 on Debian 11 and eduVPN Android app version 3.0.1 on LineageOS 20 (Android 13) on a OnePlus 5. The traffic has been visualized in Figure 5.

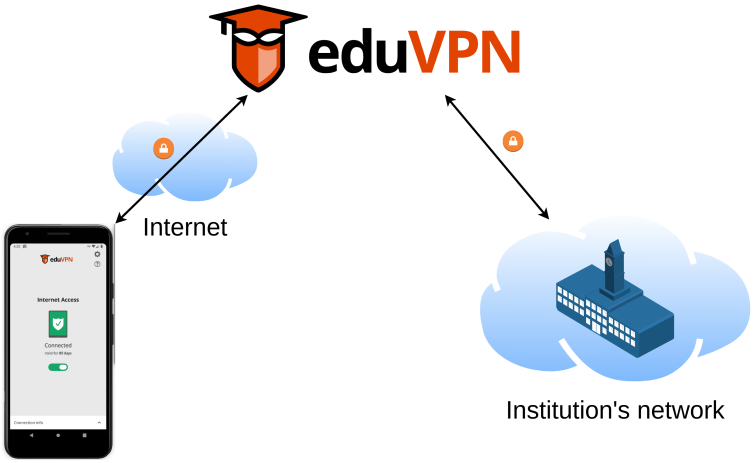

Figure 4: Visualization of a connection to an institution using the eduVPN Android App

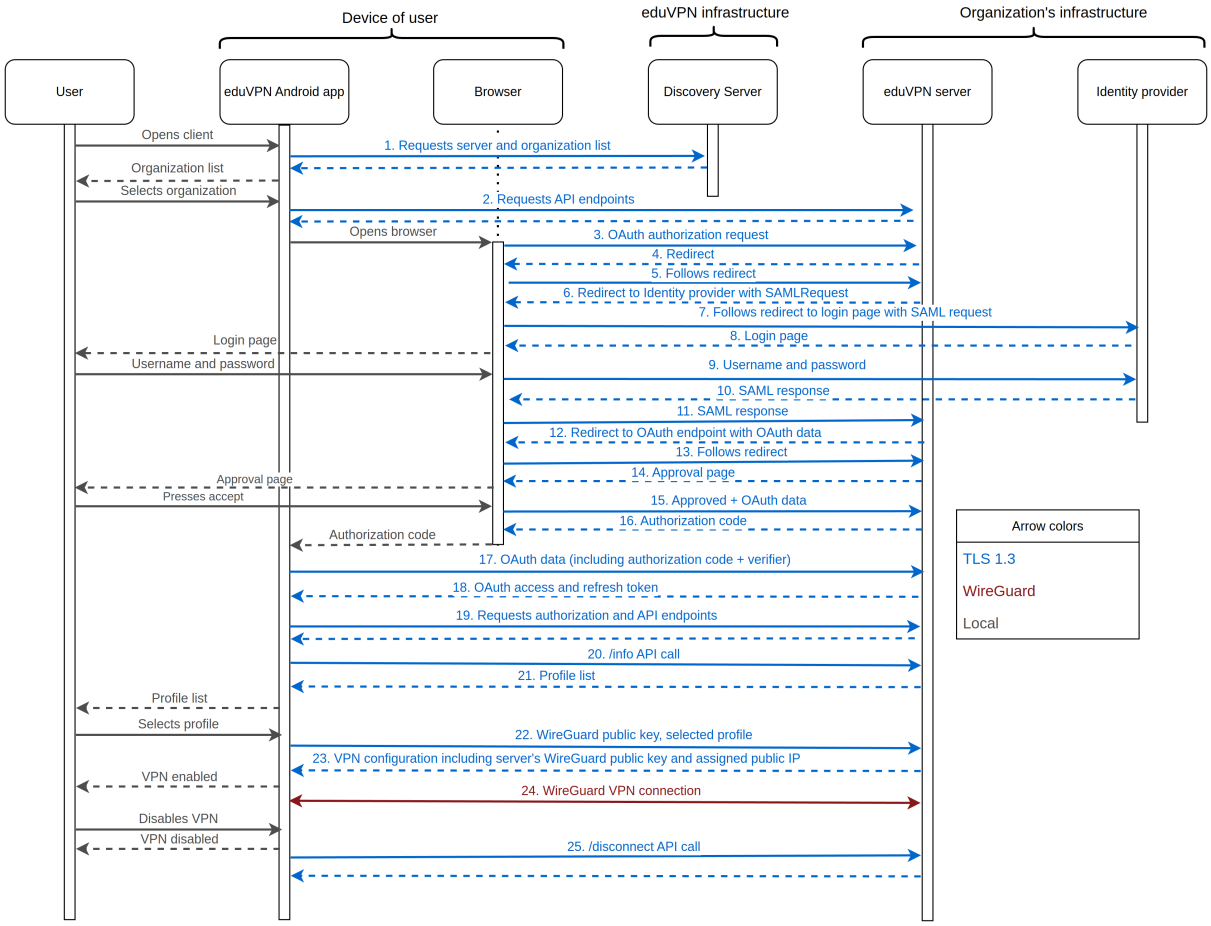

Figure 5: Sequence diagram of the data flow between components when connecting using WireGuard using the eduVPN Android client

Figure 5 contains multiple entities:

- The user.
- The eduVPN Android app.
- The browser on the phone of the user. We assume this browser supports TLS 1.3.
- The Discover Server. This server is used by the eduVPN clients to retrieve a list of known eduVPN servers. The user can then easily search for and select his organization.
- The eduVPN server. This is the VPN server which is managed by the organization.
- The identity provider. This is the identity provider used by the organization. This could be for example Active Directory or an LDAP server.

As can be seen in Figure 5, there are 2 protocols in use that protect data: TLS 1.3 and WireGuard.

6 What data should be protected against future decryption?

An overview of the connection flow and the data send can be seen in Figure 5. For all data has been determined if this data is confidential and for how long. The results can be seen in Table 1.

Table 1: Data send in eduVPN and its confidentiality

Number	Data	Confidential	Confidential time
1	Organization and corresponding server list	✗	
2	API endpoints	✗	
3	OAuth authorization request	✗	
4-5	Redirect to SAML endpoint	✗	
6	Redirect to Identity provider with SAML request	✗	
7	Redirect to Identity provider with SAML request	✗	
8	Login page	✗	
9	Username	✓	Forever
9	Password	✓	Until password change
10-11	SAML response	✓ [11] [1]	8 hours (by default)
12-13	Redirect to OAuth endpoint with OAuth data	✗	
14	Approval page	✗	
15	Approved + OAuth data	✗	
16	OAuth Authorization code	✗	
17	OAuth Verifier	✓	Until first use, this request

Number	Data	Confidential	Confidential time
17	OAuth data excluding verifier	✗	
18	OAuth access token	✓	1 hour (by default)
18	Refresh token	✓	90 days (by default) or until first use
19	Authorization and API endpoints	✗	
20-21	Profile list	✗	
22	WireGuard public key, selected profile	✗	
23	Assigned public IP	✓	Forever
23	WireGuard configuration excluding public IP	✗	
24	WireGuard VPN connection	✓	Forever
25	Disconnect	✗	

Although it is hard to estimate when exactly a quantum computer will be build that can break existing cryptography, if we assume it will be here in 5 years, we can determine which data should be protected.

As can be seen in Table 1, the following data is confidential but not long enough confidential to need protection against future decryption:

- Password: A password is only confidential until the user changes this password. To protect the password, a user should change his password after it can be sent over a post-quantum connection. We assume the user will change his password then.
- SAML response: The SAML response expires after 8 hours (by default) and can not be used after that time.
- OAuth verifier: As soon as the OAuth verifier is sent over the TLS connection, it is retrieved by the server and becomes unusable.
- OAuth access token: OAuth access tokens expire after 1 hour and can not be used after expiry.
- OAuth refresh token: OAuth refresh tokens can not be re-used, after it is used, it becomes invalid. If it is not used, it becomes invalid after 90 days.

As can be seen in Table 1, we would like to protect the following data forever:

- Username
- Assigned public IP
- WireGuard VPN connection

The username and password is sent between the browser and an identity provider. This is for example a UvA student logging in with his UvA account to a Microsoft login page. This interaction does not involve any systems that eduVPN controls, therefore we will leave the username out-of-scope.

[1]We assume the SAML response only contains a pseudonym of the user. This might not be the case on all eduVPN deployments.

7 How can confidential data be protected against decryption by quantum computers?

As determined in section 6, both the data send over the VPN connection and the IP that is assigned to the user is confidential data that should be protected. To answer why this confidential data is currently not protected, we will first analyze why the protocols that should protect confidential data do not protect against quantum computers. Then we will describe possible solutions.

7.1 Post-quantum analysis of protocols in eduVPN

To determine if confidential data is currently protected against quantum computers, the protocols used by eduVPN that protect confidential data, TLS and WireGuard, will be analyzed.

7.1.1 What is quantum secure?

We define a cryptographic primitive or a cryptographic protocol as quantum secure if it is at least as hard to crack as AES-128, with or without a quantum computer. This is the first security level as defined by NIST, NIST defines level 1 as[12]:

> Any attack that breaks the relevant security definition must require computational resources comparable to or greater than those required for key search on a block cipher with a 128-bit key (e.g. AES128).

Level 5, the highest security level, has the same definition but with a 256 bit key (e.g. AES 256)[12].

We will also refer to quantum secure as post-quantum.

7.1.2 WireGuard

WireGuard is the VPN protocol used to connect to the organization's network. Using the previously established TLS connection to the eduVPN server, the public keys used for the WireGuard connection are exchanged.

WireGuard uses the following cryptographic primitives[13]:

Table 2: Cryptographic primitives in WireGuard

Name	Used for	Quantum secure	Necessary against future decryption by quantum computer
ECDHE[2]	Key exchange	✗ [14–16]	✓

Name	Used for	Quantum secure	Necessary against future decryption by quantum computer
ChaCha20 (256 bit)	Symmetric encryption	✓ [16] [3] (level 5)	✓
Poly1305 (128 bit)	MAC[4]	✓ [16] (level 5)	✗
BLAKE2s	Hashing and HMAC[5]	Assumed secure	✗
SipHash	Hash table	Assumed secure	✗
HKDF3	Key derivation	Assumed secure	✗

Assumed secure means no source was found claiming either secure or insecure with regards to quantum computers.

If the key-exchange or the symmetric encryption can be broken by an attacker, data can be decrypted. As can be seen from Table 2, the only cryptographic primitive that is problematic is ECDHE. ECDHE is necessary to prevent decryption but can be broken by a quantum computer.

7.1.3 TLS 1.3

TLS is the cryptographic protocol used by for example your web browser to securely connect to websites. Contrary to WireGuard, it has crypto-agility, meaning it supports and can switch between different cryptographic primitives.

TLS provides confidentiality and authentication using different different cryptographic primitives.

7.1.3.1 Authentication Authentication in TLS is provided by signatures. Different signature algorithms can be used in TLS[17]:

- RSA
- ECDSA
- EdDSA

All three signature algorithms are not quantum secure[16]. Once there is a sufficient quantum computer, TLS 1.3 signatures could be forged.

All three signature algorithms are assumed to be secure against attacks by classical computers. TLS authentication is only necessary against an active MiTM. As determined in our attack model, we do not aim to protect against an active MiTM by a quantum

[2]Elliptic-curve Diffie–Hellman Ephemeral. WireGuard uses ECDHE with Curve25519.
[3]The source only mentions Salsa20, we assume it has at least the same security as its successor ChaCha20.
[4]Message authentication code.
[5]Hash-based message authentication code.

computer. The authentication provided by TLS is therefore good enough under our attack model. It does thus not have to be replaced by a post-quantum version.

7.1.3.2 Confidentiality and integrity

Confidentiality and integrity in TLS are provided by different cryptographic primitives. Because of the crypto agility of TLS, the actually used cryptography can differ depending on the client and server.

Both the eduVPN server and the Android client[18] support the following TLS 1.3 cipher suites:

- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256

The server supports more cipher suites, but those are not enabled by default, and since the Android client does not support it, will not be used.

Table 3: Cryptographic primitives in TLS 1.3 cipher suites.

Name	Used for	Quantum secure	Necessary against future decryption by quantum computer
AES (128 bit)	Symmetric encryption	✓ [12] (level 1)	✓
AES (256 bit)	Symmetric encryption	✓ [12] (level 5)	✓
ChaCha20 (256 bit)	Symmetric encryption	✓ [16] [6] (level 5)	✓
GCM	MAC[7]	✓ [16] (level 5)	✗
Poly1305 (128 bit)	MAC[8]	✓ [16] (level 5)	✗
SHA256	Hashing	✓ [12] (level 2)	✗
SHA384	Hashing	✓ [12] (level 4)	✗

Since TLS 1.3, the key exchange method is not part of the cipher suite anymore. The key exchange is necessary against future decryption by quantum computers. TLS 1.3 supports the following key exchange methods[17]:

Table 4: Key-exchange methods in TLS 1.3

Name	Quantum secure
ECDHE[9]	✗ [14–16]
DHE[10]	✗ [16]

[6] The source only mentions Salsa20, we assume it has at least the same security as its successor ChaCha20.
[7] Message authentication code.
[8] Message authentication code.

Name	Quantum secure
PSK-only	Depends on PSK
PSK with (EC)DH	Depends on PSK

As can be seen from Table 4, the (EC)DHE key exchange is problematic. Using a PSK (Pre-Shared Key) option is save, but this option does not actually do a key exchange. The PSK option uses a key that has to be securely exchanged outside of the TLS connection and can thus not be used as a drop-in replacement for EC(DHE).

7.2 Post-quantum TLS

To secure the complete TLS connection, a post-quantum key exchange has to be done. This can be done over the plain as described in section 7.2.1 or over an existing TLS connection as described in section 7.2.2.

7.2.1 Post-quantum key exchange in TLS

Instead of Diffie-Hellman we could replace Diffie-Hellman with a post-quantum KEM. To prevent classical computers from being able to decrypt the TLS connection in case the KEM is discovered to be insecure, a post-quantum KEM should be used alongside Diffie-Hellman. This could be implemented using an OpenSSL provider or by replacing the TLS implementation.

7.2.1.1 OpenSSL provider The eduVPN servers use an Apache module to provide TLS functionality. This Apache module uses OpenSSL to provide TLS. OpenSSL providers can add cryptography dynamically using a module. Creating an OpenSSL providers that adds post-quantum KEMs to OpenSSL is possible with OpenSSL version 3 and has been done by the Open Quantum Safe project[19]. Unfortunately, as of 13 February 2023, only 8% of eduVPN servers use a Linux distribution with OpenSSL 3[1]. Such a solution can thus not be rolled out any time soon.

7.2.1.2 Replacing TLS implementation The implementation of TLS used on the client and on the server could be modified to also do a post-quantum key exchange alongside Diffie-Hellman, as done in [8] and [9]. The problem with this solution is that it requires to replace the existing TLS implementations with our own. On the server this means that we are now responsible for updating security critical software, instead of the OS vendor (i.e. Debian or Red Hat). This also means that in case a new version of the TLS implementation is released to fix a security issue, we now have to update our modified implementation before the eduVPN servers can be updated. This increases the time in which a security issue can be abused, and is thus undesirable.

[9]Elliptic-curve Diffie–Hellman Ephemeral. WireGuard uses ECDHE with Curve25519.
[10]Finite field Diffie-Hellman Ephemeral.

Another problem with adding a post-quantum key exchange to TLS is that routers or other middleboxes might refuse the connection because it does not understand the modified protocol[20]. Such middleboxes caused delays in the rollout of TLS 1.3[21].

7.2.2 Post-quantum key exchange over TLS

As determined in 6.1, only the key-exchange part of TLS does not provide the necessary security. Both TLS and WireGuard specify a pre-shared key mode in their specification[13]. If we can exchange a key in a way that a quantum computer can not recover this key, we can set this key as the pre-shared key for TLS (TLS-PSK). TLS will then derive the actual key used for symmetric encryption using a key derivation function from parameters including the pre-shared key.

To prevent problems with middleboxes and to prevent an active MiTM by a classical computer, we can do a key exchange over a normal TLS connection. We can then use the new shared secret as the key used for a new TLS connection.

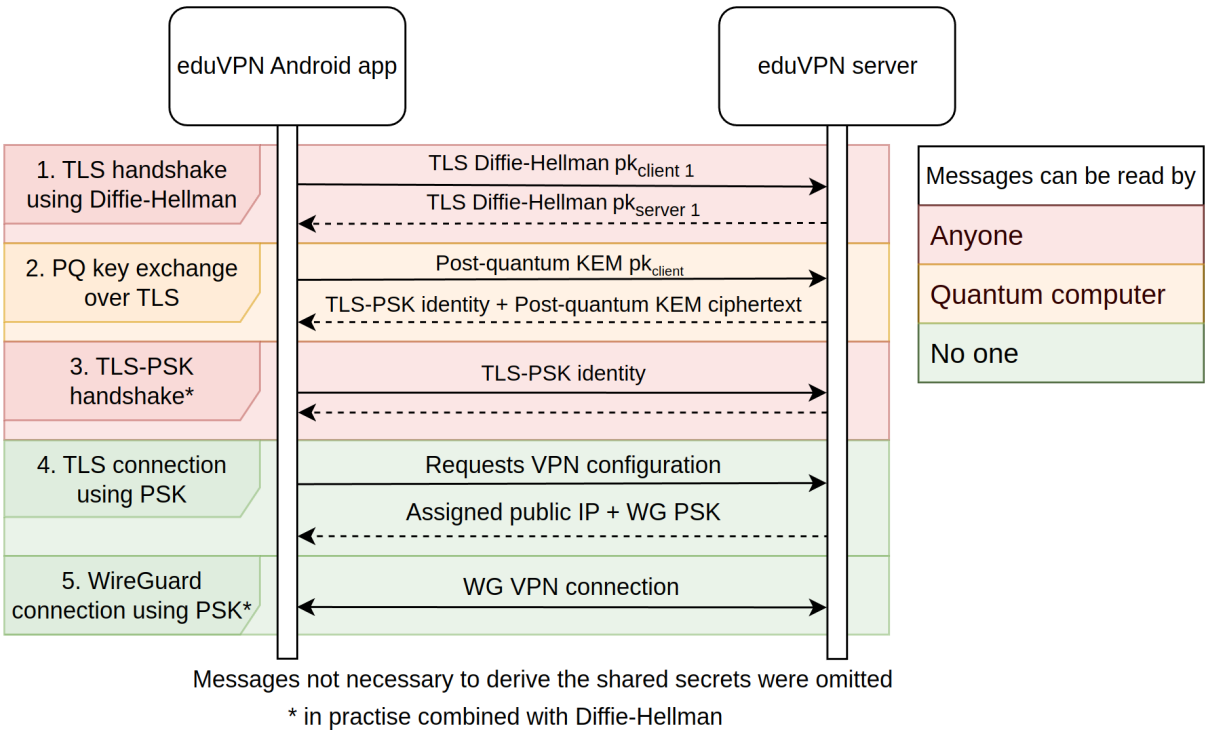

Figure 6: Messages used to securely share a pre-shared key used for the WireGuard VPN connection using TLS-PSK

As can be seen in Figure 6, securely exchanging a key that can be used for WireGuard consists of the following steps:

1. Establish a TLS connection to the server.
2. Do a post-quantum key exchange over the TLS connection (for example using Kyber).
3. Use the shared secret from the post-quantum key exchange in the TLS connection.

4. Use the secure TLS connection to share a randomly generated pre-shared key.
5. Use the pre-shared key send in the previous step for the WireGuard VPN connection.

With both WireGuard and TLS, the PSK is not directly used, it is combined with another shared secret established using an ECDH key exchange. In the case of WireGuard, this is required by the protocol. In the case of TLS, both are allowed by the protocol, but the TLS implementation used on Android only supports the TLS-PSK variant that also does an ECDH key exchange.

The mechanism used to establish a post-quantum TLS connection (step 1, 2 and 3) are not eduVPN specific. This mechanism can thus be used by anyone.

7.2.2.1 Implementation eduVPN uses Apache as web server. To add this functionality to Apache, an Apache module was created that handles requests to `/.well-known/qkem`. The module uses Kyber as post-quantum KEM. A client can send his Kyber public key to this endpoint and the server will return a ciphertext (step 2). The client can decrypt this ciphertext to recover the shared secret, which can then be used as TLS pre-shared key (step 3). When a client connects, it sends the PSK identity that it received from the server to the server in a new TLS connection, the server can then lookup the shared secret.

In case the server does not have the qkem Apache extension, a call to `/.well-kwown/qkem` will return a 404 not found, and the client can continue using the existing TLS connection.

The source code of the Apache module has been published.

On both the server and the client, some issues had to be resolved to get TLS-PSK working.

7.2.2.1.1 Server implementation challenges On the server side, a workaround is necessary to prevent Apache from terminating the connection. This is caused by OpenSSL setting a property on the PSK connection that client verification was not done[22]. This causes Apache to terminate the connection. Unfortunately, the workaround does not use the OpenSSL public API[23], which means the workaround can break in case of an OpenSSL update, causing Apache to terminate TLS-PSK connections. When a security update for OpenSSL is released that is incompatible with the Apache module because of the non-public API, the module would have to be disabled or the OpenSSL update delayed. Both situations are not desirable.

7.2.2.1.2 Client implementation challenges On the client side, the TLS library had to be replaced by one that supports TLS-PSK (Bouncy Castle instead of Conscrypt). Even though it supported TLS-PSK, the library still had to be modified to be able to set a PSK.

The library used to make HTTP calls (which uses the TLS library) needed multiple workarounds to prevent it from crashing. These crashes occur when the library tries to verify the validity of TLS certificates. TLS-PSK does not use certificates, which causes the application to crash.

7.3 Post-quantum VPN configuration

Securing the complete TLS connection, as described in section 7.2, is possible, but might break on an OpenSSL update. Therefore, a different solution was developed which securely exchanges the VPN configuration which includes the assigned public IP and the WireGuard VPN connection.

First, the design is explained. Then the choice of post-quantum KEM is discussed. We then determine how multiple key exchanges can be used and we discuss the symmetric encryption. Lastly, we discuss the choice of post-quantum library.

7.3.1 Design

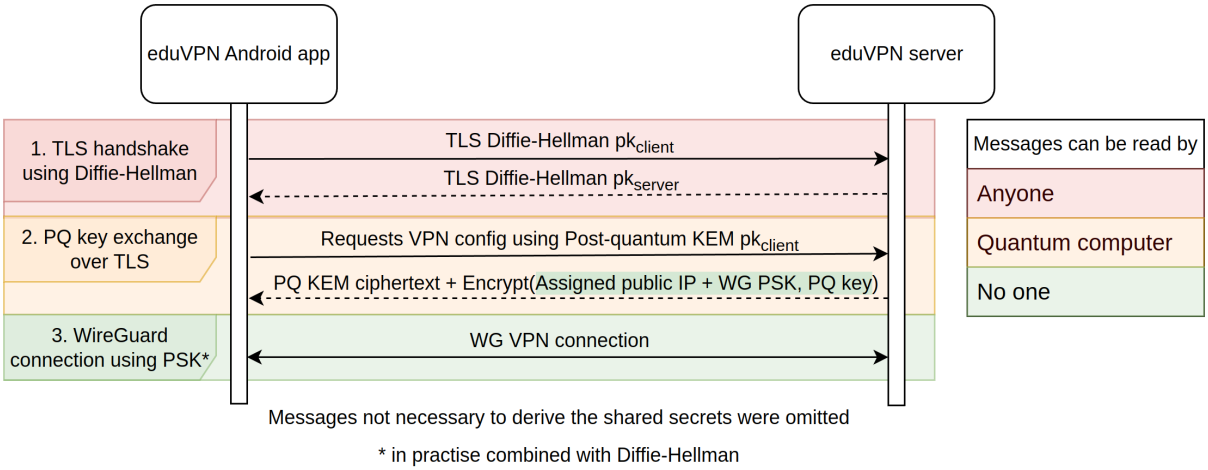

Figure 7: Messages used to securely share the PSK used for the WireGuard VPN connection using an API call

As can be seen in Figure 7, securely exchanging a VPN configuration consists of the following steps:

1. Establish a TLS connection to the server.
2. Do a post-quantum key exchange over the TLS connection (for example using Kyber). Protect the VPN configuration including the public IP and the WireGuard pre-shared key using the shared secret from the post-quantum key exchange.
3. Use the pre-shared key for the WireGuard VPN connection.

As can be seen in step 2, on top of the symmetric encryption used by TLS, another layer of symmetric encryption was added to secure the VPN configuration.

7.3.2 Choice of post-quantum KEM

In 2016, NIST announced a competetition for post-quantum cryptography[24]. The selected algorithms will be standardized by NIST. NIST called for proposals for digital signature, public-key encryption and key-exchange algorithms. 69 public-key encryption algorithms

where published for round 1. In July 2022 NIST selected Kyber to be standardized and announced a 4th round to standardize one of[6]:

- BIKE
- Classic McEliece
- HQC
- SIKE

In august 2022, SIKE was broken[7] and will thus not be standardized. The fact that SIKE has only been broken in round 4 of the NIST competition raises questions about the security of other new KEMs. Post-quantum KEMs, especially lattice based KEMs, are considered risky[25]. These post-quantum KEMs have been around for shorter than RSA and AES and have thus received less security analysis. To not depend on the security of one post-quantum KEM, we would like to use 2 KEMs. If one breaks, data is still protected as long as the other KEM is secure. Preferably, the used KEMs should not rely on the same mathematical problem.

The public keys of Classic McEliece are at least 1MB for the level 5 parameters. For comparison, Kyber public keys for level 5 are only 1568 bytes. In our protocol the public keys have to be send over the network. Big public keys will cause a delay on slow connections which makes Classic McEliece unfit for our purpose.

In our implementation we have selected BIKE as the second KEM, because the HQC implementation in the library we use on the server is not up-to-date.

In case one of the KEMs is broken, the broken KEM should be replaced with another one. To facility this cryptographic agility, the server lists the KEMs it supports and then the client can select the KEMs that it wants to use. In the current implementation Kyber, BIKE and Frodokem are supported as those are the only KEMs that both the library used on the client (Bouncy Castle) and the library used on the server (liboqs) support. The server thus sends a list of the supported KEMs to the client. The client then decides which KEMs and how many it is going to use, in our implementation Kyber and BIKE. With an update of either the client or the server, a broken KEM can be disabled and in this case Frodokem will then automatically replace the broken KEM. Note that right now one of the three KEMs can be removed from the server without consequence because all clients will either support no post-quantum crypto (old clients) or all three (new clients). If in the future another KEM is added, old KEMs should not be removed from the server since old clients might not support the new KEM.

In case all used KEMs are broken, our connection is still secure against attacks by classical computers, this protection is provided by TLS.

We do not use the 90s variant of Kyber as this variant will not be standardized by NIST.

7.3.3 Combining shared secrets

Both TLS and WireGuard only do one key-exchange. Our protocol does two. When performing multiple key exchanges, the result will be multiple keys. We can only use one key for symmetric encryption, therefore the keys have to be combined. As explained in section 7.3.2, we want our system to still be secure if one key exchange is broken, therefore we need to determine how we can securely combine the keys.

Giacon et al. show possible ways to combine keys from KEMs in [26]. The most simple one shown is the XOR combiner. Giacon et al. show that the XOR combiner retains CPA security but does not retain CCA security.

What this means is that the XOR combiner is not safe against an active MiTM, but is safe against a passive attacker. We assume TLS can not be broken right now and thus TLS provides the authentication and integrity which makes an active attack against the post-quantum key exchange impossible. We can therefore safely use the XOR combiner.

Using a CCA secure combiner will not help against an active MiTM, because an attacker could just remove the public keys from the VPN configuration request that the clients sends. This would result in the server not encrypting the configuration because of backward-compatibility.

Unlike with Diffie-Hellman, we do not need to use a KDF on the shared secret. The shared secret of the KEMS, Kyber, BIKE, HQC and FrodoKEM can directly be used for symmetric encryption[[4],Appendix;[27],2.3.2;[28],2.2.11;].

Note that Kyber, BIKE and HQC produce a shared secret of 256 bits. HQC produces a shared secret of 512 bits. To XOR the HQC shared secret with another KEM producing a 256 bit shared key, we can use the first 256 bits of the 512 bits shared secret. This can be done without losing security, as the shared secret from HQC is a 512 bit output of the SHAKE256 hashing algorithm.

7.3.4 Symmetric encryption

After exchanging a 256 bit key (PQ key in Figure 7), we still have to somehow protect our data with this key. This can be achieved by using a symmetric encryption algorithm.

To achieve level 5 security, we will need a symmetric encryption algorithm that has security level 5. Both AES-256 and ChaCha20 would be suitable, they both provide level 5 security[12,16]. The implementation uses ChaCha20. Since integrity and authentication is already provided by TLS, we do not need a message authentication code. ChaCha20 without a message authentication code is not provided by the public APIs of the cryptography libraries already in use on both the client and the server, therefore we will still use an authentication code by using ChaCha20-Poly1305.

ChaCha20 requires both a key and a nonce for encryption. The nonce does not have to be random, but should never be reused with the same key. Since every encapsulation of a public key generates a new shared secret, a shared secret will only ever be used once. Combining multiple shared secrets still produces a key that will only ever be used once. Therefore, instead of choosing a nonce and sending it over to the other party, we can just fix the nonce to a known value. The nonce has to be 12 bytes, therefore the value `eduVPNeduVPN` was chosen for the nonce.

7.3.5 Choice of post-quantum library

The implementation of post-quantum KEMs can be provided by multiple libraries. The client and server are written in different programming languages, which result in a different choice for the post-quantum library on the client and server.

Table 5: Considered post-quantum cryptography libraries

Name	Maintainer	API	Kyber	BIKE	HQC	FrodoKEM
liboqs	Open Quantum Safe	C	✓	✓	✗ [11]	✓
Bouncy Castle	Legion of the Bouncy Castle	Java	✓	✓	✓	✓
CIRCL	Cloudflare	Go	✓	✗	✗	✓

Table 5 shows the libraries with post-quantum KEM implementations that have been considered. KEM support as of 9 February 2023.

The Bouncy Castle library exposes a Java API which can be directly used in the Android App. Although a Java wrapper for liboqs exists, at the time of writing, 11 February 2023, it is outdated. On the client we thus chose to use Bouncy Castle.

The server code that handles API requests is written in PHP. No PHP implementation or wrapper for post-quantum KEMs could be found, therefore the PHP Foreign Function Interface (FFI) was used to call the liboqs C API. Even though the exposed API is C, for all listed KEMs in the table, on x86_64, an optimized implementation written in assembly is used if the CPU supports the required CPU extensions.

8 Conclusions

An adversary that stores data now is able to decrypt this data later. To protect against this attack, we determined what confidential data is sent between the eduVPN client and server and how long this data should be protected. We then determined how this data can be protected against decryption by quantum computers.

To protect against future decryption, the protocols in use (TLS and WireGuard) do not have to be replaced. Both WireGuard and TLS can be protected against future decryption by quantum computers by using a pre-shared key that quantum computers can not recover. Exchanging this pre-shared key can be securely done by a post-quantum key exchange over a TLS connection. The shared secret can then be used to encrypt confidential data and the WireGuard pre-shared key. The WireGuard pre-shared key can be used to protect the WireGuard connection.

We presented two methods:

- Exchanging a key by calling an API implemented in the web server and using this key as the pre-shared key for TLS. This protects the complete TLS connection. This is possible on Android and with an Apache server using an Apache module. The implemented Apache module is however not guaranteed to keep working in case of OpenSSL updates.
- Exchanging a key over the eduVPN API and using this key to encrypt confidential data and a WireGuard pre-shared key.

[11] KEM implementation present, but outdated version of KEM.

Both methods have been implemented and work. However, there are still improvements that could be done. In the next section, Chapter 9: Future work, we will discuss these improvements including OpenVPN support and making sure different post-quantum library implementations are compatible with each other.

9 Future work

9.1 Post-quantum library compatibility evaluation

On the client and the server two different implementations of the same KEMs are used. Although the implementations are based on the same specification, there could be an edge case with a bug causing the implementations to be incompatible with each other. To detect this kind of bugs, a setup could be created in which the implementations are tested with each other using a lot of different public keys. The client implementation creates a public key, from which the server implementation generates a ciphertext, which the client implementation then decapsulates. The shared secret should then be compared. To find bugs quicker, the normally random input could be replaced by input from a guided fuzzer.

Another method to prevent incompatibilities is to use the same library on both the client an server. This could be achieved by updating the Java wrapper of liboqs and using it on the client.

9.2 Client support

Besides a client for Android, eduVPN also has clients for Linux, Windows, macOS and iOS. Support for the protocol presented in this paper should also be added to those clients.

9.3 OpenVPN

Administrators deploying eduVPN can choose to enable WireGuard and/or OpenVPN. This paper only considers WireGuard, more work is necessary to protect OpenVPN connections.

9.4 Formal verification

A formal verification of a protocol would give more confidence in the security of that protocol. WireGuard has been formally verified using Tamarin [29], the same should be done for the protocols in this paper.

9.5 Active MiTM by quantum computer

The protocol in this paper does not protect against an active MiTM by a quantum computer. Further research should be done to protect against such an attack. By the time a quantum computer can break currently used cryptography, a new version of TLS might be post-quantum, which can then replace the key-exchange mechanism presented in this paper.

10 Acknowledgments

I would like to thank Rogier Spoor for supervising the project and for feedback on earlier versions of this paper and presentation. I would like to thank Jeroen Wijenbergh and François Kooman for insightful discussions and for feedback on earlier versions of this paper and presentation.

References

1. *Server status*. (n.d.). Retrieved January 4, 2023, from https://status.eduvpn.org
2. Shor, P. W. (1994). Algorithms for quantum computation: Discrete logarithms and factoring. *Proceedings 35th Annual Symposium on Foundations of Computer Science*. https://doi.org/10.1109/sfcs.1994.365700
3. Kooman, F. (2022, October 19). *eduVPN 3.0*. https://argon.tuxed.net/fkooman/presentations/Kooman-DFN-Betriebstagung-20221019.pdf
4. Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J. M., Schwabe, P., Seiler, G., & Stehlé, D. (2017). *CRYSTALS – kyber: A CCA-secure module-lattice-based KEM*. Cryptology ePrint Archive, Paper 2017/634. https://doi.org/10.1109/EuroSP.2018.00032
5. Hülsing, A., Ning, K.-C., Schwabe, P., Weber, F., & Zimmermann, P. R. (2020). *Post-quantum WireGuard*. Cryptology ePrint Archive, Paper 2020/379. https://eprint.iacr.org/2020/379
6. NIST. (2022, July 5). *Announcing PQC candidates to be standardized, plus fourth round candidates | CSRC*. https://csrc.nist.gov/News/2022/pqc-candidates-to-be-standardized-and-round-4
7. Castryck, W., & Decru, T. (2022). *An efficient key recovery attack on SIDH (preliminary version)*. Cryptology ePrint Archive, Paper 2022/975. https://eprint.iacr.org/2022/975
8. Crockett, E., Paquin, C., & Stebila, D. (2019). *Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH*. Cryptology ePrint Archive, Paper 2019/858. https://eprint.iacr.org/2019/858
9. Schwabe, P., Stebila, D., & Wiggers, T. (2020). Post-quantum TLS without handshake signatures. *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 1461–1480. https://doi.org/10.1145/3372297.3423350
10. Appelbaum, J., Martindale, C., & Wu, P. (2019). *Tiny WireGuard tweak*. Cryptology ePrint Archive, Paper 2019/482. https://eprint.iacr.org/2019/482

11. Hirsch, Philpott, & Maler. (2015, March 15). *Security and privacy considerations for the OASIS security assertion markup language (SAML) V2.0*. https://docs.oasis-open.org/security/saml/v2.0/saml-sec-consider-2.0-os.pdf
12. NIST. (2017, January 3). *Post-quantum cryptography*. https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/evaluation-criteria/security-(evaluation-criteria)
13. Donenfeld, J. (2020, May). *WireGuard: Next generation kernel network tunnel*. https://www.wireguard.com/papers/wireguard.pdf
14. Roetteler, M., Naehrig, M., Svore, K. M., & Lauter, K. E. (2017). Quantum resource estimates for computing elliptic curve discrete logarithms. *ASIACRYPT (2)*, 241–270. https://doi.org/10.1007/978-3-319-70697-9_9
15. Grassl, M., Langenberg, B., Roetteler, M., & Steinwandt, R. (2016). Applying grover's algorithm to AES: Quantum resource estimates. *Post-Quantum Cryptography*, 29–43. https://doi.org/10.1007/978-3-319-29360-8_3
16. Bernstein, D. J., & Lange, T. (2017). Post-quantum cryptography - dealing with the fallout of physics success. *IACR Cryptology ePrint Archive*, *2017*, 314–314. https://pure.tue.nl/ws/files/92552870/bernpost2017.pdf
17. Rescorla. (2018, August). *RFC 8446: The transport layer security (TLS) protocol version 1.3*. https://www.rfc-editor.org/rfc/rfc8446
18. google, & Google. (2020, November 16). *Conscrypt/CAPABILITIES.md at master · google/conscrypt*. https://github.com/google/conscrypt/blob/master/CAPABILITIES.md
19. Open Quantum Safe. (2023, February 7). *GitHub - open-quantum-safe/oqs-provider: OpenSSL 3 provider containing post-quantum algorithms*. https://github.com/open-quantum-safe/oqs-provider
20. Celi, S. (2022, January 21). *The post-quantum state: A taxonomy of challenges*. https://blog.cloudflare.com/post-quantum-taxonomy/
21. Sullivan, N. (2017, December 26). *Why TLS 1.3 isn't in browsers yet*. https://blog.cloudflare.com/why-tls-1-3-isnt-in-browsers-yet/
22. *Openssl/ssl_sess.c at f6e921b416758598774b0a6c06d3f3bc9c5fbaf7 · openssl/openssl*. (2020, December 8). https://github.com/openssl/openssl/blob/f6e921b416758598774b0a6c06d3f3bc9c5fbaf7/ssl/ssl_sess.c#L78
23. Aquina, N. (2023, February 2). *Qkem/mod_qkem.c at 612f45470818219544a9d8f209a6e46a3c334e54 · fantostisch/qkem*. https://github.com/fantostisch/qkem/blob/612f45470818219544a9d8f209a6e46a3c334e54/mod_qkem.c#L118
24. Kimball. (2016, December 20). *Announcing request for nominations for public-key post-quantum cryptographic algorithms*. https://www.federalregister.gov/documents/2016/12/20/2016-30615/announcing-request-for-nominations-for-public-key-post-quantum-cryptographic-algorithms
25. NTRU Prime Risk-Management Team. (2021, March 1). *Risks of lattice KEMs*. https://ntruprime.cr.yp.to/latticerisks-20211031.pdf
26. Giacon, F., Heuer, F., & Poettering, B. (2018). KEM combiners. *Public-Key Cryptography – PKC 2018*, *10769*, 190–218. https://doi.org/10.1007/978-3-319-76578-5_7
27. Melchor, C. A., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.-C., Gaborit, P., Persichetti, E., Zémor, G., Bos, Dion, Lacon, Robert, & Véron. (2022, October 1). *Hamming quasi-cyclic (HQC) fourth round version*. https://csrc.nist.gov/csrc/media/Projects/post-quantum-cryptography/documents/round-

4/submissions/HQC-Round4.zip
28. Alkim, Naehrig, Bos, Nikolaenko, Ducas, Peikert, Longa, Raghunathan, & Mironov. (2021, June 4). *FrodoKEM learning with errors key encapsulation.* https://frodokem.org/files/FrodoKEM-specification-20210604.pdf
29. Donenfeld, & Milner. (2018, June 7). *Formal verification of the WireGuard protocol.* https://www.wireguard.com/papers/wireguard-formal-verification.pdf