Effects of PQC on Wireguard protocol

Qiguang Wang, Degree

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science (Future Network System)

Supervisor: Stephen Farrell

August 2023

Effects of PQC on Wireguard protocol

Qiguang Wang, Master of Science in Computer Science

University of Dublin, Trinity College, 2023

Supervisor: Stephen Farrell

This paper introduces post-quantum WireGuard, a post-quantum instantiation of the WireGuard protocol. The study is based on the KEMTLS protocol (ACM CCS 2020), which uses key-encapsulation mechanisms (KEMs) instead of post quantum signatures to authenticate communication parties. It brings both the TLS and the traditional post-quantum TLS into a more efficient protocol. Notably, this variant does not merely put focus on few aspects of security such as authentication and forward secrecy, as commonly pursued by earlier research at designing post-quantum protocols. Instead, it also offer bilateral post-quantum authentication with moderate cost of computation and communication. To accomplish this, we substitute the existing Elliptic Curve Diffie-Hellman key exchange with a new asymmetric crypto primitive, namely KEM. This thesis explores various combinations of different KEMs to alter existing WireGuard implementations. We propose three incremental modifications, each providing different level of security against quantum threats.

Acknowledgments

Thank you Mum & Dad.

Qiguang Wang

University of Dublin, Trinity College
August 2023

Contents

Abstract **i**

Acknowledgments **ii**

Chapter 1 Introduction **1**
1.1 Motivation . . . 2
1.1.1 WireGuard . . . 2
1.1.2 KEMTLS . . . 3
1.1.3 Post-Quantum Cryptography . . . 3
1.2 Structure & Contents . . . 4

Chapter 2 State of the Art **5**
2.1 Background . . . 5
2.2 Closely-Related Work . . . 5
2.2.1 Aspect #1 . . . 6
2.2.2 Aspect #2 . . . 6
2.3 Summary . . . 6

Chapter 3 Design **7**
3.1 Problem Formulation . . . 7
3.1.1 Identified Challenges . . . 7
3.1.2 Proposed Work . . . 7
3.2 Overview of the Design . . . 8
3.3 Summary . . . 8

Chapter 4 Implementation **9**
4.1 Overview of the Solution . . . 9
4.2 Component One . . . 9
4.3 Summary . . . 10

Chapter 5 Evaluation **11**
5.1 Message Sizes 11
5.2 Individual runtime 12
5.3 Experiments 13
5.4 Results 13
5.5 Summary 13

Chapter 6 Conclusions & Future Work **14**
6.1 Future Work 14

Bibliography **15**

List of Tables

2.1 Comparison of Closely-Related Projects 6

5.1 Bytes needed for the NIST-PQC level I primitives 12
5.2 Runtime for the NIST-PQC level I primitives (in millisecond) 12
5.3 Runtime for the NIST-PQC level I primitives 13

List of Figures

Chapter 1

Introduction

WireGuard is an innovative secure network protocol that functions as a virtual network interface within the Linux kernel in most cases. Its primary goal is to serve as a drop-in replacement for similar secure network protocol including IPsec and OpenVPN, providing enhanced security, better performance and improved friendliness. The secure tunnel is established by binding the peer's public key with its corresponding source address. The design of WireGuard is directly deriving from the Noise framework, bearing in mind the problem of pervasive surveillance and active tampering. Also, to get over the bandwidth and computational cost , WireGuard streamline the handshake process, utilizing a single round trip to achieve security of both perfect forward secrecy (PFS) and key-compromise impersonation (K-CI) resistance[]. But, all these security properties only holds in the classical computer model. With the arrival of the quantum computers, it is necessary to reimage of WireGuard Diffie-Hellman (DH) style handshake to use key encapsulation mechanisms (KEMs) for authentication, rather than pre-shared static key as a transitional method. **Migrating to post-quantum WireGuard** To achieve higher security levels against quantum adversaries without losing the same security level against the classical adversaries, the original WireGuard protocol needs to be revised. The main focus of this research is on amending the handshake part of the WireGuard. The primary goal of WireGuard is to establish a secure channel between communicating peers. Specifically, a secure channel should guarantee at least the following properties - Authentication: If the the other entity is authenticated, the initiator of authentication request *should only talk to the particularly identified one he think he is talking to* - Secrecy: After a successful handshake, only the communication parties are able to derive the session key or shared secret. Meanwhile, the data sent over the channel should be accessible by the communication parties. - Integrity: Both the entities can be aware of any unexpected modification to the data sent over the channel made by third parties. Since the integrity and secrecy is

assured by the combination of public key encryption and secret key encryption, and the session key derived by public key cryptography is the most important component of the subsequent secret key encryption, we should put most effort into how we apply the public key encryption for authentication. Typically, there are two categories of authentications: - Implicit authentication: In most cases, key authentication is referred to as "implicit key authentication" because it does not involve explicit validation of the key itself, but rather relies on the security properties of the underlying cryptographic primitives and the secrecy of the key to ensure its authenticity. It is a property of key agreement protocol that only the other intended entity may be able to get the session key. This property can be either unilateral (authenticated for one entity) or mutual (authenticated for both entities). - To explain the difference between implicit authentication and the other one, the notion "key confirmation" will be introduced. It's a similar property to implicit authentication. It means one entity can be sure that some other entities do get hold of the session key. It does not require "only identified entities" to have the key, it is just rather a guarantee of "having the key". - Explicit authentication: It requires both implicit authentication and key confirmation properties. It means the entity being explicitly authenticated not only is the intended one who is able to acquire the session key but also actually have the key. In the WireGuard scenario, it does not explicitly authenticate all the communication data which restricts its security in regard to eCK-PFS model[]. Therefore , we proposes a more efficient and safer way to leverage the KEM to achieve mutual explicit authentication, in a similar manner to the kemtls and kemtlspdk and achieve both post-quantum authentication and eCK-PFS. The research also instantiates with a variety of post-quantum secure KEMs based on fundamentally different mathematical assumptions.

1.1 Motivation

1.1.1 WireGuard

WireGuard provides a very simple and novel interface for setting up secure channels. All the cryptography primitives are cutting edge: Curve25519, ChaCha20, and Poly1305 and the complexity, as well as the sheer amount of code is ignorable. Unlike TLS, which offers backward compatibility and cryptography agility. The WireGuard hard-codes all the cryptography primitive choices. It's one of the reasons why WireGuard has gained a reputation for performance and reliability. In the meantime, these primitives provide unparalleled degree of performance in the first place, introduce fewer dependencies and less complexity to manage which will make it easier to ensure that the new design is secure. Furthermore, Its simplicity and minimalist design philosophy aligns well with the current

state of post-quantum cryptography. Many post-quantum cryptographic algorithms are still being tested and optimized, and are often more computationally intensive or require larger key sizes than their classical counterparts. Given the diverse range of performance trade-offs of post-quantum algorithms, exploring alternatives to the TLS-like ECDH combined with signature seems a sound way of improvement. Such as the KEMTLS[] protocol and its state-of-art variant KEMTLS-PDK[] which is the abbreviation for KEMTLS with pre-distributed keys. Both replace the ECDH and signature with KEMs (Strunk and White (1999)).

1.1.2 KEMTLS

1.1.3 Post-Quantum Cryptography

In recent years, there has been a considerable amount of research on quantum computers which can solve specific mathematical problems that are difficult for classical computers by exploiting quantum phenomena. In the foreseeable future, the large-scale quantum computer probably will bring unprecedented speed and power in computing to reality. However, this quantum edge also poses serious problems which now is so-called quantum apocalypse - "store now, decrypt later" within the field of post-quantum cryptography. One of such threats lies in the domain of public key cryptography. When it comes time, the quantum computer could potentially break many of the cryptography systems, which were designed and secured under the premises of classical computing. It's crucial that we adopt a proactive approach and remain one step ahead of the potential risks. We can not afford to idle until the quantum computers begin dismantling today's public-key cryptography is running. Therefore, the goal of this project is to scrutinize the vulnerable part of the WireGuard and propose a new WireGuard which is post-quantum safe while still keeps the virtue of original WireGuard protocol. The necessity for quantum safe cryptography stems from the capability of Shor's algorithm's ability to efficiently resolve problems related to factorization and discrete logarithm which are the security backbones of **RSA**, **ECC** and **ECDSA** that rely on the **IFP** (integer factorization problem), the **DLP** (discrete logarithms problem) and the **ECDLP** (elliptic-curve discrete logarithm problem) respectively. By using Shor's algorithm, a k-bit (in binary) number can be factored in time of order $O(k^3)$ using a quantum computer of 5k+1 qubits. As the NIST standardization process is ongoing [], this standardization doesn't assert the supremacy of one suggestion over another. Given that the NIST standardization process has already chosen the **Selected Algorithms 2022** [], the most secure transition method will be integrate the standardized Post-Quantum Cryptography algorithms instead of waiting for national bodies to finalize PQC algorithms in which case where the

risk associated with quantum broken cryptography is not acceptable.

1.2 Structure & Contents

At the end of the introduction, a layout of the structure and the contents of the following chapters should be provided for the reader. The overall goal of all descriptions of contents that follows these descriptions is to prepare the reader. The reader should not be surprised by any content that is being presented and should always know how content that is currently being read fits within an overall dissertation.

Chapter 2

State of the Art

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

From the perspective of this document forming part of your degree, this chapter should demonstrate to the reader your knowledge of the area of your dissertation project. It should present your knowledge in a coherent and detailed form. The reader should understand that you have in-depth knowledge of the area of the dissertation without being overloaded with information.

2.1 Background

A section on the background of the dissertation should provide the reader with an introduction to existing technologies and concepts that form the basis of the work presented in the dissertation.

2.2 Closely-Related Work

Work in research areas tends to address a number of specific aspects. Ideally, the discussion of published research should focus on the aspects that have been addressed by various publications - and not a discussion of the individual publications.

For example, if the topic would be a discussion of work on programming languages, the subsections of the related work could be discussions of object orientation and its realisation in various languages or the use of lambda functions by these languages.

2.2.1 Aspect #1

2.2.2 Aspect #2

2.3 Summary

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

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

Table 2.1: Caption that explains the table to the reader

Chapter 3

Design

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

3.1 Problem Formulation

This section should provide the reader with an overall description of the problem that will be addressed in the dissertation. In contrast to a generic discussion of the dissertation topic in the Introduction chapter, this section should provide a detailed discussion of the problem that has been identified based on the existing work that has been discussed in the preceeding chapter.

In some dissertations, it may make sense to convert this section into a short chapter of its own which follows the discussion of the existing work and preceeds the discussion of the work of the dissertation.

3.1.1 Identified Challenges

This section should present a short description of the gaps in the existing work and the relationship of these gaps to the work described in this dissertation.

3.1.2 Proposed Work

This section should provide a thorough description of the problem and an overview of the work proposed to address the problem.

3.2 Overview of the Design

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

3.3 Summary

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

Chapter 4

Implementation

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

4.1 Overview of the Solution

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

Listing 4.1: Lengthy caption explaining the code to the reader

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

4.2 Component One

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

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

Listing 4.2: Second Lengthy caption

4.3 Summary

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

Chapter 5

Evaluation

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

5.1 Message Sizes

Adding more key exchange primitives to the handshake protocol obviously adds additional data, in the form of public keys and encapsulated values. This data needs to be transmitted in the initiation or response message. With PQ KEMs that data is relatively large for network transmissions, hundreds of Bytes up to hundreds of Kilobytes in the worst cases. Thus, packets may easily become larger than the Ethernet maximum transmission unit (MTU) of 1500 Bytes. Since WireGuard is based on UDP, there is no segmentation functionality on the Transport Layer. If large data is sent over UDP, packet fragmentation may happen on the IP layer. Relying on this functionality is considered fragile, and the IETF specifically advises against it. [8] This is because the IP standard allows routers to drop large packets silently. A way around IP fragmentation is to split the datagrams on the application layer. When using UDP, this can still be terrible for performance because the handshake will fail anytime one of the datagrams is lost. The more datagrams we need for one message, the higher the probability for losing one of them is. Thus, breaking up messages into more datagrams makes the handshake take longer on average. A datagram size of 1436 Bytes is reasonable for this, because according to analysis by Shannon and Moore [31] around 98% of MTUs are between 1484 1500 Bytes. Subtracting 8 Bytes

for the UDP header and 40 Bytes for the possibility of an IPv6 header, we arrive at the number above. Also, the IPv6 standard recommends an MTU of at least 1500 Bytes, and requires acceptance of packets up to that same size. Choosing cryptographic primitives with small key sizes is thus essential to prevent unnecessary overhead, in the form of excessive transmission times. Cryptographic primitives with excessively large public keys or encapsulated values have thus been excluded from all following considerations. For example, all code-based cryptosystems, which need many Kilobytes for a public key.

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

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

5.2 Individual runtime

Runtime	Key Generation	Encapsulation	Decapsulation
mceliece348864f	51.88	0.35	19.70
mceliece460896f	166.49	0.66	43.96
kyber512	0.02	0.01	0.01
kyber768	0.03	0.02	0.02
BIKE-L1	0.13	0.03	0.5
HQC-128	0.03	0.05	0.09

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

Protocol	Initiation size (bytes)	Response size (bytes)	Time in microseconds		
			Server	Client	Total
WireGuard (original)	148	92	0.36	0.47	1.37
KEM-WireGuard (kyber512 only/time-optimal)	1700	1596	0.21	0.34	0.57
KEM-WireGuard (mcliece348864f only)	261348	252	20.82	91.47	133.36
KEM-WireGuard (BIKE-L1)	3246	3206	1.46	3.65	4.60
KEM-WireGuard (HQC-128)	6862	9022	0.49	0.61	0.84
KEM-WireGuard (M&K/size-optimal)	1028	252	20.40	20.95	41.47

Table 5.3: Runtime for the NIST-PQC level I primitives

5.3 Experiments

5.4 Results

5.5 Summary

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

Chapter 6

Conclusions & Future Work

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

6.1 Future Work

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

Bibliography

Strunk, Jr., W. and White, E. B. (1999). *The Elements of Style*. Pearson Education, 4th edition.