Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

Sander Strand Engen

Hybridising Classical and Post-Quantum Cryptography in WireGuard

Master's thesis in Communication Technology and Digital Security
Supervisor: Professor Colin Boyd
Co-supervisor: Assistant Professor Bor de Kock
January 2023

NTNU
Norwegian University of Science and Technology

Sander Strand Engen

Hybridising Classical and Post-Quantum Cryptography in WireGuard

Master's thesis in Communication Technology and Digital Security
Supervisor: Professor Colin Boyd
Co-supervisor: Assistant Professor Bor de Kock
January 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of
Science and Technology

Title: Hybridising Classical and Post-Quantum Cryptography in Wireguard

Student: Engen, Sander Strand

Problem description:

Whitfield Diffie and Martin Hellman's namesake algorithm Diffie-Hellman (DH) and its variations have been the standard for Public Key Cryptography following its publication in 1976 [DH76]. The algorithm's security properties are based on the assumption that the underlying Discrete Logarithm problem is computationally intractable on classical computers. This however is not the case for quantum computers. In 1994, Peter Shor published a method for solving the Discrete Logarithm (DL) problem on a quantum computer [Sho94], hence known as Shor's Algorithm. DH and its variations have seen continued use even though it has been considered broken for decades, on the assumption that the existence of quantum computers with the necessary strength to efficiently apply Shor's Algorithm is a far-away reality. In recent years, this assumption no longer holds water. Developments in quantum computing has lead researchers to assume that this post-quantum paradigm is probable to be reached within the next 30 years [SR20].

However, cryptographers have not been idle in the decades since Shor's Algorithm was discovered. The field of Post-Quantum Cryptography aims to develop public key-exchange (PKE) algorithms based on mathematical primitives different from the DL problem, that still remain secure in a post-quantum world. A multitude of algorithms have been proposed to supplant DH as the standard. A lot of these algorithms have not withstood the years of scrutiny that DH has. Promising candidates have been broken, and more are probable to follow before we reach the post-quantum paradigm. With an increasing need to put post-quantum algorithms into use early combined with the uncertainty of the security properties of these algorithms, researchers have proposed the solution of using post-quantum algorithms in tandem with classical algorithms like DH, in a hybrid setting [GCR22]. This entails that the resulting hybrid algorithm remains secure as long as one of its two components are considered secure.

This study will implement this type of hybrid construction in the VPN protocol WireGuard. The purpose of this implementation is to analyse the computational and performance costs of transitioning to a post-quantum paradigm through the use of hybrid schemes. The implementation aims to test different post-quantum primitives in combination with WireGuard's existing Elliptic Curve Diffie-Hellman (ECDH) key exchange. The goal of the study is to present a working construction of a hybrid scheme that remains secure as long as one of the components is considered secure.

Further, it aims to gain insight into how different post-quantum candidates lend themselves to being combined in terms of computational performance on key exchange initiator and responder machines, and within the restrictions of the transport medium between them. Ultimately, it aims to lay the groundwork for a discussion on the feasibility of putting hybrid schemes into widespread use today.

Approved on:	2022-10-17
Main supervisor:	Professor Boyd, Colin, NTNU
Co-supervisor:	Assistant Professor de Kock, Bor, NTNU

Abstract

Due to the uncertainty of the security properties of post-quantum asymmetric cryptosystems, early adoption of post-quantum cryptography should be done through the use of hybrid systems, where the post-quantum cryptosystem is implemented in tandem with a classical cryptosystem, and the system maintains its security properties as long as at least one of its two components are considered secure.

This thesis describes a method for implementing such a hybrid cryptosystem through the use of an XOR-then-MAC combiner, and outlines the proccess of implementing this hybrid cryptosystem in the userspace Rust implementation of the VPN protocol WireGuard, called BoringTun. With this implementation, the thesis presents performance metrics derived from using different post-quantum components along with a key encapsulation mechanism constructed using ECDH.

The results show that this hybrid implementation increases the time to complete a handshake by a factor of three to four times that of a handshake performed with ECDH. The thesis concludes that lattice-based cryptography is best suited for this application, with the NTRU cryptosystem outperforming the other candidates with respect to the NIST standardisation process' security category V. Further it shows that putting a hybrid cryptosystem into use is feasible today, without impacting the end user experience significantly.

Sammendrag

På grunn av usikkerheten rundt sikkerhetsegenskapene til post-kvante kryptosystemer bør tidlig adopsjon av post-kvante kryptografi bli utført ved å bruke hybride systemer, hvor et post-kvante kryptosystem blir implementert i kombinasjon med et klassisk kryptosystem, og hvor systemet opprettholder sine sikkerhetsegenskaper så lenge minst en av de to komponentene er ansett som sikker.

Denne oppgaven beskriver en metode for å implementere denne typen hybride kryptosystem ved å bruke en XOR-så-MAC kombinator, og beskriver arbeid utført for å implementere dette hybride kryptosystemet i brukerområde-implementasjonen av VPN-protokollen WireGuard i Rust, BoringTun. Gjennom testing av denne implementasjonen beskriver oppgaven ytelsesberegninger for bruk av forskjellige post-kvante komponenter i kombinasjon med en nøkkelenkapsuleringsmekanisme konstruert ved å bruke ECDH.

Resultatene viser at denne hybride implementasjonen øker tiden for å gjennomføre et handshake med en faktor på tre til fire ganger tiden for et handshake utført med kun ECDH. Oppgaven konkluderer med at gitterbasert kryptografi egner seg best for applikasjonen, og at kryptosystemet NTRU gir bedre resultatet enn andre kandidater med tanke på NIST's standardiseringsprosess sin sikkerhetskategori V. Videre viser den at det å sette i bruk et hybrid kryptosystem i dag er gjennomførbart, uten å ha store konsekvenser for brukeropplevelsen.

Preface

This thesis marks the end of my studies at the Master's program in Communication Technology and Digital Security (MTKOM) at NTNU. I would like to thank my supervisor Bor de Kock and Prof. Colin Boyd for all the exemplary guidance they provided throughout the pre-project and when writing the thesis.

I would also like to thank all the friends I made working with the student paper, Under Dusken, for four years, that made my time studying in Trondheim enjoyable. Finally, I would like to thank my family for all the support they have provided throughout my studies.

Contents

List of Acronyms

CPU Central Processing Unit.

DFT Discrete Fourier Transform.

DH Diffie-Hellman.

DL Discrete Logarithm.

ECDH Elliptic Curve Diffie-Hellman.

epk Ephemeral Public Key.

esk Ephemeral Private Key.

GCD Greatest Common Divisor.

GNFS General Number Field Sieve.

IF Integer Factorisation.

IND-CCA Indistinguishability against a Chosen Ciphertext Attack.

IND-CPA Indistinguishability against a Chosen Plaintext Attack.

IP Internet Protocol.

KDF Key Derivation Function.

KEM Key Encapsulation Mechanism.

MAC Message Authentication Code.

MTU Maximum Transmission Unit.

NIST United States National Institute of Standards and Technology.

NTNU Norwegian University of Science and Technology.

OQS Open Quantum Safe.

OS Operating System.

PKE Public Key Exchange/Encryption.

PQ Post-quantum.

psk Pre-shared Key.

QFT Quantum Fourier Transform.

RAM Random Access Memory.

RSA Rivest–Shamir–Adleman cryptosystem.

SD Syndrome Decoding.

shk Shared key: Describes an encapsulated KEM key.

SIDH Super-singular Isogeny Diffie-Hellman.

spk Static Public Key.

ssk Static Private Key.

SVP Shortest Vector Problem.

TLS Transport Layer Security.

UDP User Datagram Protocol.

VPN Virtual Private Network.

XOR Exclusive or/disjunction.

Chapter 1
Introduction

This chapter will introduce some of the key concepts related to post-quantum cryptography and the project as a whole. With these concepts in mind, it aims to provide motivation for researching the project detailed in this thesis. Finally, this chapter will introduce the research questions that will be discussed.

1.1 Public Key Cryptography

Before the digital age that we live in today, the field of cryptography was more or less limited to the study of what we now call symmetric cryptography, where two or more communicating parties have access to the same secret key, which is used for both encryption of plaintext and decryption of ciphertext. With the emergence of computer-based communications, two main problems arose with this type of cryptographic system.

The first problem with this type of system was that for a number of parties n, each party needs to keep track of $n(n-1)$ symmetric keys. With an increasing number of parties, the amount of keys quickly becomes unmanageable due to the polynomial rate of growth. The second problem is that a symmetric key needs to be shared securely before the parties can exchange encrypted messages. Without a perfectly secure channel, ironically being what the system is intended to create, sharing this secret is impossible.

The solution to these two, along with a myriad of other related problems, was published in 1976 by Diffie and Hellman [DH76]. Their namesake algorithm Diffie-Hellman (DH) created an asymmetric cryptosystem based on the computational hardness assumption of the Discrete Logarithm (DL) problem. The system allowed for a party to keep track of a single key pair; a public key for encryption of messages intended for themselves, and a private key to decrypt those messages. A party could now request the public key of the recipient, encrypt their message and transmit it

across a public channel, with the recipient being the only party able to decrypt it. We call this scheme a Public Key Exchange (PKE).

1.2 Post-Quantum Cryptography

Since its invention, DH and its variation Elliptic Curve DH (ECDH) have been one of, if not the most important building blocks of secure communication across the internet. However, assumptions have a tendency to be proven false. The computational hardness assumption for DL was broken in 1994 when Peter Shor discovered an algorithm that solved the Integer Factorisation (IF) problem [Sho94], and subsequently the DL problem in polynomial time. The catch that has kept DH in use all this time is that Shor's Algorithm requires a quantum computer to realise its efficiency.

Cryptographic schemes based on the IF and DL problems have continued to see widespread use throughout the years since Shor's Algorithm's discovery, on the assumption that a quantum computer with sufficient strength to realise the algorithm is a far-away reality. Recent developments in quantum computing is starting to indicate that this assumption no longer is valid. Researchers believe that this point will be reached within the next 30 years [SR20], making way for the field of Post-Quantum (PQ) Cryptography, which aims to replace current cryptographic schemes with new ones based on different mathematical primitives than the IF or DL problem.

There is an increasing need to put quantum-resistant cryptosystems into use sooner than later. You don't need to don a tinfoil hat to assume that several private and nation state actors are collecting massive amounts of encrypted data that might very well be forcefully decrypted in the near future, and information that is sensitive today will still remain sensitive in 30 years. A prime example of an actor capable and willing of doing this is the United States National Security Agency (NSA) with their surveillance program PRISM, leaked in 2013 by Edward Snowden [GP13].

To facilitate a transition to PQ schemes, the US National Institute of Standards and Technology (NIST) initiated a standardisation process for quantum-resistant algorithms in 2016 [Nat16], with a focus on both PKE and Signature schemes. Currently, the process is in its fourth round of evaluation, with submitted algorithms mainly falling into two categories, Code-based or Lattice-based. Code-based schemes rely on the computational hardness assumption of the Syndrome Decoding (SD) problem [OS09], while Lattice-based schemes rely on the Shortest Vector Problem (SVP) [MR09].

1.3 Hybrid Key Exchange

The standardisation process has been going on for several years. Even though some selections have been made, how long we have to wait before a PQ ciphersuite is standarised and put into use is still unclear. The pressing need to transition to PQ schemes has however not gotten any less urgent.

The problem is that even though these new PQ schemes seem promising, they have not withstood the years of scrutiny that DH has, and there is an uncertainty whether their security assumptions are correct. To mitigate this problem while still implementing PQ schemes, some researchers have suggested using a hybrid scheme during this transitional period [GCR22].

A hybrid scheme entails combining a post-quantum scheme with a scheme that is susceptible to Shor's algorithm in a way that preserves the security properties of the combined system as long as one of the component algorithms remains secure. By combining a variation of DH with a PQ scheme in this way, one can at the very least ensure that the scheme is secure in a pre-quantum paradigm, and hopefully secure in a PQ paradigm assuming the PQ scheme stays unbroken.

In fact, a promising PQ candidate was just recently proven susceptible to an attack performed on a classical computer. Supersingular Isogeny-based DH (SIDH) [CCH+19] was assumed quantum-secure and stood out amongst other standardisation candidates with a small key size and the possibility to apply a PKE transmission pattern that was analogous to DH, as opposed to the Key Encapsulation Mechanism (KEM) applied by the other candidates. In august 2022, Castryck and Decru published a highly efficient classical attack on SIDH which rendered the entire scheme insecure [CD22], based on a 1997 theorem by Kani [Kan97].

The case of SIDH proves that this type of hybrid approach is not only prudent, but necessary while PQ schemes are being scrutinised. There is a clear possibility, if not a guarantee, that other assumed quantum-resistant schemes will be broken either by a classical or theoretic quantum attack throughout this process.

1.4 Research Questions

This thesis will be presenting a version of the VPN protocol WireGuard with a hybrid scheme as its handshake component, for the purpose of testing how such a scheme would work in a real world setting, and what effects it would have on the application as compared to the original, classically secure version. With this implementation we will be testing performance metrics of the hybrid implementation and discuss the results with respect to the following research questions:

- How does a hybrid setting impact WireGuard's performance?
- Which post-quantum primitive provides the best performance metrics in a hybrid setting in WireGuard while adhering to WireGuard's original performance goals?
- Is the computational trade-off from implementing post-quantum cryptography through a hybrid mechanism proportional to the security benefits gained, and thereby making an early transition feasible?

Chapter 2
Background

This chapter aims to lay the groundwork for understanding the interworkings of the research project. It describes research and theory related to the field of post-quantum cryptography, and details component mechanisms to construct a hybrid cryptosystem consisting of a classical and a post-quantum component cryptosystem.

2.1 Post-Quantum Cryptography

Post-quantum cryptography is a field of research that aims to develop cryptographic algorithms that resists attacks from quantum computers. The following subsections will detail quantum computing as a concept and the current state of technological developments, the consequences for modern cryptography and cryptographic schemes utilising computational problems that avoid them.

2.1.1 Quantum Computing

The field of quantum computing has been steadily growing since the proposition of a Turing machine that could operate under the laws of quantum mechanics by Benioff in 1980 [Ben80]. The prevailing method of realising such a machine is through the use of quantum logic gates that together can create a quantum circuit [SW95], analogous to how classical computers are built on transistor logic gates that together create a digital circuit.

The main difference between a quantum computer and a classical computer is related to how they deal with memory and units of information. In classical computing, information is stored and manipulated as binary units, represented with [1]s and [0]s. Through the use of the quantum mechanical property of superposition, quantum computers replace the bit with a qubit, which exists in a superposition of the classical bits [1] and [0]. In simple terms, this means that the information exists with a certain probability of being either in state [1] or [0], and collapses into existing as one of the states according to that probability when measured.

In essence, this allows a quantum computer to store multiple values in a single qubit. By preparing a number of qubits in an exponential number of specific superpositions, quantum computers can then perform a computation on those inputs instantly. These computations allow for quantum algorithms to be implemented. A quantum algorithm is an algorithm that utilises a quantum mechanical property such as superposition or entanglement to realise a considerable algorithmic complexity speedup when performed on a quantum computer as opposed to a classical computer.

One example of such an algorithm is the Quantum Fourier Transform (QFT), which is a quantum equivalent of the Discrete Fourier Transform (DFT) [Cop94]. The QFT uses a quantum register, a series of qubits in prepared superpositions, as input and performs a Fourier transformation as its output. While the DFT requires $O(n2^n)$ gates, the QFT uses the property of the input being 2^n to require just $O(n^2)$ gates, which equates to an exponential increase in efficiency. Following the discovery of the QFT by Coppersmith in 1994, several quantum algorithms have been developed making use of its efficiency.

Bringing quantum computers out of the theoretical world and into the real world is however not an easy task. To create a quantum computer one would need to be able to isolate and manipulate particles small enough to be susceptible to quantum mechanics such as electrons or ions, with high precision. Current implementations such as IBM's Quantum System One requires very rare superconducting materials such as niobium [IBM22b] to achieve this ability.

One of the largest challenges with manipulating quantum particles is quantum decoherence, which results in quantum computers being increasingly error-prone the longer an operation takes to complete [CLSZ95]. These particles are inherently unstable unless perfectly isolated from outside interactions such as radiation. In the real world perfect isolation is impossible as components of the quantum processor itself can be the source of these interactions.

A way to mitigate this issue is to use several qubits for error correction. When talking about qubits, it's common to differentiate between physical and logical qubits. A physical qubit is the physical device emulating a two-state quantum system, for example an ion trapped in an electromagnetic field, along with mechanisms to manipulate its quantum mechanical properties and to measure them. A logical qubit is an abstract construction of multiple physical qubits that is able to perform as specified when doing a quantum computation. To construct a single logical qubit, one would need at least 1000 physical qubits according to current estimates to ensure stability and error correction [FMMC12].

Despite these challenges, progress is being made in the construction of quantum computers. In the case of IBM's Quantum System, their 433-qubit processor, IBM

Osprey, was unveiled in November 2022 [CN22], a year after the unveiling of the 127-qubit IBM Eagle. In addition to this, IBM has indicated that their research will lead to the announcement of a 1121-qubit processor in 2023 through their development roadmap [IBM22a], which could possibly realise the first implementation of a generalised logical qubit.

If the yearly increase in qubits in IBM's quantum processors remains a continuing trend, there seems to be an exponential growth per year, reminiscent of Moore's Law which implies that computational power for classical processors doubles every two years. If there is in fact a quantum equivalent of Moore's Law, the quantum supremacy where quantum algorithms are able to outperform classical algorithms for applicable problems with realistic parameters may very well arrive sooner than expected.

2.1.2 Shor's Algorithm

The same year as Coppersmith's discovery of the QFT, mathematician Peter Shor applied it to make another discovery that would have serious consequences for asymmetric cryptography. Combining the efficiency of the QFT with modular exponentiation through repeated squarings, Shor developed an algorithm capable of finding the prime factors of an integer with algorithmic complexity of approximately order $O((log\ n)^3))$ when performed on a quantum computer [Sho94]. Compared to the General Number Field Sieve (GNFS), the fastest known classical integer factorisation algorithm which runs in sub-exponential complexity, Shor's algorithm provides an exponential speedup.

Shor's algorithm works in simple terms by first reducing the problem of factoring to order-finding [Wol87]. It then uses the QFT through initialising the input as a quantum superposition of pairs of integers of the form $(r, x^r\ mod\ N)$, where N is the integer the algorithm is trying to factor, x being a random number not divisible by N's factors, and r being possible periods. When collapsing the QFT, interference cancels out all periods other than the true period r. One can then find a factor of N by computing the greatest common divisor (GCD) of $r - 1$ and N.

By reducing the algorithmic complexity from sub-exponential to polynomial, the problem of factorisation goes from being computationally intractable to trivial. In essence, a factorisation that would take thousands of years to solve on a classical computer could be performed on a quantum computer in a couple of minutes. As a consequence, Shor's algorithm practically breaks several cryptosystems.

Public-key cryptography is essential to how confidentiality and authenticity is maintained in the major part of today's digital communication. PKE algorithms such as DH and RSA are widely used in several internet protocols. Every time you

open a webpage, an HTTPS connection is made, invoking one of the aforementioned algorithms through the use of Transport Layer Security (TLS) [Res18]. DH and RSA are built upon the assumption that respectively the DL and IF problem are computationally intractable.

Similar to how the order-finding problem is a reduction of the IF problem, the IF problem is a reduction of the DL problem [Wol87]. As a result, Shor's algorithm renders the security assumptions of both DH and RSA broken when subjected to an attack from a sufficiently strong quantum computer. Despite the progress in quantum computing, a quantum computer strong enough to break for example RSA with 2048 bits of security in less than 24 hours would require at least 4098 logical qubits [GM19]. This would presumably equal more than 100 million physical qubits, which is several orders of magnitude more than the 433 physical qubits of the state-of-the-art IBM Osprey.

Since the discovery of Shor's algorithm, cryptographic algorithms based on susceptible computational problems have seen continued use for several decades, on the assumption that the existence of quantum computers of sufficient strength wouldn't be a reality in the foreseeable future. However, in recent years the developments in quantum computing have been accelerating. Extrapolating the progress in recent years, one can assume that a quantum computer capable of factoring a 2048 bit RSA key will be a reality within the next 30 years [SR20].

2.1.3 NIST Standardisation process

It's becoming clear that quantum computers with a large amount of processing power have gone from being theoretical, to a complicated engineering challenge, to an approaching inevitability. The rapid approach of the quantum supremacy with the realisation of Shor's algorithm's potential has been motivating the field of Post-quantum Cryptography (PQC). Creating a new public-key cryptography infrastructure to resist quantum computer attacks is not a simple task, and a deadline is starting to become apparent.

Since the discovery of Shor's algorithm, researchers have been developing algorithms based on different, quantum-resistant computational problems to DL and IF. With a lot of possible schemes being researched and developed, the US National Institute of Standards and Technology (NIST) initialised a process to establish a standard for post-quantum key exchange algorithms and signature schemes in 2016 [Nat16].

In the NIST process' call for proposal [Nat16], they define five categories for different security levels. Each category corresponds to computational resources required to break the security definitions for existing symmetric block ciphers or

NIST Level	Symmetric Equivalent	Computational cost
I	128-bit Block Cipher	2^{143}
II	256-bit Hash function	2^{146}
III	192-bit Block cipher	2^{207}
IV	384-bit Hash Function	2^{210}
V	256-bit Block cipher	2^{272}

Table 2.1: NIST security levels given in [Nat16].

hash functions. Table 2.1 shows the symmetric primitive with equivalent security definitions, along with the computational cost for an attack that breaks the level, given in the amount of classical logic gates required.

In the case of key exchange algorithms, the NIST process has throughout three rounds of evaluation whittled down the number from several dozen potential suitors to a handful. The relevant submitted schemes that have been submitted throughout the process can roughly be categorised into three categories, based on what problem they base their complexity assumption upon: Lattice-based, Code-based and Isogeny-based schemes.

2.1.4 Lattice-based Schemes

One of the most promising quantum-resistant computational problems that can be used for cryptography is the Shortest Vector Problem (SVP) [MR09]. In simple terms, the SVP asks to find the shortest non-zero vector in a vector space, given a lattice basis for the vector space and a norm, usually the euclidean norm.

Formally, the SVP is defined as following: Given a lattice basis $\mathbf{B}$, find a nonzero vector $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ such that $||\mathbf{v}|| = \lambda_1(\mathcal{L}(\mathbf{B}))$, where $\lambda_1(\mathcal{L})$ denotes the length of the shortest vector of the lattice $\mathcal{L}$. By additionally restricting the coefficients to be integers, this problem becomes computationally hard, with no known speedups using quantum computing.

The best known methods of solving or approximating the SVP are lattice enumeration and lattice sieving [Yas21]. Algorithms based on these methods are known to be of at least exponential complexity order, making them computationally intractable. Using problems that can presumably be reduced to the SVP, such as the Learning with Errors (LWE) or Learning with Rounding (LWR) problem, allows the creation of a quantum-resistant public key cryptosystem. This is due to their functionality as trapdoors, ie. easily computable one-way functions with computationally hard inverses [CK17].

Lattice-based schemes stand out among the other categories by having relatively short keys, allowing for more efficient transmission across the communication medium. In addition to efficient key sizes, some of these schemes are able to outperform private-key computations more efficiently than classical PKE algorithms such as RSA.

Several lattice-based schemes have been submitted to the NIST process. The lattice-based algorithm Kyber [BDK+18] was selected as the first standardised post-quantum key exchange algorithm in July 2022, following the third round of the process. With a lattice-based scheme selected for standardisation, trust in the security assumption for schemes based on the SVP remains strong.

The schemes NTRU [HRSS17] and Saber [DKSV18] were also considered during the third round of submissions, but were taken out of consideration for the fourth round of submissions as a result of Kyber being standarised.

2.1.5 Code-based Schemes

Code-based cryptography is probably the field within post-quantum cryptography that has existed for the longest time. Schemes within this category base their computational complexity assumption on the problem of finding the closest codeword to another, minimising the hamming distance, given a vector and a linear code. This is a problem within Coding Theory known as decoding a linear code over a noisy channel, and is known to be computationally hard [OS09], with the most efficient solution algorithms working in exponential complexity order.

The most well-known code-based scheme is the McEliece cryptosystem, developed in 1978 by Robert McEliece [McE78]. In essence, the scheme works by a receiving party providing a generator for a public linear code and a given number of allowed errors, and retaining a secret efficient decoding algorithm. Another party may then compute a product of the generator and a message, and randomly add errors according to the allowed number, resulting in a ciphertext. The receiving party may then decode the message using their secret decoding algorithm.

A modified McEliece cryptosystem from the original 1978 scheme [BCL+17] has been one of the candidates for the NIST process throug all the three completed rounds, along with other code-based schemes such as HQC [MAB+18]. A disadvantage with code-based schemes is that they often require larger public and private keys, making them highly inefficient when transmitting over a communication channel. Some schemes try to alleviate this problem using different methods to reduce these parameters, such as HQC with Quasi-Cyclic codes, but they still remain larger than their lattice-based competitors.

2.1.6 Isogeny-based Schemes

Though most of the submitted schemes of the NIST fall into either the lattice-based or code-based categories, there are some exceptions. Supersingular Isogeny Diffie-Hellman Key Exchange (SIDH/SIKE) [CCH+19] progressed through the three completed rounds of the process and showing promise in comparison to its competitors due to its ability to be used non-interactively, akin to the properties of the ECDH key exchange it aimed to be a quantum-resistant replacement of.

SIDH used secret isogenies between elliptic curves as its private keys as opposed to the elliptic curve scalars used in ECDH. Through this method, SIDH avoided being susceptible to being broken by Shor's algorithm by relying on isogeny walks instead of exponentiation. Though not susceptible to Shor's algorithm, weaknesses in SIDH's construction proved it not to be bullet-proof.

In August 2022, Castryck and Decru published an efficient attack on SIDH [CD22] exploiting information about auxiliary torsion points and properties of the starting curves shared publicly by the communicating parties, and thereby completely breaking the security of SIDH extremely efficiently using a classical computer. Though some isogeny-based schemes such as CSIDH might not be susceptible to the attack, trust in these types of schemes was severely reduced by the attack, and how it suddenly and completely broke SIDH. If new and better isogeny-based schemes might rise from the ashes of SIDH remains to be seen.

2.2 Key Encapsulation Mechanisms

Symmetric cryptosystems usually outperform asymmetric cryptosystems for encryption and decryption computations. Asymmetric schemes are mostly just used to transmit or otherwise agree upon a symmetric key for use in secure communication outside of the initial handshake. For this reason, all of the post-quantum schemes submitted to the NIST process are constructed as a Key Encapsulation Mechanism (KEM), instead of a key agreement protocol.

A KEM is a construction that uses asymmetric cryptography to encapsulate a symmetric key for a symmetric key encryption scheme (SKE), such as AES or ChaCha, and potentially additional information used for symmetric cryptography or padding. In most cases, the KEM derives information from the asymmetric scheme to generate the symmetric key.

KEMs are made up of three main components: A key-generation function KeyGen, an encapsulating function Encaps, and a decapsulating function Decaps. Required inputs and outputs for these functions are shown in algorithms 2.1, 2.2 and 2.3 respectively.

Algorithm 2.1 KEM.KeyGen()

1: **return** (pk, sk)

Algorithm 2.2 KEM.Encaps(pk)

1: **return** (c, shk)

Algorithm 2.3 KEM.Decaps(sk, c)

1: **return** shk'

The key-generation function generates two keys; a public key and a private key. The public key is then sent to a recipient who derives a symmetric key, usually from a shared secret generated through combining the recipient's private key with the initiator's public key, and encapsulates it using the encapsulating algorithm with the initiator's public key. The responder then sends the encapsulation to the initiator, who uses their private key and the responder's public key with the decapsulating algorithm to retrieve the symmetric key.

By using the KEM construction, a secure system can be easier abstracted without selecting a specific algorithm, and thereby provides flexibility for developers. In addition, KEM's provide efficiency for sharing symmetric keys, requiring only two messages to establish a symmetric key: The transmission of a public key, and in response transmitting the encapsulated symmetric key using a shared secret derived from the initiator's public key and the responder's private key.

2.3 Hybrid Scheme

DH and its derivatives have several years of research and scrutiny behind them, and apart from the quantum threat of Shor's algorithm, these schemes have maintained their security properties throughout. This is however not the case for the supposed quantum-resistant schemes. The field of post-quantum cryptography is not considered mature, with the attack on SIDH proving that promising candidates might very well be broken with further scrutiny.

Nevertheless, there is a need to put quantum-resistant schemes into use today. Information that is sensitive today is highly likely to be sensitive 30 years into the future. Personal information, economic data and medical information are some examples of data that is often transmitted with the assurance of the security properties of PKE schemes. An adversary might collect encrypted data containing these types of information today, with the intent of decrypting it using future methods and using that information with malicious intent.

With the uncertainty surrounding the security properties of supposed quantum-resistant algorithms along with the need to put these algorithms into use, researchers have suggested the use of post-quantum algorithms in a hybrid setting during the transitional period to quantum supremacy [GCR22]. This entails combining a post-quantum algorithm with a classical algorithm like ECDH in a way that ensures the security of the system as long as at least one of the component algorithms remains secure. That way, if the assumed post-quantum algorithms are broken through the use of a classical computer, akin to what happened to SIDH, the security properties of the classical component would at least preserve security against a classical adversary.

Defining how a system is secure is no easy task however, and when combining several schemes into one, security proofs for the component algorithms might not hold up unless the combination is done in a way that preserves those properties. To define security properties in a systematic way, the concept of semantic security is frequently used when providing proofs for cryptographic algorithms.

Semantic security is in modern security proofs most frequently defined through one of two definitions. Indistinguishability under a chosen plaintext attack (IND-CPA) entails that it is computationally infeasible for an attacker to distinguish between two different ciphertexts that were created from two different plaintexts, even if the attacker is able choose the plaintexts to be encrypted. In essence, ciphertexts should be indistinguishable from random noise, and as long as the encryption scheme is pseudo-non-deterministic, this property holds.

Indistinguishability under a chosen ciphertext attack (IND-CCA) is similar, but in addition the attacker may request decryption of ciphertexts other than the challenge. If a scheme is considered IND-CCA secure, it follows that it is IND-CPA secure as well, and IND-CCA is considered the standard for secure cryptosystems. When combining schemes, this is the property we would like to preserve.

Bindel et. al. [BBF+19] presents a method that allows a combined scheme to preserve the IND-CCA property as long as one of the constituent schemes and a Message Authentication Code (MAC) are IND-CCA secure, and provides a formal proof for this property. The method consists of performing an XOR operation on the symmetric keys generated by two KEMs and then calculating a MAC on the concatenation of the key encapsulations.

As all post-quantum cryptosystems submitted to the NIST process are required to be IND-CCA secure, and ECDH is IND-CCA secure with respect to a classical attacker, this method perfectly fits for application as a combiner for a hybrid scheme.

2.4 Wireguard

The Virtual Private Network (VPN) protocol WireGuard was developed and released by Donenfeld in 2015 with the aim of providing a VPN protocol with an emphasis of simplicity and auditability as compared to existing alternatives such as OpenVPN and IPsec [Don17]. In recent years, WireGuard has started to emerge as the preferred VPN protocol for many adopters, with several VPN service providers, such as NordVPN [Juo22] and ProtonVPN [Pro21], transitioning due to better performance results, and the protocol being integrated in the mainline Linux kernel in 2020.

To establish a symmetric key, WireGuard uses ECDH in its handshake to arrive at a shared secret. As a result, WireGuard is susceptible to an attack from an adversary with a sufficiently strong quantum computer that is capable of implementing Shor's algorithm efficiently.

After the handshake is performed, WireGuard uses ChaCha20-Poly1305 for symmetric key encryption, and Blake2s for hashing and message authentication purposes. Even though Grover's algorithm [Gro96] on a quantum computer might provide a quadratic speedup, both of these algorithms are still considered quantum-safe as long as the symmetric key length is doubled to achieve the same level of security. This essentially makes the asymmetric handshake component the only part of WireGuard that is particularly susceptible to a quantum computer attack.

2.4.1 Noise Protocol

WireGuard achieves a relatively small codebase compared to its alternatives through multiple methods. Of note, it implements its public key exchange through the use of the Noise Protocol framework [KNB19]. Noise uses set structures of tokens representing cryptographic primitives and message patterns to systematically implement cryptographic features, such as key exchange handshakes and message exchanges. Specifically, WireGuard uses the Noise_IKpsk2 pattern for its handshake.

In figure 2.1, the message token s defines that a static key share is being performed. In the case of WireGuard, this static key serves as the identifier of a party, and is pre-authenticated outside of the communication band. In practice, this may be done through some key distribution mechanism or simpler mechanisms such as a secure email transfer.

In message **A**, an initiator requests a handshake with a responder by sending a message containing their ephemeral public key e, their identity and static public key s. Of note, this static public key is already assumed known by the responder, and is not sent in plaintext, but encrypted and hashed so that the responder may verify it, for the purpose of identity hiding.

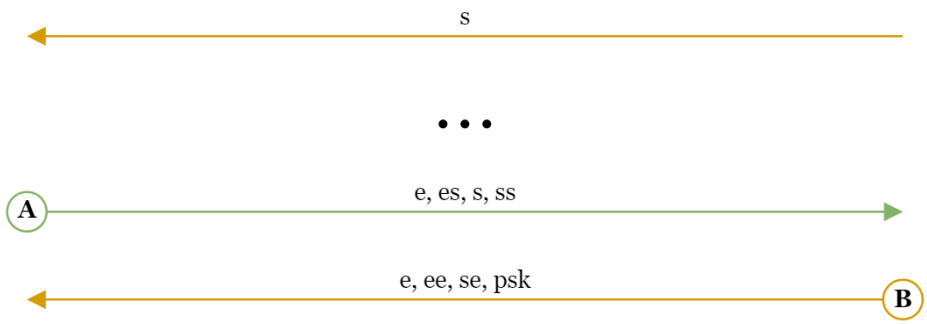


Figure 2.1: The Noise_IKpsk2 pattern.

Further, the token *es* signifies that they are calculating a shared secret using their ephemeral private key and the responders static public key, while the token *ss* indicates that they are calculating a shared secret using their static private key and the responder static public key.

The responder then calculates an equivalent *es* shared secret with their static private key and and the initiators ephemeral public key, and *ss* shared secret with their static private key and the initiators static public key, before responding with message **B**.

In message **B**, the responder replies with their ephemeral public key *e*, and indicates that they are calculating a shared secret *ee* using the initiators ephemeral public key and their ephemeral private key, as well as the calculation *se* indicating a shared secret using the initiators static public key and their own ephemeral private key. The *psk* token indicates that a pre-shared symmetric key is to be used as additional keying material, which is optional in WireGuard.

The shared secrets *ee*, *es*, *se*, *ss* and optionally *psk* are throughout the handshake combined into a session key using a key derivation function (KDF). The result of these calculations is a symmetric session key that incorporates the properties of the different shared secrets. Multiple properties are gained from mixing the ephemeral and static secrets, with the main ones being resistance against key-compromise impersonation and replay attacks, along with perfect forward secrecy which ensures that session keys stay uncompromised even if static secrets are compromised [Don15].

Being based on a DH variant key-exchange, the Noise_IKpsk2 pattern doesn't directly lend itself to usage with post-quantum cryptosystems. The first problem that arises is that the handshake pattern has to be modified to be based on a KEM triple of algorithms instead of the public and private key exchange scheme used by

the pattern.

The second problem isn't directly apparent from the Noise_IKpsk2 pattern, but is a result of the static and ephemeral secrets exchanged through the pattern. WireGuard uses these secrets to compute a chaining key that incorporates their properties. To create a hybrid implementation of WireGuard one would have to create a chaining key that incorporates components that achieve the same properties through the use of a KEM construction. As KEMs are inherently ephemeral, creating an equivalent to the *ss* shared secret is especially difficult.

2.4.2 Post-Quantum WireGuard

In 2021, Hülsing et al. [HNS+21] presented a fully post-quantum implementation of WireGuard called PQ-WireGuard, which uses a combination of Classic McEliece to replace the static component and a tweaked implementation of Saber that removes the Fujisaki-Okamoto transform to create a lightweight IND-CPA version of Saber for the ephemeral component. This implementation avoids Classic McEliece's inherent problem of extremely large key sizes by having them as pre-shared static keys, and considers IND-CPA to be adequate security for the ephemeral component.

PQ-WireGuard relies on the assumption that both Saber and Classic McEliece are and remain secure both under a classical and post-quantum paradigm. While Classic McEliece might be the post-quantum scheme that has existed for the longest without being broken, it has glaring problems in respect to performance and key sizes, while Saber is still a relatively new scheme and might very well be broken given further scrutiny and new discoveries.

As opposed to PQ-WireGuard, this project aims to research an alternative implementation that ensures security against a classical adversary should the assumed quantum-secure component be broken, in addition to the assumed properties against a quantum adversary that PQ-WireGuard provides, given the post-quantum component remains unbroken. Additionally, it aims to develop a version with flexibility in regards to which post-quantum KEM that should be used for testing purposes.

Chapter 3
Research Methodology

This chapter will describe the method in which the research project was performed. It will go into detail how the hybrid algorithm was designed and implemented as a handshake component in the BoringTun implementation of the VPN protocol WireGuard. Further, it will describe how testing results were gathered.

3.1 Algorithm Design

The following subsections describe how a KEM is constructed using ECDH, and how this construction can be combined with a post-quantum KEM to create a hybrid KEM.

3.1.1 From ECDH to ECKEM

To create a hybrid cryptosystem that uses ECDH in combination with a post-quantum KEM, the ECDH component has to be redefined to work as a KEM instead of being simply a key agreement scheme. The main components required to create a KEM from ECDH is a KDF and an encapsulation mechanism. Fittingly, WireGuard already uses HKDF as its KDF with Blake2s as its HMAC function which works adequately for a KDF, and ChaCha20 can serve as the encapsulation mechanism.

Algorithm 3.1 ECKEM.Encaps(pk, sk)

1: $k \leftarrow SharedSecret(pk, sk)$
2: $shk \leftarrow KDF(k)$
3: $c \leftarrow SymmetricEncryption(shk, k)$
4: **return** (c, shk)

Key generation works exactly the same as in ECDH, with a pair of public and private keys being generated. For encapsulation, the responder derives a shared key from the shared secret generated by the initiator's public key and the responder's private key using the KDF. This shared key is then used to encapsulate a generated

KEM key with the symmetric cryptosystem, which produces a ciphertext that is sent to the initiator, along with the responder's public key.

Algorithm 3.2 ECKEM.Decaps(pk, sk, c)

1: $k' \leftarrow SharedSecret(pk, sk)$
2: $shk' \leftarrow SymmetricDecryption(c, k')$
3: **return** shk'

For decapsulation, the same shared secret is generated with the initiator's private key and the responder's public key, and the shared key is derived through the KDF. The initiator then decapsulates the KEM key with the shared key. The end result is both parties having the same symmetric key. Following, this KEM-like construction of ECDH will be referred to as ECKEM.

With two KEMs, our ECKEM described and a post-quantum KEM (PQKEM), a combined hybrid KEM (HKEM) can be constructed. As the combination method, the XOR-then-MAC combiner as described in Bindel et al. [BBF+19] will be used, specified according to using ECKEM and a PQKEM selection as the component algorithms.

3.1.2 Key Generation

Algorithm 3.3 HKEM.KeyGen()

1: $pk_{ECKEM}, sk_{ECKEM} \leftarrow KeyGen_{ECKEM}()$
2: $pk_{PQKEM}, sk_{PQKEM} \leftarrow KeyGen_{PQKEM}()$
3: $pk_{HKEM} \leftarrow (pk_{ECKEM}, pk_{PQKEM})$
4: $sk_{HKEM} \leftarrow (sk_{ECKEM}, sk_{PQKEM})$
5: **return** (pk_{HKEM}, sk_{HKEM})

The hybrid key generation algorithm (Algorithm 3.3) does not require any additional operations to retain the security properties of the component KEMs. The main purpose of the key generation algorithm is to generate two pairs of public and private keys, one from the ECKEM and one from the PQKEM, and output these keys according to the definition of a KEM.

In the HKEM construction, the public keys from the two KeyGen algorithms are concatenated into a single key tuple, pk_{HKEM}. Similarly, the private keys are concatenated into a single key tuple sk_{HKEM}. These key tuples are then output from the hybrid KeyGen algorithm as the public and private key.

Algorithm 3.4 HKEM.Encaps(pk_{ECKEM}, pk_{PQKEM})

1: $(c_{ECKEM}, k_{ECKEM}||k_{MAC-EC}) \leftarrow Encaps_{ECKEM}(pk_{ECKEM})$
2: $(c_{PQKEM}, k_{PQKEM}||k_{MAC-PQ}) \leftarrow Encaps_{PQKEM}(pk_{PQKEM})$
3: $k_{HKEM} \leftarrow k_{ECKEM} \oplus k_{PQKEM}$
4: $k_{MAC} \leftarrow (k_{MAC-EC}, k_{MAC-PQ})$
5: $c \leftarrow (c_{ECKEM}, c_{PQKEM})$
6: $\tau \leftarrow MAC_{k_{MAC}}(c)$
7: **return** $((c, \tau), k_{HKEM})$

3.1.3 Key Encapsulation

The hybrid encapsulation algorithm (Algorithm 3.4) takes as input the public key tuple of the initiator. The encapsulation algorithm of the two component KEMs generate two keys and two ciphertexts containing these keys, encrypted with the corresponding public keys extracted from the initiator's public key tuple. These keys are then separated into two components each, a KEM key and a MAC key for each component. A bitwise XOR is then performed on the two KEM keys.

The MAC keys and the ciphertexts are concatenated into a single hybrid MAC key and ciphertext, and the ciphertext gets signed using the MAC function, in our case Blake2s, with the hybrid MAC key. The algorithm outputs the ciphertext concatenated with the signed tag, and the hybrid KEM key. The encapsulated key along with the tag is sent to the initiator, while the KEM key is kept secret by the responder.

3.1.4 Key Decapsulation

Algorithm 3.5 HKEM.Decaps((sk_{ECKEM}, sk_{PQKEM}),$((c_{ECKEM}, c_{PQKEM}), \tau)$)

1: $k'_{ECKEM}||k'_{MAC-ECKEM} \leftarrow Decaps_{ECKEM}(sk_{ECKEM}, c_{ECKEM})$
2: $k'_{PQKEM}||k'_{MAC-PQKEM} \leftarrow Decaps_{PQKEM}(sk_{PQKEM}, c_{PQKEM})$
3: $k'_{HKEM} \leftarrow k'_{ECKEM} \oplus k'_{PQKEM}$
4: $k'_{MAC} \leftarrow (k'_{MAC-EC}, k'_{MAC-PQ})$
5: **if** $\mathbf{MVf}_{k'_{MAC}}((c_{ECKEM}, c_{PQKEM}), \tau) = 0$: **then**
6: **return** $\perp$
7: **else**
8: **return** k'_{HKEM}

The hybrid decapsulation algorithm (Algorithm 3.5) works as the inverse of the encapsulation algorithm, and takes as input the received ciphertext along with the tag, and the initiator's private key tuple. The algorithm uses the decapsulation algorithms of the component KEMs with their respective private keys to extract the KEM and MAC key components from the received ciphertexts.

The algorithm then constructs the hybrid KEM key and MAC key in the same way as the encapsulating algorithm. It then checks the ciphertext against the tag and the constructed MAC key, and rejects the handshake if the verification fails. A failure of the verification would indicate that the received message has been either forged or tampered with in some way. Finally, given a successful verification, the algorithm outputs the composite KEM key to the initiator, resulting in both parties having the same shared secret hybrid KEM key.

3.2 Implementation Design

The following subsections describe considerations and decisions taken when designing and implementing the hybrid cryptosystem in a userspace version of WireGuard.

3.2.1 Noise Protocol Changes

As mentioned in section 2.4.1, the Noise_IKpsk2 pattern assumes that a non-interactive DH variant is used for the key exchange, and does not lend itself directly to being used with KEMs. To make a version of the pattern that uses KEM constructions instead of DH, the calculation tokens need to be replaced with keying material that preserves the properties of the four static and ephemeral randomness combinations.

PQ-WireGuard [HNS+21] with its KEM-based handshake serves as a model for how to derive the keying material for all of the combinations except for the static-static pairing through the use of key encapsulation. An encapsulated key is inherently ephemeral as it is generated during the instance of performing the encapsulation. Encapsulating such a key with the static public key and the ephemeral public key of the other party provides the functionality of the ephemeral-static, ephemeral-ephemeral and static-ephemeral shared secrets.

The main purpose of the static-static pairing in WireGuard's implementation of the Noise_IKpsk2 handshake is to prevent unknown-keyshare attacks [HRSS17]. To provide a construction that mitigates this attack, the optional *psk* token is used to indicate that a hashed XOR of the initiator and responder's static public keys is to be used as keying material. With the static public keys already being considered secret and never transmitted in plaintext, this value additionally serves as material to mix static-static randomness into the session key.

Figure 3.1 shows the new Noise pattern with changes from the Noise_IKpsk2 pattern indicated in blue. A new token, $enc(a_b)$, is introduced, indicating an encapsulated key using the public key a originating from the communicating party b. In this new pattern, the changed tokens no longer simply indicate a calculation to be

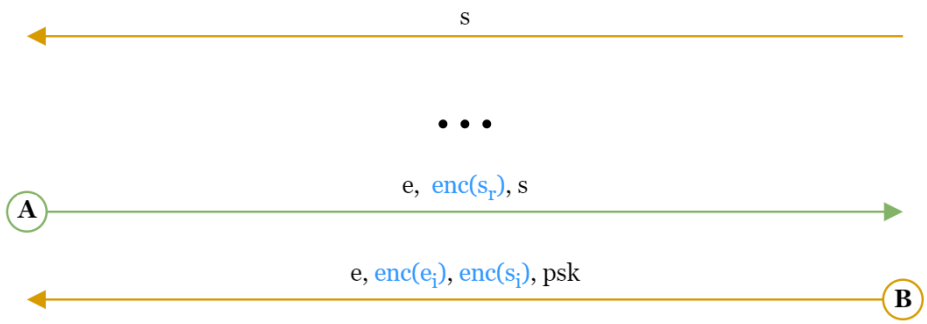


Figure 3.1: Modified Noise_IKpsk2 pattern.

performed, but an actual value sent as part of the message along with an instruction to decapsulate the value. The *psk* token is no longer optional, and indicates that the hashed XOR of the static public keys should be mixed into the session key.

Key	Initiator Calculation	Responder Calculation
K1: es	$(ct_1, \mathbf{shk_1}) \leftarrow \text{HKEM.Encaps}(spk_r)$	$\mathbf{shk_2} \leftarrow \text{HKEM.Decaps}(ssk_r, ct_2)$
K2: ee	$\mathbf{shk_2} \leftarrow \text{HKEM.Decaps}(ssk_r, ct_2)$	$(ct_2, \mathbf{shk_2}) \leftarrow \text{HKEM.Encaps}(epk_i)$
K3: se	$\mathbf{shk_3} \leftarrow \text{HKEM.Decaps}(ssk_i, ct_3)$	$(ct_3, \mathbf{shk_3}) \leftarrow \text{HKEM.Encaps}(spk_i)$
K4: ss	$\mathbf{psk} \leftarrow (H(spk_r \oplus spk_i))$	$\mathbf{psk} \leftarrow (H(spk_r \oplus spk_i))$

Figure 3.2: Key material equivalents with KEM-based construction.

With the new pattern, figure 3.2 shows the new key equivalents in bold, and how they are derived respectively by the initiator and responder. In the figure, *ct* denotes a ciphertext encapsulating a shared KEM key *shk*, *spk* a static public key, *ssk* a static private key, *epk* an ephemeral public key and *esk* an ephemeral private key. Subscript letters denote the originator of the key and subscript integers chronologically differentiate the ciphertext encapsulations and shared KEM keys generated throughout the handshake.

3.2.2 Post-quantum Algorithms Library

When testing for multiple different post-quantum cryptosystems, implementing each cryptosystem that warrants testing from the ground up would be an insurmountable and error-prone task. There are multiple software libraries that provide implementations of several NIST candidates, with the Open Quantum Safe (OQS) project's

liboqs library [SM17] being the most prominent option with implementations of every current NIST candidate algorithm, along with several versions with different parameters.

A problem arises when using an external library along with the C implementation of WireGuard. This implementation is developed for the Linux kernel to improve performance, but comes with the caveat of not being able to use the C standard library of functions. Liboqs and other post-quantum algorithm libraries are all dependent on the use of these functions.

This leaves two options. Either implementing several post-quantum algorithms from scratch without the use of the C standard library, or making the implementation based on a user-space implementation of WireGuard instead. Luckily, there are several versions of WireGuard, with for example WireGuard-go [Don19], implemented in the Go programming language, or Cloudflare's Rust implementation of WireGuard, BoringTun [Clo19], both of which are user-space implementations and open source.

3.2.3 BoringTun

The BoringTun implementation was the natural choice to build the hybrid implementation from. BoringTun provides increased performance metrics as compared to WireGuard-go. Rust as a programming language provides several advantages compared to Go with memory safety, simple integration with C libraries such as liboqs and generally being a more accessible language to work with.

The main part of BoringTun that had to be modified was the handshake component. The first task was to change the ECDH handshake construction to the KEM-based construction described in section 3.2.1. With simple equivalents to the session key components and a similar messaging pattern, this mainly entailed changing the existing shared secret calculations to decapsulations, and populating the handshake messages with additional values for the encapsulations when generating handshake initiations and responses. Parameter declarations were changed from ECDH-based keys and shared secrets to new keys and types defined by our new module HKEM.

The second task was to construct the hybrid KEM mechanism. The ECKEM component was constructed with HMAC with Blake2s as its KDF and ChaCha20 as the encapsulation mechanism as described in section 3.1.1. The key generation, encapsulation and decapsulation algorithms were then constructed as described in algorithms 3.3, 3.4 and 3.5 respectively, with Blake2s in keyed mode as the MAC algorithm.

New structures for public and secret keys, shared secrets, MAC keys, MAC tags

and ciphertexts were created as items that could be referenced through the use of the HKEM module in the same way that the originial Boringtun implementation references the x25519_dalek ECDH module.

The implementation had to be made with a simple mechanism for changing the post-quantum component used in the hybrid algorithm. With KEMs being constructed as a set triplet of algorithms with set inputs and outputs, the only factor that would change apart from function calls between the different algorithms were expected parameter sizes for values such as keys and ciphertext.

Otherwise, all liboqs KEM functions are referenced using foreign function integration with the templates $OQS\_KEM\_\{scheme\}\_\{keypair/encaps/decaps\}$, where *scheme* is the algorithm and version, such as for example Kyber_1024. The selected algorithms were instantiated with their parameter sizes and liboqs references in separate modules and configured as feature selections upon compilation, defined in the HKEM module.

3.2.4 Post-Quantum Algorithm Choices

When selecting which algorithms to use, there are two main considerations. The first consideration is which version of the algorithm to use. The NIST candidates have different versions with different parameters according to five security categories defined in the standardisation process' call for proposals [Nat16]. The highest level, NIST level V, requires parameters that are capable of resisting an attack with resources comparable to those required to break a block cipher with 256 bits of security.

The NIST process is mainly focusing algorithms by their first three levels, which at most is analogous to a block cipher with 192 bits of security. With the hybrid system being designed to stay secure for several decades, and expecting computational attacking capabilities to be significantly improved throughout those years, this project will be testing algorithms with parameters according to NIST level V.

The second consideration is the public key and ciphertext sizes of the algorithms with parameters according to NIST level V. These two values are the largest values that are transmitted as part of a handshake and together make up most of the datagrams of message A and B shown in figure 3.1. The sizes of the selected KEMs from round 3 of the NIST process along with alternative candidates are shown in table 3.1 with their NIST level V parameters.

With WireGuard being based on UDP, the maximum transmission unit (MTU) for a datagram is constrained by the MTU for an IP packet, which is roughly 65 000 bytes [Pos81]. This limit practically rules out both Classic McEliece and FrodoKEM

KEM	PK (bytes)	SK (bytes)	Ciphertext (bytes)
Classic-McEliece-6688128	1044992	13892	240
Kyber1024	1568	3168	1568
NTRU-HPS-4096-821	1230	1590	1230
FireSaber	1312	3040	1472
FrodoKEM-1344-AES	21520	43088	21632
HQC-256	7245	7285	14469

Table 3.1: Public Key, Secret Key and Ciphertext sizes for NIST finalist and alternative KEMs with NIST level V parameters [SM17].

as eligible, as both messages in the handshake would require fragmentation on the IP layer, which is ill advised in under normal circumstances [BBH+20], and particularly in cryptographic settings. IP fragmentation is particularly vulnerable to packets being dropped, and WireGuard will not initiate a new session before 15 seconds have passed since the last packet arrived [Don15], which would likely result in several failed connections if these algorithms were to be used.

There is also a further constricting MTU of 1500 bytes for a link layer frame, which results in about 1400 bytes of information per package when accounting for headers. This limit is set to 1420 bytes in WireGuard by default. This segmentation is done at the application layer. This provides a certain degree of reliability as compared to IP fragmentation, but with larger packet sizes, more segmentation is required leading to more frame headers, which again leads to more of the bandwidth and processing time being used for fragmentation.

Apart from the previously mentioned schemes, the secondary MTU limit only becomes somewhat of a problem for HQC-256, which requires at least 36 183 bytes of data for the two ciphertexts and the ephemeral key in message **B**, just accounting for the post-quantum component. This results in a segmentation into at least 26 packet segments, which through preliminary testing resulted in several failed handshakes. With Classic McEliece, FrodoKEM and HQC being eliminated, we are left with three algorithms to test: Kyber, Saber and NTRU.

3.3 Testing

The following subsections describe the testing environment in which the testing was performed, and the tools and methods used to gather the results.

3.3.1 Testing Environment

With a hybrid implementation of BoringTun ready, we proceed to set up a testing environment. A virtual Ubuntu workstation was set up on resources allocated from NTNU's Openstack implementation SkyHigh [Obr22], with the following specifications:

- OS: Ubuntu 22.04.1
- RAM: 4 GB Virtual DDR4-3200
- CPU: 4 Virtualised Intel Haswell Cores @ 2.0 GHz

The operating system was initialised as a minimal installation of Ubuntu, and testing was done with a console terminal being the only application running on the workstation, besides the BoringTun application being compiled and tested through the use of the Rust package manager Cargo. During the tests, packets are sent through a tunnel through localhost.

3.3.2 Testing Methodology

For testing, an internal benchmarking module was made for the handshake component through the use of the Criterion package [AH22], which is used in the original implementation of BoringTun. Criterion benchmarks an internal routine by first performing a warmup by running the routine for a given time, before it performs measurements for a set measurement time with a set sample-size.

The average time for a sample is calculated by dividing the time taken for a sample to complete by the number of iterations of the routine performed by the sample. For each sample completed, the number of iterations of the routine are increased linearly by a factor calculated from the estimate of an iteration completion during the warmup.

The results presented in the following chapter are results of benchmarks using a measurement time of 300 seconds with a sample size of 100. The measured times include both the runtime of algorithmic calculations and packet transmission.

The handshake consists of four message transmissions; Initiation with a hashed static public key, message **A** and **B** shown in figure 3.1, along with a single confirmation packet sent using the final session key.

The original implementation of Boringtun (version 0.5.2) [Clo19] was benchmarked for comparison purposes, in addition to hybrid implementations using Kyber1024, FireSaber and NTRU-HPS-4096-821 respectively as their post-quantum component.

Chapter 4
Results

This chapter presents the results of testing different post-quantum components in a hybrid system in the form of performance metrics. Additionally, this chapter provides supplementary data related to the post-quantum components that helps to explain and analyse the results.

4.1 Test Results

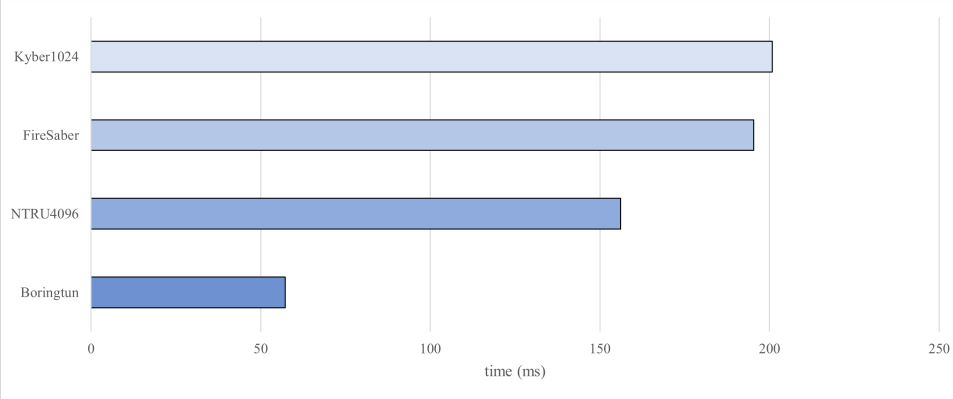


Figure 4.1: Bar graph of average handshake completion times for different implementations of Boringtun.

Figure 4.1 shows a bar graph of the average time in milliseconds for a handshake to complete over 100 Criterion samples. The original implementation of Boringtun had an average time of 57.2 ms, and is significantly faster than the hybrid implementations. Among the hybrid implementations, the NTRU–HPS-4096-821 version stands out with an average time of 156.1 ms, while the FireSaber and Kyber1024 versions perform quite similarly at 195.4 and 200.8 ms respectively.

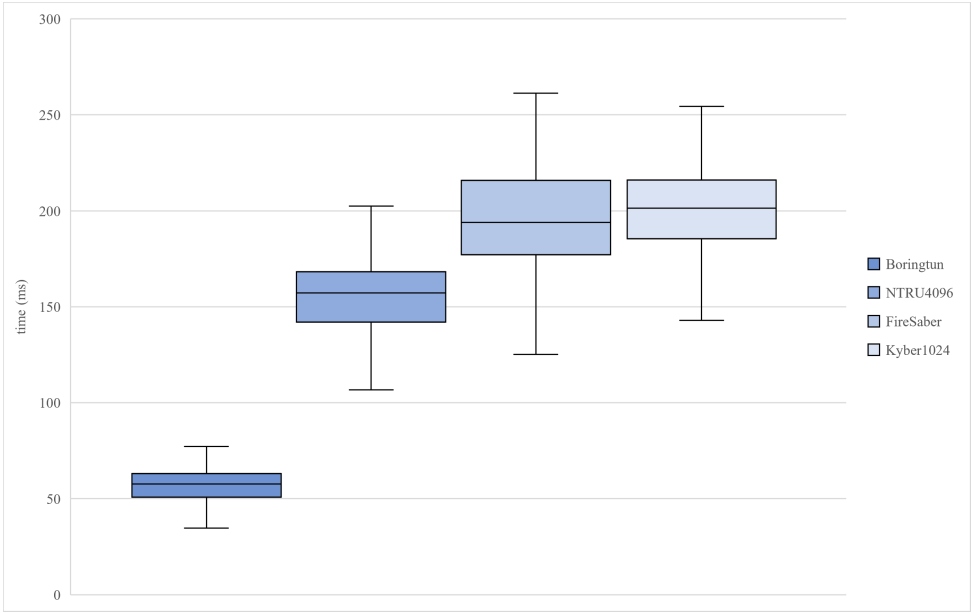


Figure 4.2: Box plot of sample data from different implementations of Boringtun.

Figure 4.2 shows a standard box plot of the distribution of all 100 Criterion testing samples. From the medians in the plot being close to the averages in figure 4.1 and from the distributions having a low amount of skew, we can infer that the results from each of the tests are relatively normally distributed.

The original implementation of Boringtun has a significantly lower variance in the time it takes to complete a handshake. Among the hybrid implementations, NTRU again outperforms with both a lower variance and median. FireSaber outperforms Kyber by a margin on the average, but does so with greater variance.

4.2 Context data

The following subsections will present additional data that serves to put the gathered results into context, and explain and discuss the results in Chapter 5.

4.2.1 KEM subroutine benchmarks

OQS provides up-to-date benchmarks for their KEM implementations in liboqs [SM22]. Using this data, we can make estimates of how much the key generation,

encapsulation and decapsulation calculations impact the time for a handshake to complete.

Of note, OQS's testing environment differs from this project's testing environment with a stronger processor. The benchmarking appears to be single-threaded and ran on a single core with a nominal speed of 2.5 GHz, as opposed to similar execution with a nominal speed of 2.0 GHz in this project's testing environment. Accounting for potential overclocking, it's safe to assume that the corresponding times for this project's test environment are in reality larger by a factor between 1.25-1.5.

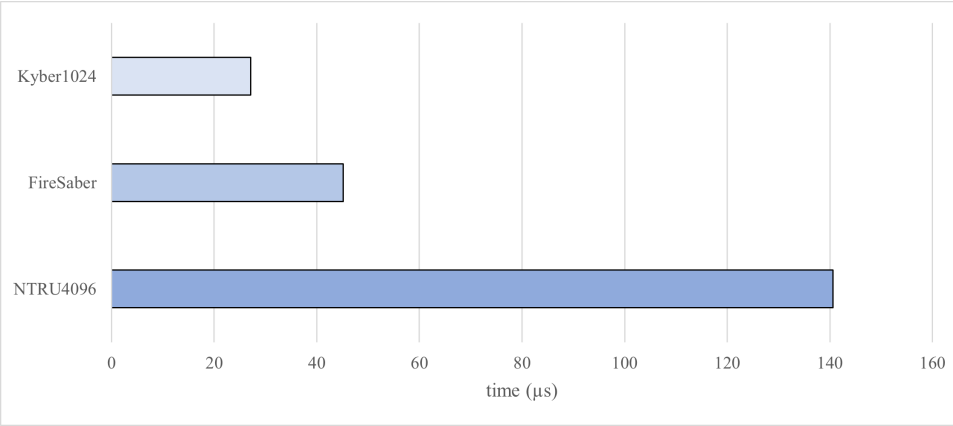


Figure 4.3: Average key generation routine completion time, from OQS benchmark [SM22].

Figures 4.3, 4.4 and 4.5 respectively show the time for a key generation, encapsulation and decapsulation routine to complete in microseconds. As part of a handshake in our hybrid implementation, key generation is called at two instances, while encapsulation and decapsulation are called at three instances each.

The total time in microseconds for all eight of the post-quantum component function calls is shown in figure 4.6 . In the case of all three post-quantum KEMs, the calculations specifically related to that component amount to less than one percent of the total handshake time, even when accounting for a increase in time averages by a factor of 1.5.

4.2.2 Message size contributions

With KEM calculations having a negligible impact on the handshake completion time, the differences in the test results might rather be explained by the sizes of messages transmitted as part of the handshake. Using the values for the different

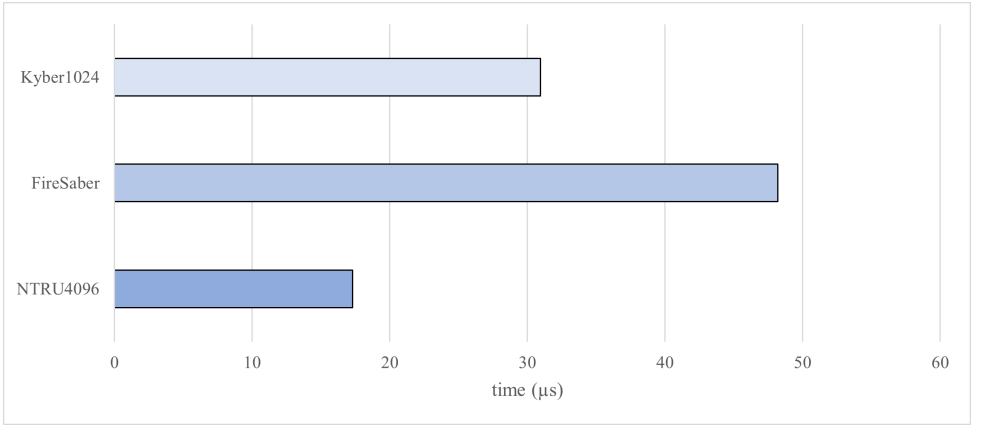


Figure 4.4: Average encapsulation routine completion time, from OQS benchmarks [SM22].

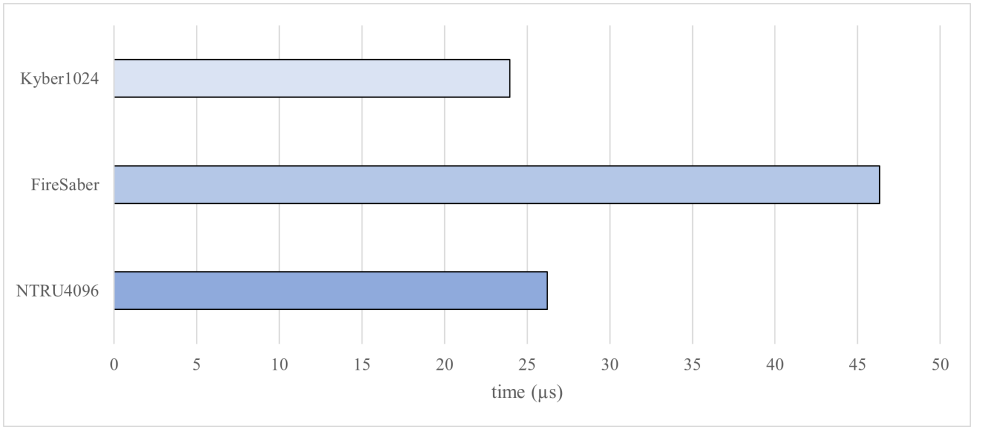


Figure 4.5: Average decapsulation routine completion time, from OQS benchmarks [SM22].

KEM functions given in table 3.1, the post-quantum component's contribution to the message sizes can be calculated.

The figures in this section does not take into account the contribution from the classical component, which in most cases produces an additional parameter of 32 bytes when a post-quantum parameter is generated.

Message **A** consists of an ephemeral public key, a ciphertext and a hashed static

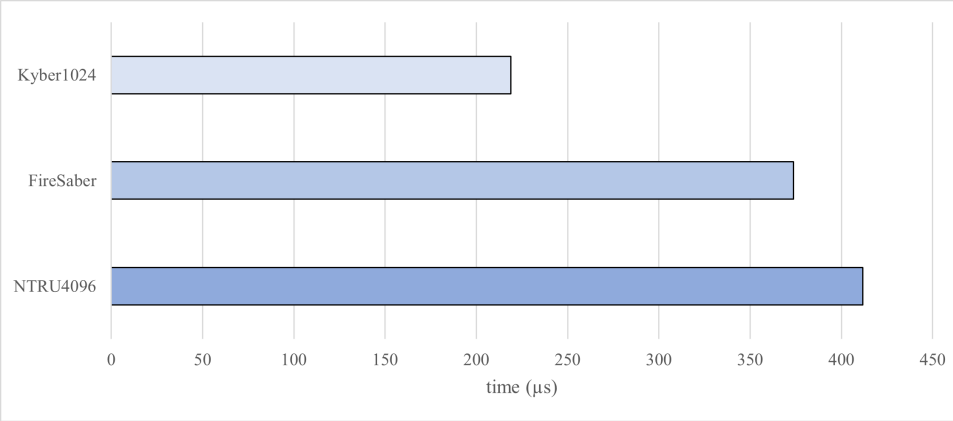


Figure 4.6: Total time for two key generation routines, three encapsulation routines and three decapsulation routines, based on OQS benchmarks [SM22].

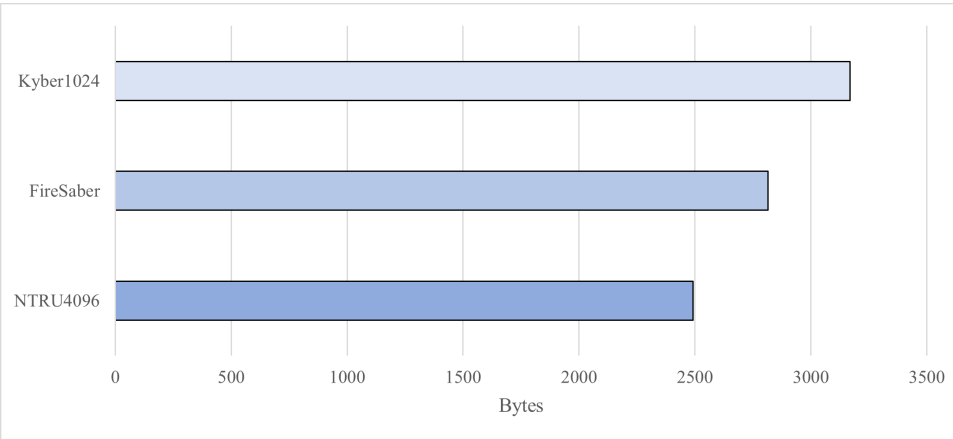


Figure 4.7: Total size contribution from the post-quantum component to the size of message **A** in table 3.1.

public key with a size of 32 bytes. Figure 4.7 shows each post-quantum component's total contribution to the size of message **A**. Keeping in mind the 1420 byte limit in the link layer, Kyber's contribution to the message size requires fragmentation into at least three datagrams, while Saber and NTRU need at least two. Considering the additional data required for a MAC tag for the encapsulation, as well as classical components, Saber's implementation is likely to require fragmentation into at least three datagrams, while NTRU remains at two.

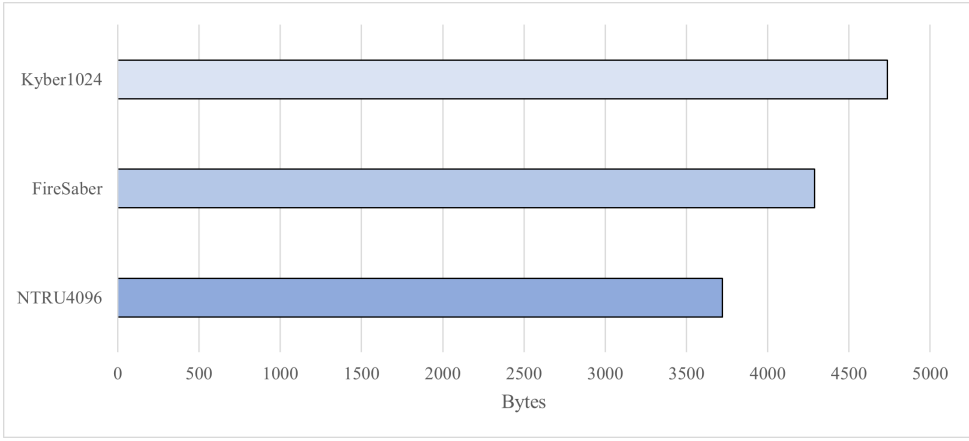


Figure 4.8: Total size contribution from the post-quantum component to the size of message **B** in table 3.1.

Message **B** consists of one ephemeral public key, two ciphertexts and a 32 byte hash of the XOR of both static public keys. Figure 4.8 shows each post-quantum component's total contribution to the size of message **B**. Kyber and Saber's contributions require fragmentation into at least four datagrams, while NTRU requires at least three. In this case, the additional data from two MAC tags and the classical component is not likely to require additional fragmentation.

In total, Kyber and Saber's implementation requires at least nine datagrams for a full handshake: One for the sharing of the hashed static public key, three for message **A**, four for message **B** and at least one for the final confirming message. NTRU's implementation requires at least seven datagrams for a full handshake: One for the sharing of the hashed static public key, two for message **A**, three for message **B** and at least one for the final confirming message.

Chapter 5
Discussion

This chapter aims to discuss and analyse the gathered results with respect to the research questions put forward in section 1.4. Further, it will discuss how the project's findings might contradict NIST's assumptions when selecting Kyber as the sole standardised lattice-based KEM.

5.1 Performance Impact

The results shown in figure 4.1 give a clear view of the performance cost of replacing a classical system with a hybrid system as described in this project. Putting aside latency and bandwidth, the time cost amounts to about three to four times as much as that of a fully classical system for a handshake routine. This factor could very well be different for other use cases than WireGuard, where for example static components are unnecessary or authentication is done through a different method, but it is unlikely to increase by a significant order of magnitude.

In the case of WireGuard and in most other cases, the purpose of the asymmetric handshake is to establish a session key to be used for symmetric encryption of further messaging. As a result, the handshake often represents only a small part of the traffic it's a part of.

In WireGuard, session keys are renewed through a new handshake every 165 seconds by default as long as the session remains active, though this interval can be modified. Thus, the actual impact of the handshake heavily depends on how much traffic is transmitted using the session key during those intervals. Nonetheless, this impact is strictly decreasing with an increasing amount of traffic, and will never exceed the factor the handshake time is increased by.

The main contributor to the time cost of a handshake appears to be the size of the public key and the ciphertext of the post-quantum component, while the time to perform calculations seems to be having a negligible impact. This is seen from the

results shown in figure 4.6 in conjunction with figure 4.1, where even though Kyber is generally the fastest to calculate and NTRU is the slowest, NTRU is fastest in application, while Kyber is the slowest.

Specifically, how many datagrams a message packet has to be fragmented into seems directly proportional to the time for a handshake to complete. Section 4.2.2 shows how both Kyber and Saber require fragmentation into at least nine datagrams for an entire handshake, while NTRU requires seven. Comparing this fact to the results figure 4.1, the similarity of the handshake times for Kyber and Saber and the factor of which NTRU outperforms them makes perfect sense.

5.2 Post-quantum KEM comparison

WireGuard is in its design cryptographically opinionated. This entails that through its design, it chooses specific cryptographic primitives for purposes such as key establishment, authentication, hashing and encryption. It does this to reduce complexity, maintain a minimal codebase and to be able to decisively respond to newly discovered vulnerabilities [Don17].

Following this design goal, for a quantum-resistant implementation of WireGuard, a specific post-quantum primitive has to be chosen for the handshake component. With SIKE being proven not secure and code-based schemes being massively inefficient with regards to their public key and ciphertext sizes, our current options are unfortunately limited to lattice-based schemes.

Relying on the single computationally hard problem of solving the SVP means that if one of these schemes are proven to not be secure, most likely all of them are. If anything, this further motivates using post-quantum primitives in conjunction with classical ones.

Among the post-quantum primitives tested throughout this project, NTRU stands out as the clear victor for this use case. Being the only NIST candidate with both public key and ciphertext sizes well below the MTU for the link layer with NIST level V parameters, this seems likely to be the case for most other use cases as well.

With these results in mind, NIST's decision to standardise Kyber and subsequently stop considering other lattice-based primitives seems like a rushed one. NIST backs their decision up by referring to Kyber's faster calculation times, exemplified in figure 4.6, but fails to evaluate the schemes with the MTU of link layer datagrams in mind.

Unless the entirety of the internet infrastructure changes to allow for larger datagram sizes, Kyber might in fact be the least efficient lattice-based option with respect to NIST level V, like the results of this project exemplifies, even though it

might outperform the others with NIST level III which the standardisation process prioritises.

When designing a system that is expected to remain secure for several decades, and potentially against active quantum attackers, I believe that NIST level III will be inadequate. Subsequently, I reason that the ability for a post-quantum KEM to keep its public key and ciphertext sizes well below the MTU limit of 1500 bytes for all levels of security should be valued higher when considering standardisation.

5.3 Utility of hybrid cryptosystems

The usefulness of transitioning to a cryptosystem that incorporates post-quantum security ultimately depends on the use case and the sensitivity of the information that is to be encrypted. The implementation described in this project adds an additional 100-150 ms to the process of establishing a VPN session.

It's worth noting that as most communication protocols utilise symmetric cryptography for most of their traffic, the initial session establishment using asymmetric cryptography serves as only a small part of the total traffic that is exchanged. It is however the main instance where a user interacts with the system and expects a response in the form of a session establishment, while the remaining security features of the session are performed without specific user interaction.

Card et al. [CRM91] concludes that response times of 100 ms results in the user experiencing that the system is reacting to input instantaneously, while a response time of 1000 ms serves as the limit for when a user starts noticing a significant delay. Every version of a hybrid system tested throughout this project stays within this range with respect to their handshake component. Although a minor change in latency might be noticed, the end user experience is not likely to be impacted in a significant matter by implementing post-quantum security in this matter.

Thus, for instances where one wishes to encrypt information that might be sensitive for the foreseeable future, such as medical and economic data, there is both motivation to implement post-quantum security, utility gained from it and possibility of implementation without negatively impacting the user experience in a significant matter.

5.4 Findings

In NIST's status report on the third round of the standardisation process [AAC+22] they express that one of their most significant factors in their selection of Kyber over NTRU is due to NTRU's performance, particularly key generation, being less

efficient. The results of this project show that this focus on the efficiency of the calculations might be an improper prioritisation when selecting among post-quantum cryptosystems, especially when it comes to lattice-based systems with very similar metrics.

When it comes to putting these cryptosystems into real-world use, the size of publicly transmitted parameters such as public keys and ciphertexts appears to be the main contributor to the efficiency of the scheme by a significant margin. This is especially significant when these sizes are close the MTU of 1500 bytes, which all of the lattice-based schemes are when configured for NIST security level V. Kyber being the only scheme among them that breaks this limit serves as a more significant counterargument against its standarisation in place of Saber or NTRU than NIST might have considered.

Chapter 6
Conclusion

6.1 Summary

This thesis presents an applicable method to implement post-quantum key-exchanges in tandem with classical asymmetric key-agreement protocols such as ECDH by constructing a hybrid system. Using this method to implement such a system in a user-space version of WireGuard, it presents performance metrics with different post-quantum KEMs as the post-quantum component.

The results show that among the post-quantum KEMs submitted to the NIST standardisation process, NTRU outperforms all other algorithms with respect to NIST's security level V, performing handshake completions approximately 20% faster than the other lattice-based algorithms Kyber and Saber.

Further, the results shed light on the importance of relatively small public key and ciphertext sizes for the performance of a post-quantum KEM in a real-world application. Particularly, it shows that KEMs with plaintext and ciphertext sizes well below the MTU of 1500 bytes for the link layer perform significantly better than KEMs that exceed that limit in today's internet infrastructure.

Finally, the work shows that implementing post-quantum cryptography through the use of a hybrid system is already feasible. Though it adds a significant impact on the time required to establish a session, the impact amounts to fractions of a second, which is unlikely to result in a significantly worse user experience.

Regrettably, this project was unable to analyse post-quantum components based on other quantum-resistant problems than the SVP. Preliminary testing was done with code-based schemes, but their excessively large parameters quickly proved to be unmanageable, and resulted in an unreliable implementation that did not warrant application. Initially, the option of testing SIKE was intended, but developments resulting in the system being considered insecure while the project was underway

eliminated the system from being an option.

In December of 2022, around the same time as results were gathered for this project, both Saber and NTRU were removed from the liboqs repository [Ope23], presumably due to Kyber being standardised and thereby removing other lattice-based systems from consideration for standardisation. With the results of this project showing a use case where Kyber has the worst performance metrics among the lattice-based systems, these changes will hopefully be reverted before the next version of liboqs is released.

In any case, the standardisation of Kyber and the removal of other lattice-based cryptosystems from consideration might have been a rushed decision, with faulty priorities. While Kyber may be perfectly adequate for certain applications, cases such as the one shown in this project shows that other lattice-based schemes have more merit in others, and should not be ruled out for further consideration.

6.2 Further Work

There are many more use cases where hybrid implementations should be further analysed. Work has been done on testing hybrid systems in TLS and SSH [CPS19], but many applications and network protocols still remain untested. Analysis with different levels of security for different use cases could very well lead to different conclusions.

WireGuard is special in how it is cryptographically opinionated, while most systems have a degree of cryptographic agility, where the ciphersuite is negotiated as part of the handshake. Further work into analysing how to negotiate different combinations of key-agreement mechanisms while avoiding combinatorial explosion is a research topic of interest.

Finally, the results from this project show that the impact and significance of parameter sizes for post-quantum KEMs is a research topic that warrants further study and experimentation.

References

[AH22] J. Aparicio and B. Heisler. «Criterion.rs». (2022), [Online]. Available: https://github.com/bheisler/criterion.rs (last visited: Jan. 12, 2023).

[BBF+19] N. Bindel, J. Brendel, *et al.*, «Hybrid key encapsulation mechanisms and authenticated key exchange», in *International Conference on Post-Quantum Cryptography*, Springer, 2019, pp. 206–226.

[BBH+20] R. Bonica, F. Baker, *et al.*, «IP fragmentation considered fragile», RFC 8900, IETF, September, Tech. Rep., 2020.

[BCL+17] D. J. Bernstein, T. Chou, *et al.*, «Classic mceliece», 2017.

[BDK+18] J. Bos, L. Ducas, *et al.*, «CRYSTALS-Kyber: A CCA-secure module-lattice-based KEM», in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, IEEE, 2018, pp. 353–367.

[Ben80] P. Benioff, «The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines», *Journal of statistical physics*, vol. 22, no. 5, pp. 563–591, 1980.

[CCH+19] M. Campagna, C. Costello, *et al.*, *Supersingular isogeny key encapsulation*, 2019.

[CD22] W. Castryck and T. Decru, «An efficient key recovery attack on SIDH (preliminary version)», *Cryptology ePrint Archive*, 2022.

[CK17] R. Choi and K. Kim, «A classification of lattice-based trapdoor functions», in *Proceedings of the Forty-Fourth Symposium on Cryptography and Information Security*, 2017.

[Clo19] Cloudflare. «Boringtun: Userspace WireGuard implementation in rust». (2019), [Online]. Available: https://github.com/cloudflare/boringtun (last visited: Jan. 12, 2023).

[CLSZ95] I. L. Chuang, R. Laflamme, *et al.*, «Quantum computers, factoring, and decoherence», *Science*, vol. 270, no. 5242, pp. 1633–1635, 1995.

[CN22] H. Collins and C. Nay, «IBM unveils 400 qubit-plus quantum processor and next-generation IBM quantum system two», *IBM*, Nov. 9, 2022. [Online]. Available: https://newsroom.ibm.com/2022-11-09-IBM-Unveils-400-Qubit-Plus-Quantum-Processor-and-Next-Generation-IBM-Quantum-System-Two (last visited: Jan. 12, 2023).

[Cop94] D. Coppersmith, «An approximate fourier transform useful in quantum factoring», *arXiv preprint quant-ph/0201067*, 2994.

[CPS19] E. Crockett, C. Paquin, and D. Stebila, «Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH.», *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 858, 2019.

[CRM91] S. K. Card, G. G. Robertson, and J. D. Mackinlay, «The information visualizer, an information workspace», in *Proceedings of the SIGCHI Conference on Human factors in computing systems*, 1991, pp. 181–186.

[DH76] W. Diffie and M. Hellman, «New directions in cryptography», *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[DKSV18] J.-P. D'Anvers, A. Karmakar, *et al.*, «Saber: Module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM», in *Progress in Cryptology – AFRICACRYPT 2018*, Springer, 2018, pp. 282–305.

[Don15] J. A. Donenfeld. «WireGuard protocol & cryptography». (2015), [Online]. Available: https://www.wireguard.com/protocol/ (last visited: Jan. 12, 2023).

[Don17] J. A. Donenfeld, «WireGuard: Next generation kernel network tunnel.», in *NDSS*, 2017, pp. 1–12.

[Don19] J. A. Donenfeld. «sWireGuard-Go». (2019), [Online]. Available: https://git.zx2c4.com/wireguard-go (last visited: Jan. 12, 2023).

[FMMC12] A. G. Fowler, M. Mariantoni, *et al.*, «Surface codes: Towards practical large-scale quantum computation», *Physical Review A*, vol. 86, no. 3, p. 032 324, 2012.

[GCR22] A. A. Giron, R. Custódio, and F. Rodríguez-Henríquez, «Post-quantum hybrid key exchange: A systematic mapping study», *Journal of Cryptographic Engineering*, pp. 1–18, 2022.

[GM19] V. Gheorghiu and M. Mosca, «Benchmarking the quantum cryptanalysis of symmetric, public-key and hash-based cryptographic schemes», *arXiv preprint arXiv:1902.02332*, 2019.

[GP13] B. Gellman and L. Poitras, «U.S., British intelligence mining data from nine U.S. internet companies in broad secret program», *The Washington Post*, Jun. 7, 2013. [Online]. Available: https://www.washingtonpost.com/investigations/us-intelligence-mining-data-from-nine-us-internet-companies-in-broad-secret-program/2013/06/06/3a0c0da8-cebf-11e2-8845-d970ccb04497_story.html (last visited: Jan. 12, 2023).

[Gro96] L. K. Grover, «A fast quantum mechanical algorithm for database search», in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 212–219.

[HNS+21] A. Hülsing, K.-C. Ning, *et al.*, «Post-quantum WireGuard», in *2021 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2021, pp. 304–321.

[HRSS17] A. Hülsing, J. Rijneveld, *et al.*, «High-speed key encapsulation from NTRU», in *International Conference on Cryptographic Hardware and Embedded Systems*, Springer, 2017, pp. 232–252.

[IBM22a] IBM. «2022 development roadmap». (2022), [Online]. Available: https://www.ibm.com/quantum/roadmap (last visited: Jan. 12, 2023).

[IBM22b] IBM. «Learn quantum computing: A field guide». (2022), [Online]. Available: https://quantum-computing.ibm.com/composer/docs/iqx/guide#learn-quantum-computing-a-field-guide (last visited: Jan. 12, 2023).

[Juo22] E. Juodyte. «NordLynx protocol – the solution for a fast and secure VPN connection». (2022), [Online]. Available: https://nordvpn.com/blog/nordlynx-protocol-wireguard (last visited: Jan. 12, 2023).

[Kan97] E. Kani, «The number of curves of genus two with elliptic differentials.», 1997.

[KNB19] N. Kobeissi, G. Nicolas, and K. Bhargavan, «Noise explorer: Fully automated modeling and verification for arbitrary noise protocols», in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, IEEE, 2019, pp. 356–370.

[MAB+18] C. A. Melchor, N. Aragon, *et al.*, «Hamming quasi-cyclic (HQC)», *NIST PQC Round*, vol. 2, no. 4, p. 13, 2018.

[McE78] R. J. McEliece, «A public-key cryptosystem based on algebraic coding theory», *January and February 1978*, p. 114, 1978.

[MR09] D. Micciancio and O. Regev, «Lattice-based cryptography», in *Post-quantum cryptography*, Springer, 2009, pp. 147–191.

[Nat16] National Institute of Standards and Technology, «Submission requirements and evaluation criteria for the post-quantum cryptography standardization process», 2016.

[Obr22] E. Obrestad. «Openstack at NTNU». (2022), [Online]. Available: https://www.ntnu.no/wiki/display/skyhigh/Openstack+at+NTNU (last visited: Jan. 12, 2023).

[Ope23] Open Quantum Safe. «Liboqs version history- src/kem». (2023), [Online]. Available: https://github.com/open-quantum-safe/liboqs/commits/main/src/kem (last visited: Jan. 19, 2023).

[OS09] R. Overbeck and N. Sendrier, «Code-based cryptography», in *Post-quantum cryptography*, Springer, 2009, pp. 95–145.

[Pos81] J. Postel, «Internet protocol», Tech. Rep., 1981.

[Pro21] ProtonVPN. «WireGuard enables high-speed, secure VPN connections on proton VPN». (2021), [Online]. Available: https://protonvpn.com/secure-vpn/wireguard (last visited: Jan. 12, 2023).

[Res18] E. Rescorla, «The transport layer security (TLS) protocol version 1.3», Tech. Rep., 2018.

[Sho94] P. W. Shor, «Algorithms for quantum computation: Discrete logarithms and factoring», in *Proceedings 35th annual symposium on foundations of computer science*, Ieee, 1994, pp. 124–134.

[SM17] D. Stebila and M. Mosca. «Open quantum safe - software for prototyping quantum-resistant cryptography». (2017), [Online]. Available: https://openquantumsafe.org/liboqs/ (last visited: Jan. 12, 2023).

[SM22] D. Stebila and M. Mosca. «KEM performance». (2022), [Online]. Available: https://openquantumsafe.org/benchmarking/visualization/speed_kem.html (last visited: Jan. 17, 2023).

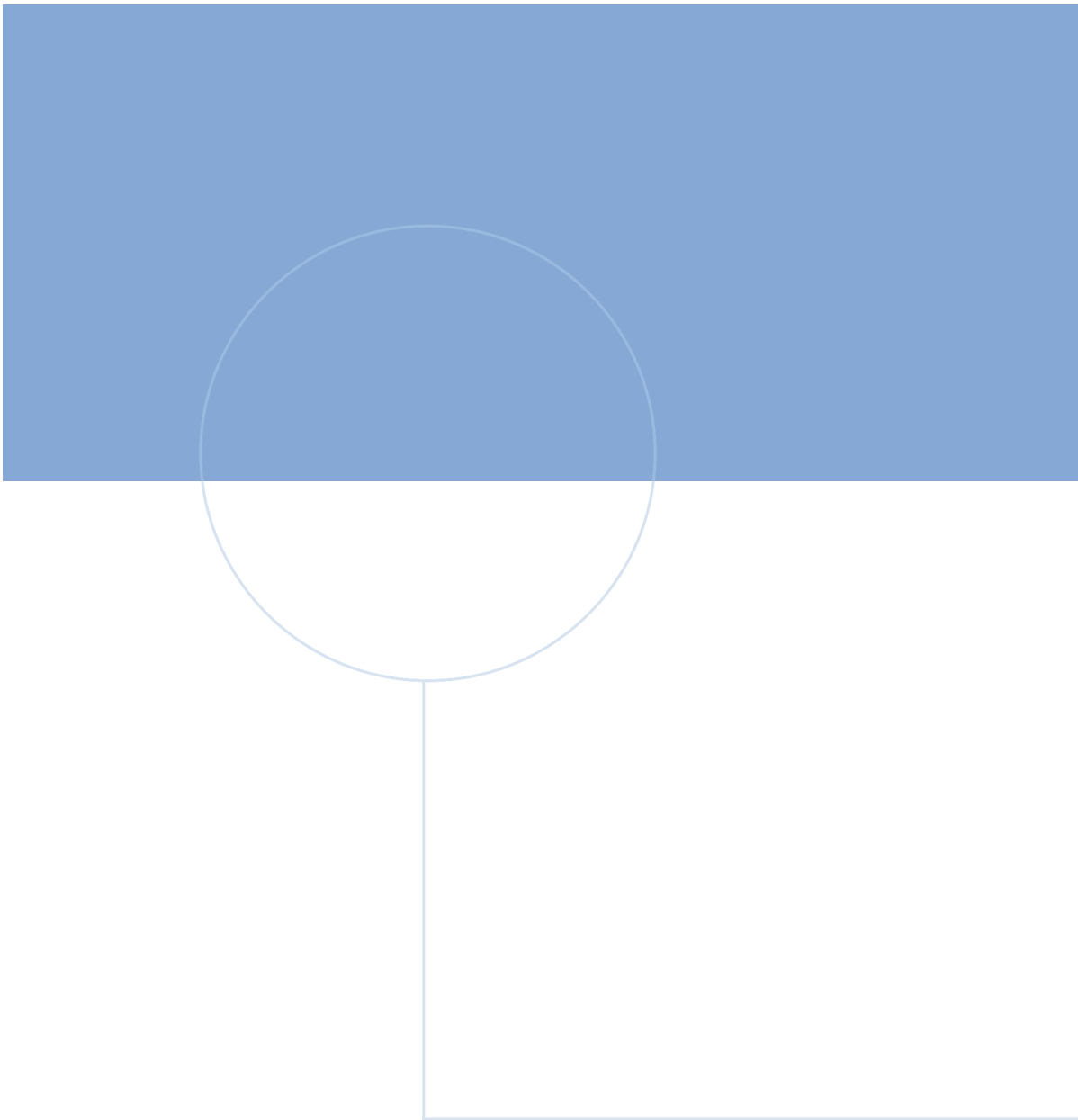
[SR20] J. Sevilla and C. J. Riedel, «Forecasting timelines of quantum computing», *arXiv preprint arXiv:2009.05045*, 2020.

[SW95] T. Sleator and H. Weinfurter, «Realizable universal quantum logic gates», *Physical Review Letters*, vol. 74, no. 20, p. 4087, 1995.

[Wol87] H. Woll, «Reductions among number theoretic problems», *Information and Computation*, vol. 72, no. 3, pp. 167–179, 1987.

[Yas21] M. Yasuda, «A survey of solving SVP algorithms and recent strategies for solving the SVP challenge», in *International Symposium on Mathematics, Quantum Theory, and Cryptography*, Springer, Singapore, 2021, pp. 189–207.

[AAC+22] G. Alagic, D. Apon, *et al.*, «Status report on the third round of the NIST post-quantum cryptography standardization process», *US Department of Commerce, NIST*, 2022.



NTNU

Norwegian University of
Science and Technology