Your Title here

Jane Doe, Degree

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science (Data Science)

Supervisor: Sean Doe

August 2021

Your Title here

Jane Doe, Master of Science in Computer Science

University of Dublin, Trinity College, 2021

Supervisor: Sean Doe

This paper introduces post-quantum WireGuard, a post-quantum instantiation of the WireGuard protocol. Notably, this variant does not merely put focus on few aspect of security such as post-quantum confidentiality or forward secrecy, as commonly pursued by earlier research at designing post-quantum protocols. Instead, it also offer bilateral post-quantum authentication with moderate cost of computation and communication. To accomplish this, we substitute the existing Elliptic Curve Diffie-Hellman key exchange with a new asymmetric crypto primitive:, namely key-encapsulation mechanisms (KEMs). This thesis explores various combinations of different KEMs to alter existing WireGuard implementations. We propose three incremental modifications, each providing different level of security against quantum threats.

Acknowledgments

Thank you Mum & Dad.

Jane Doe

University of Dublin, Trinity College
August 2021

Contents

List of Tables

List of Figures

Chapter 1

Introduction

WireGuard is an innovative secure network protocol that functions as a virtual network interface within the Linux kernel in most cases. Its primary goal is to serve as a drop-in replacement for similar secure network protocol including IPsec and OpenVPN, providing enhanced security, better performance and improved friendliness. The secure tunnel is established by binding the peer's public key with its corresponding source address. The design of WireGuard is directly deriving from the Noise framework, bearing in mind the problem of pervasive surveillance and active tampering. Also, to get over the bandwidth and computational cost , WireGuard streamline the handshake process, utilizing a single round trip to achieve security of both perfect forward secrecy (PFS) and key-compromise impersonation (K-CI) resistance. But, all these security properties only holds in the classical computer model. With the arrival of the quantum computer, it is necessary to reimage of WireGuard Diffie-Hellman (DH) style handshake to use key encapsulation mechanisms (KEMs) for implicit authentication, rather than pre-shared static key for explicit authentication. This research proposes a more efficient and safer way to integrate the kem algorithm into the WireGuard protocol, in a similar manner to the kem-tls and kemtlspdk and achieve both post-quantum authentication and stronger forward secrecy. The research also instantiate with a variety of post-quantum secure KEMs based on fundamentally different mathematical assumptions.

1.1 Motivation

1.1.1 WireGuard

WireGuard is an upcoming project to replace IPSec with a newer more modern and secure VPN protocol. It mainly lives inside the kernel , but cross platform implementations are also available. It provides a very simple and novel interface for setting up secure en-

crypted network tunnels. All the cryptography primitives are cutting edge: Curve25519, ChaCha20, and Poly1305 and the complexity, as well as the sheer amount of code is ignorable. Unlike TLS, which offers backward compatibility and cryptography agility. The WireGuard hard-codes all the cryptography primitive choices. The decision to hard-code its cryptography primitives was driven by a desire for simplicity and consistency. It's one of the reasons why WireGuard has gained a reputation for performance and reliability. In the meantime, these primitives provide unparalleled degree of performance in the first place, introduce fewer dependencies and less complexity to manage which make it easier to ensure that the new design is secure and that all components of the protocol work correctly together. Furthermore, Its simplicity and minimalist design philosophy aligns well with the current state of post-quantum cryptography. Many post-quantum cryptography algorithms are still being tested and optimized, and are often more computationally intensive or require larger key sizes than their classical counterparts. Having a lightweight, efficient protocol like WireGuard as the starting point could be advantageous in balancing these new algorithms' overhead and maintaining a high-performance, secure VPN.

1.1.2 Post-Quantum Cryptography

In recent years, there has been a considerable amount of research on quantum computers which can solve specific mathematical problems that are difficult for classical computers by exploiting quantum phenomena. In the foreseeable future, the large-scale quantum computer probably will bring unprecedented speed and power in computing to reality. However, this quantum edge also poses serious problems which now is so-called **quantum apocalypse** - "store now, decrypt later" within the field of post-quantum cryptography. One of such threats lies in the domain of public key cryptography. When it comes time, the quantum computer could potentially break many of the cryptography systems, which were designed and secured under the premises of classical computing. It's crucial that we adopt a proactive approach and remain one step ahead of the potential risks. We can not afford to idle until the quantum computers begin dismantling today's public-key cryptography is running. Therefore, the goal of this project is to scrutinize the vulnerable part of the WireGuard and propose a new WireGuard which is post-quantum safe while still keeps the virtue of original WireGuard protocol. The necessity for quantum safe cryptography stems from the capability of Shor's algorithm's ability to efficiently resolve problems related to factorization and discrete logarithm which are the security backbones of **RSA**, **ECC** and **ECDSA** that rely on the **IFP** (integer factorization problem), the **DLP** (discrete logarithms problem) and the **ECDLP** (elliptic-curve discrete logarithm problem) respectively. By using Shor's algorithm, a k-bit (in binary)

number can be factored in time of order $O(k^3)$ using a quantum computer of 5k+1 qubits. As the NIST standardization process is ongoing [1][2], this standardization doesn't assert the supremacy of one suggestion over another. Given that the NIST standardization process has already chosen the **Selected Algorithms 2022** [3], the most secure transition method will be integrate the standardized Post-Quantum Cryptography algorithms instead of waiting for national bodies to finalize PQC algorithms in which case where the risk associated with quantum broken cryptography is not acceptable.

1.2 Goals

The major goal of this thesis is to add post-quantum cryptography into WireGuard, aiming to uphold perfect forward secrecy (PFS), secrecy, authenticity, and even security against active attacks. In the current version of WireGuard, there is an efficient approach [4] applying to the WireGuard protocol that can achieve transitional post-quantum security to the WireGuard via utilizing optional pre-shared key (PSK) value. This key, calculated independently of the key agreement protocol, could be used to initialize a PQ key exchange [4] In order to address this issue, it is necessary to integrate PQ key agreements directly into the WireGuard protocol. While addressing these primary goals, the secondary objectives revolve around maintaining optimal performance and usability. These aspects should be preserved as effectively as possible, even while achieving the main goals. Thus, this thesis strives to keep a balance between advancing WireGuard's security measures to confront quantum threats, without compromising its hallmark simplicity and performance.

1.3 Contribution

In this thesis, we introduce KEM-WireGuard, a generic post-quantum augmentation to the WireGuard protocol. This innovation not only improves the security design to keep secret against quantum attacks, as done in previous attempts at transitioning to post-quantum security, but also ambitiously aims for comprehensive post-quantum security, encompassing authentication as well. We also present the KEM-based instantiation of post-quantum WireGuard using Golang native cryptographic library - CIRCL (Cloudflare Interoperable Reusable Cryptographic Library). We focus on a coherent and cohesive implementation while considering the unique characteristics of different KEMs. We employ various combinations of KEMs to maximize the utilization of their unique performance and key size characteristics, with the ultimate goal of achieving optimal bandwidth and per-

formance. Our approach involves benchmarking and comparing various instantiations of the post-quantum WireGuard variants with previous implementations of PQ-WireGuard and the original WireGuard. A important consideration is to align as closely as possible to the original WireGuard protocol in terms of its security, performance metrics and handshake flow. Consequently, it should :

- Attain all the safety characteristics of WireGuard while also being resilient against attacks by a large-scale quantum computer
- Reach a firm decision on the cryptographic primitives explicitly, thereby eliminating the need for an cryptographic negotiation.
- Reduce the on-wire format size as well as the memory consumption.

The original WireGuard protocol heavily relies on the Elliptic Curve Diffie-Hellman key exchange, which is not a simple task to substitute with post-quantum ones. It's important to note that while the application of post-quantum algorithms in the WireGuard protocol may increase security margin , it's crucial to carefully evaluate their implementation and performance to ensure they meet the necessary standards for secure communication. Given that SIDH/SIKE can be compromised in classical polynomial time, it's definitely left out of the consideration. The only feasible post-quantum key exchange is CSIDH, but its security is relatively new and unproven. So we decided to follow the results of previous research on kem, modifying the WireGuard protocol to use interactive key-encapsulation mechanisms (KEMs) exclusively. [todo] The security of WireGuard is upheld by Donenfeld and Milner's symbolic proof and the computational proof provided by Dowling and Paterson. Although the symbolic proof encompasses a wider range of security properties and is computer verified, the computational proof, when correct, ensures robust security assurances as it relies on fewer idealized assumptions. We adapt both proofs to the PQ-WireGuard scenario, thereby maintaining WireGuard's level of security guarantees. During this process, we identify and rectify a few minor errors in the computational proof. To facilitate a standalone proof of the handshake, we incorporate an explicit key confirmation message into the PQ-WireGuard handshake as recommended.

1.4 Structure

Chapter 1: Introduction

1.1 Background and Motivation

1.2 Problem Statement

Chapter 2

State of the Art

2.1 Quantum Algorithms

2.2 Background

2.2.1 Shor's Algorithm

Shor's algorithm, introduced by mathematician Peter Shor in 1994, is a quantum algorithm which can efficiently solve the factoring problem (get p and q from $\boldsymbol{n = p * q}$) and the discrete logarithm problem (get e from $\boldsymbol{x = \mathbf{Mod}\,[g^e, p]}$). It has the potential to render all widely deployed key agreement and digital signature schemes, which are both critical to the security of the Internet, obsolete, including RSA and elliptic curve cryptography. The major application of Shor's algorithm lies in its ability to find the prime factors of large numbers, a task that is exceptionally difficult for classical computers to perform. Shor's algorithm, when run on a sufficiently powerful quantum computer, can factorize these large numbers exponentially faster than the best known classical algorithm. Consequences of Shor's algorithm in terms of its security, performance metrics and handshake flow:

- Quantum order-finding algorithm can be implemented in $O(n^3)$ quantum gate steps ($k = \log n$), namely within **bounded-error quantum polynomial time** (**BQP**).
- Factoring is solvable in quantum polynomial time.
- Modified Shor can also solve discrete logarithm problem, which means it totally breaks discrete log-based and elliptic curve cryptography.

Therefore ,the underlying basis of many modern public key cryptographic systems, including RSA, DSA, and elliptic-curve cryptography (ECC), rests on the difficulty of the

integer factorization or the discrete logarithm problem, will be dead in the presence of a large-scale quantum computer, thus providing a significant impetus for the development of post-quantum cryptography.

2.2.2 Grover's Algorithm

Grover's algorithm, devised by Lov Grover in 1996, is another significant quantum algorithm. It uses amplitude amplification to efficiently perform unstructured search to pinpoint the unique input for a black box function that leads to the corresponding output. Meanwhile, While a classical computer would need $O(n)$ steps to complete the search, Grover offers a quadratic speedup over classical algorithms for similar search problems. In cryptographic terms, Grover's algorithm can effectively halve the key length of symmetric cryptographic systems, reducing, for example, the security level of a 256-bit key to that of a 128-bit key in classical terms. The application of Grover's algorithm could seriously jeopardize symmetric key algorithms by significantly speeding up exhaustive key search attacks or brute force attacks, effectively halving the cryptographic key length. For an n-bit symmetric cryptographic algorithm, there are 2^n possible keys. With the current computing platforms, the key range of 2^128 for a 128-bit AES algorithm is virtually uncrackable. However, with the advent of quantum platforms and the implementation of Grover's algorithm, the security level of AES's 128-bit key size would effectively be reduced to an insecure 64-bit equivalent key length. Fortunately, most symmetric key algorithms provide additional key lengths, enabling applications to shift to the more secure versions. Hash algorithms are also likely to be impacted by Grover's algorithm, given their nature of producing a fixed-sized output from inputs of arbitrary size. The enhanced pace of Grover's algorithm could potentially speed up collision attacks, which involve identifying two distinct inputs that yield the same output. The emergence of quantum-based platforms also poses a challenge for hash algorithms. However, hash functions such as SHA-256, which yields a 256-bit output, and SHA-512, with a 512-bit output, appear to remain resistant to quantum attacks due to their longer output lengths.

However, unlike Shor's algorithm, Grover's algorithm doesn't pose an existential threat to all symmetric cryptographic systems. The serial processing requirement of Grover's algorithm limits its impact on the symmetric cryptography. Additionally, the threat can be mitigated by simply doubling the key length, a change that would impose minimal overhead on modern computational systems but exponentially increases the time required by Grover's algorithm to find a match.

In conclusion, the advent of Grover's algorithm indeed underlines the need for transitioning to post quantum cryptographic systems that can withstand quantum adversaries,

but it has less devastating effect comparing to Shor's algorithm.

2.3 Related work

WireGuard's foundational construct is the authenticated key exchange, a critical part that determines the overall security of the protocol. Authenticated key exchange is a cryptographic method that allows two entities to securely derive a shared secret over an insecure channel. An famous type of AKE is the (Elliptic Curve) Diffie-Hellman key exchange, which is favored for its flexibility and ability to offer security attributes like forward secrecy. Currently, many practical AKE protocols are built on DH implementations, including WireGuard. As we will move into the post-quantum era soon, relying on the security of the Diffie-Hellman assumption becomes unreasonable. Thus, it is crucial to transition towards post-quantum cryptographic (PQC) alternatives that are believed to resist quantum computer attacks. The research community has put great effort into this area. NIST's Post-Quantum Cryptography Standardization ahs split the public-key cryptosystems into public key encryption and key establishment algorithms since round 2 Submissions. The result of this process is **Selected Algorithms 2022** [3]: **CRYSTALS-KYBER** for public key encryption,**CRYSTALS-DILITHIUM** for digital signature algorithm plus **Round 4 Submission** [r4]. Considering the mathematical categories of the remained algorithms, they all fall within three mathematical assumptions: Hash-based solutions use a hash function to take variable size input into an diffused, variable size output. Hash functions are also called one-way functions as they are not easily (i.e. computationally complex) computed in reverse. This is what makes them very useful in authenticating signatures. Lattice-based solutions are based on a set of defined basis vectors (think distance and direction) that generate a whole set. The security of lattice-based solutions relies on the difficulty of computing the Shortest Vector Problem (SVP). A small amount of noise is added into the computations to make it extremely difficult to recover the secret even if one was given the random matrix and resultant vector. This is called learning with errors (LWE). Variants of this problem included Ring-LWE, Module-LWE, and LWR or Learning with Rounding.

The two security aspects of AKE are confidentiality and authentication. PQC digital signature schemes and there exists three forms of authenticated key exchanges based on their construction and the latest results of NIST PQC process:

2.3.1 Transitional post-quantum constructions

A popular approach to transitional post-quantum security is to combine ECDH with a post-quantum primitive, forming a hybrid quantum-classical cryptography. This approach provides the benefits of the well-studied classical security while also providing a layer of post-quantum security via the use of a post-quantum algorithm. In [BBF2018], Bindel, Brendel, Brendel, Goncalves, Stebila established the models for hybrid authenticated key exchange protocols. Also, Google and [Cloudflare] conducted the [CECPQ2] experiment that intended to help evaluate the combination of [X25519] and a post-quantum key-agreement based on [NTRU-HRSS-KEM] a plugin for the TLS key-agreement part. This experiment implemented two hybrid quantum-classical key exchanges: [CECPQ2](lattice-based NTRU-HRSS + X25519)and [CECPQ2b] (isogeny-based SIKE + X25519), integrated them into their TLS stack and deployed on their servers and in Chrome Canary clients. The post-quantum components are KEM-based algorithms and the final shared secrets are concatenate and combine with HKDF(HMAC-based Extract-and-Expand Key Derivation Function). Crockett, Paquin, Stebila [CPS2019] prototyped and inspected different design of combining classical and post-quantum algorithms.

2.3.2 KEM-based authenticated key exchange constructions

Typically, The underlining algebraic structures of some PQC can provide a Public Key Encryption (PKE), which can then be converted into a Key Encapsulation Mechanism (KEM) through a standard transformation construction. However, a particular characteristic of these algebraic structures used in many post-quantum PKEs and KEMs lead to malleable ciphertexts, expose them to chosen-ciphertext attacks (CCA) [HNP+03]. To counter the effect, the most common way is to apply the Fujisaki-Okamoto (FO) transform or its variants and upgrade OW-CPA security to IND-CCA2 security. In 2018,[HKS+2018], Hövelmanns, Kiltz, Schäge and Unruh proposed a modification of Fujisaki-Okamoto AKE that can turn any deterministic PKE into FO-like IND-CCA secure KEMs in the quantum random oracle model. And Bindel, Hamburg, Hövelmanns, Hülsing, Persichetti tighten the bound and extended their result to the case of non-deterministic PKEs or PKEs feature decryption failure. There are excellent examples show that KEM is a drop-in replacement alternative to build the ability of key agreement without the traditional signature schemes and Diffie-Hellman key agreement schemes. In [SSW2022], Schwabe, Stebila, Wiggers introduce KEMTLS, a novel approach to implement handshake protocol of the TLS 1.3 that make full of IND-CCA-secure KEMs as a new way of sever/client authentication instead of signatures. In 2021, Hülsing, Ning, Schwabe present PQ-WireGuard that filled the emptiness in the area of post-quantum

authentication.

2.4 Summary

Summarize the chapter and present a comparison of the projects that you reviewed.

	Aspect #1	**Aspect #2**
Row 1	Item 1	Item 2
Row 2	Item 1	Item 2
Row 3	Item 1	Item 2
Row 4	Item 1	Item 2

Table 2.1: Caption that explains the table to the reader

Chapter 3

Design

As outlined in Sections I and II, the WireGuard handshake is heavily based on DH, which does not have an efficient and well established post-quantum equivalent. Hence, in this section we describe how we replace DH by KEMs, for which well-established, efficient post-quantum instantiations exist. We start by considering a simplified view on the core of the DH-based WireGuard handshake.

3.1 Secuirt Aspect

The primary goal of the adaptations proposed in this thesis is to provide security against post quantum attackers, who already are probably recording traffic at the current time for later decryption. Therefore, the necessary security properties that need to be guaranteed are listed below

3.1.1 Secrecy

Three key encapsulations using the keypairs `sski`/`spki`, `sskr`/`spkr`, and `eski`/`epki` provide secrecy. Their respective ciphertexts are called `scti`, `sctr`, and `ectr` and the resulting keys are called `spti`, `sptr`, `epti`. A single secure encapsulation is sufficient to provide secrecy. We use two different KEMs: Kyber and Classic McEliece.

3.1.2 Authenticity

The key encapsulation using the keypair `sskr`/`spkr` authenticates the responder from the perspective of the initiator. The KEM encapsulation `sski`/`spki` authenticates the initiator from the perspective of the responder. Authenticity is based on the security of Classic McEliece alone.

3.1.3 Pre-Shared Symmetric Key

We allow the use of a pre-shared key (`psk`) as protocol input. Even if all asymmetric security primitives turn out to be insecure, providing a secure `psk` will have KEM-WireGuard authenticate both peers, and output a secure shared key.

3.1.4 Forward Secrecy

Forward secrecy refers to secrecy of past sessions in case all static keys are leaked. Imagine an attacker recording the network messages sent between two devices, developing an interest in some particular exchange, and stealing both computers in an attempt to decrypt that conversation. By stealing the hardware, the attacker gains access to `sski`, `sskr`, and the symmetric secret `psk`. Since the ephemeral keypair `eski`/`epki` is generated on the fly and deleted after the execution of the protocol, it cannot be recovered by stealing the devices, and thus, KEM-WireGuard provides forward secrecy. Forward secrecy relies on the security of Kyber and on proper zeroization, i.e., the implementation must erase all temporary variables.

3.1.5 Security against State Disruption Attacks

Both WireGuard and PQ-WireGuard are vulnerable to state disruption attacks; they rely on a timestamp to protect against replay of the first protocol message. An attacker who can tamper with the local time of the protocol initiator can inhibit future handshakes, rendering the initiator's static keypair practically useless. Due to the use of the insecure NTP protocol, real-world deployments are vulnerable to this attack. Lacking a reliable way to detect retransmission, we remove the replay protection mechanism and store the responder state in an encrypted cookie called "the biscuit" instead. Since the responder does not store any session-dependent state until the initiator is interactively authenticated, there is no state to disrupt in an attack.

Note that while KEM-WireGuard is secure against state disruption, using it does not protect WireGuard against the attack. Therefore, the hybrid KEM-WireGuard/WireGuard setup recommended for deployment is still vulnerable.

3.1.6 Cryptographic Building Blocks

All symmetric keys and hash values used in KEM-WireGuard are 32 bytes long.

Hash

A keyed hash function with one 32-byte input, one variable-size input, and one 32-byte output. As keyed hash function we use the HMAC construction [@rfc_hmac] with BLAKE2s [@rfc_blake2] as the inner hash function.

```
hash(key, data) -> key
```

XAEAD

Authenticated encryption with additional data for use with random nonces. We use XChaCha20Poly1305 [@draft_xchachapoly] in the implementation, a construction also used by WireGuard.

```
hash(key, data) -> key XAEAD::enc(key, nonce, plaintext, additional_data) -> ciphertext XAEAD::dec(key, nonce, ciphertext, additional_data) -> plaintext
```

SKEM

'Key Encapsulation Mechanism'(KEM) is the name of an interface widely used in post-quantum-secure protocols. KEMs can be seen as asymmetric encryption specifically for symmetric keys. KEM-WireGuard uses two different KEMs. SKEM is the key encapsulation mechanism used with the static keypairs in KEM-WireGuard. The public keys of these keypairs are not transmitted over the wire during the protocol. We use Classic McEliece 460896 [@mceliece] which claims to be as hard to break as 192-bit AES. As one of the oldest post-quantum-secure KEMs, it enjoys wide trust among cryptographers, but it has not been chosen for standardization by NIST. Its ciphertexts and private keys are small (188 bytes and 13568 bytes), and its public keys are large (524160 bytes). This fits our use case: public keys are exchanged out-of-band, and only the small ciphertexts have to be transmitted during the handshake.

```
SKEM::enc(public_key) -> (ciphertext, shared_key) SKEM::dec(secret_key, ciphertext) -> shared_key
```

EKEM

Key encapsulation mechanism used with the ephemeral KEM keypairs in KEM-WireGuard. The public keys of these keypairs need to be transmitted over the wire during the protocol.

We use Kyber-512 [@kyber], which has been selected in the NIST post-quantum cryptography competition and claims to be as hard to break as 128-bit AES. Its ciphertexts, public keys, and private keys are 768, 800, and 1632 bytes long, respectively, providing a good balance for our use case as both a public key and a ciphertext have to be transmitted during the handshake.

```
EKEM::enc(public_key) -> (ciphertext, shared_key) EKEM::dec(secret_key, ciphertext) -> shared_key
```

Using a combination of two KEMs – Classic McEliece for static keys and Kyber for ephemeral keys – results in large static public keys, but allows us to fit all network messages into a single IPv6 frame.

KEM-WireGuard uses libsodium [@libsodium] as cryptographic backend for hash, AEAD, and XAEAD, and liboqs [@liboqs] for the post-quantum-secure KEMs.

3.2 Overview of the Design

A description of the approach that addresses the problem identified above.

3.3 Summary

Every chapter aside from the first and last chapter should conclude with a summary.

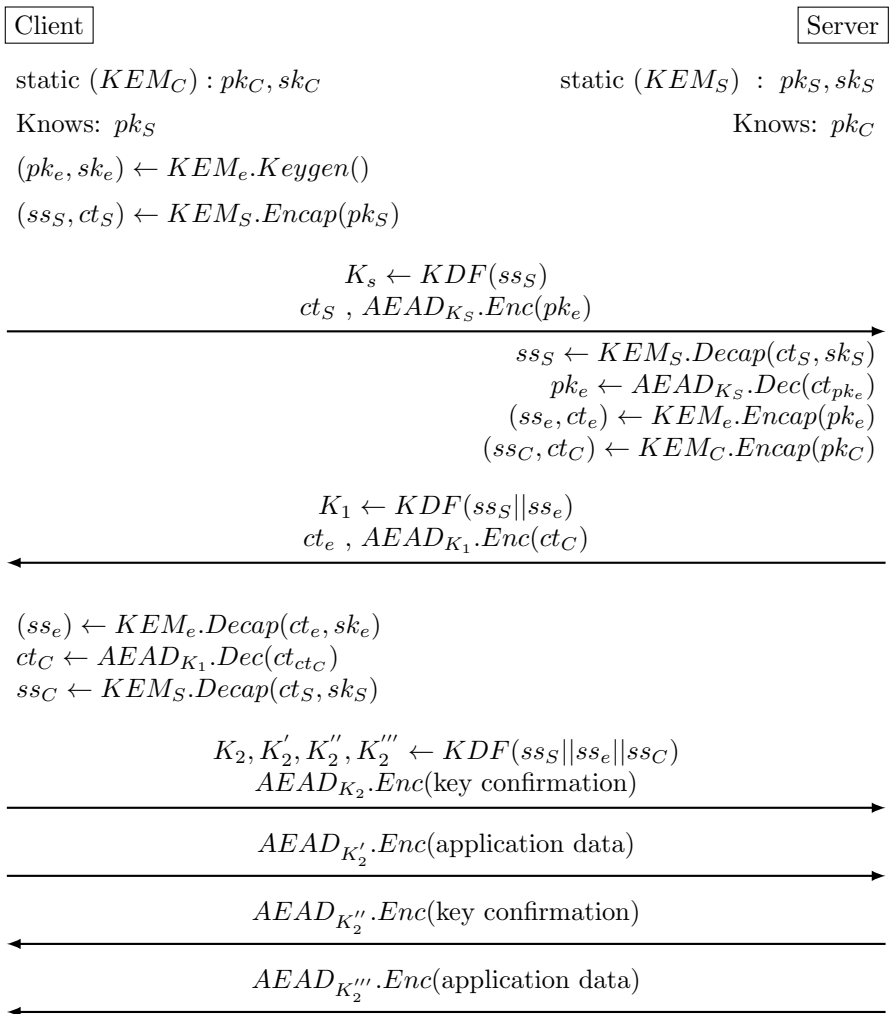


Figure 3.1: Generic KEM-WireGuard deisgn

Chapter 4

Implementation

Guess what? At the beginning of each chapter, a description should introduce the reader to the content of the chapter. The description should explain to the reader the layout of the chapter, the contribution that the chapter makes to the overall dissertation and the contribution of the individual sections towards the overall chapter.

4.1 Overview of the Solution

```
x = 1
if x == 1:
    # indented four spaces
    print("x is 1.")
```

Listing 4.1: Lengthy caption explaining the code to the reader

The code in listing 4.1 is a demonstration how to include a file with code into the template.

4.2 Component One

The code in listing 4.2 is a demonstration how to include code in the template.

```
x = 1
if x == 1:
    # indented four spaces
    print("x is 1.")
```

Listing 4.2: Second Lengthy caption

4.3 Summary

Every chapter aside from the first and last chapter should conclude with a summary.

Chapter 5

Evaluation

As the main question here is one of usability, the following will mainly be a performance comparison, comparing the proof-of-concept handshake protocol implementations to the default WireGuard protocol, as implemented in WireGuard-Go. To accommodate for any of the three higher security levels defined in the Feature Specification, only the handshake code was changed. Therefore, most focus will be put on the handshake computations, as well as initiation and response message transmissions.

5.0.1 Message Sizes

Adding more key exchange primitives to the handshake protocol obviously adds additional data, in the form of public keys and encapsulated values. This data needs to be transmitted in the initiation or response message. With PQ KEMs that data is relatively large for network transmissions, hundreds of Bytes up to hundreds of Kilobytes in the worst cases. Thus, packets may easily become larger than the Ethernet maximum transmission unit (MTU) of 1500 Bytes. Since WireGuard is based on UDP, there is no segmentation functionality on the Transport Layer. If large data is sent over UDP, packet fragmentation may happen on the IP layer. Relying on this functionality is considered fragile, and the IETF specifically advises against it. [8] This is because the IP standard allows routers to drop large packets silently. A way around IP fragmentation is to split the datagrams on the application layer. When using UDP, this can still be terrible for performance because the handshake will fail anytime one of the datagrams is lost. The more datagrams we need for one message, the higher the probability for losing one of them is. Thus, breaking up messages into more datagrams makes the handshake take longer on average. A datagram size of 1436 Bytes is reasonable for this, because according to analysis by Shannon and Moore [31] around 98Subtracting 8 Bytes for the UDP header and 40 Bytes for the possibility of an IPv6 header, we arrive at the number above. Also,

the IPv6 standard recommends an MTU of at least 1500 Bytes, and requires acceptance of packets up to that same size. Choosing cryptographic primitives with small key sizes is thus essential to prevent unnecessary overhead, in the form of excessive transmission times. Cryptographic primitives with excessively large public keys or encapsulated values have thus been excluded from all following considerations. For example, all code-based cryptosystems, which need many Kilobytes for a public key.

Primitive	PublicKeySize	PrivateKeySize	CipherTextSize
X25519	32	32	32
Kyber512	800	1632	768
NTRU LPRime	897	164	993
Streamlined NTRU	994	1518	897
BIKE-L1	1541	5223	1573
HQC-128	2249	2289	4481
McEliece348864	261120	6492	96

Table 5.1: Bytes needed for the NIST-PQC level I primitives

5.0.2 Individual runtime

Runtime	Key Generation	Encapsulation	Decapsulation
mceliece348864f	51.88	0.35	19.70
mceliece460896f	166.49	0.66	43.96
kyber512	0.02	0.01	0.01
kyber768	0.03	0.02	0.02
ntrulpr653	5.74	11.45	17.09
sntrup653	35.66	5.72	17.16
BIKE-L1	0.13	0.03	0.5
HQC-128	0.03	0.05	0.09

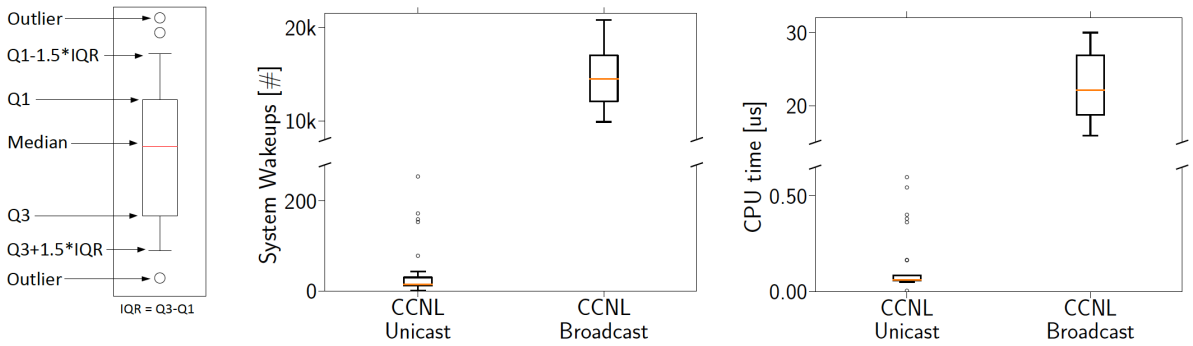
Table 5.2: Runtime for the NIST-PQC level I primitives (in millisecond)

5.1 Experiments

In the case where experiments have been carried out, the experimental setup and the values that were defined for the variables need to be presented in a table e.g. table **??**.

5.2 Results

Figures that present results such as figure 5.1 need to display descriptions of the axes, the units and scales of the measurements, statistical values, etc. Where measurements were taken from experiments, error bars or confidence intervals need to be provided to give the reader an indication of the spread of the measurements.



(b) System wakeups (c) CPU usage

Figure 5.1: Long caption that describes the figure to the reader

5.3 Summary

Every chapter aside from the first and last chapter should conclude with a summary that presents the outcome of the chapter in a short, accessible form.

Chapter 6

Conclusions & Future Work

This chapter should summarize the work presented in the dissertation and discuss the conclusions that can be drawn from the work and the results presented in chapter 5.

6.1 Future Work

The section may present a list of items that were beyond the scope of the dissertation.

Bibliography