BERGISCHE UNIVERSITÄT WUPPERTAL

Bachelor Thesis

Comparing Post-Quantum Instantiations of the TLS1.3 Handshake

Jonas Dinspel

20.02.2026

Chair for IT Security and Cryptography
University of Wuppertal

First Examiner:	**Prof. Dr.-Ing. Tibor Jager**
Second Examiner:	**Jun.-Prof. Dr.-Ing Malte Mues**

Use of AI

By submitting this thesis, I declare that I have carefully read and followed the university's guidelines and the instructions of the ITSC research group on the use of AI.

Eidesstattliche Versicherung

Affirmation in lieu of an oath

______________________________ ______________________________
Name, Vorname/ surname, first name Matrikelnummer/ student ID number

☐ Bachelorarbeit/ bachelor‘s thesis ☐ Masterarbeit (auch Staatsexamen)/ master‘s thesis

mit dem Titel/ title

____
____
____
____

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Abschlussarbeit mit dem oben genannten Titel selbstständig und ohne unzulässige fremde Hilfe (insbes. akademisches Ghostwriting) erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt; dies umfasst insbesondere auch KI-Systeme, Software und Dienste zur Sprach-, Text- und Medienproduktion. Ich erkläre, dass für den Fall, dass die Arbeit in unterschiedlichen Formen eingereicht wird (z. B. elektronisch, gedruckt, geplottet, auf einem Datenträger) alle eingereichten Versionen vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Die Strafbarkeit einer falschen eidesstattlichen Versicherung ist mir bekannt, namentlich die Strafandrohung gemäß § 156 Strafgesetzbuch (StGB) bis zu drei Jahren Freiheitsstrafe oder Geldstrafe bei vorsätzlicher Begehung der Tat bzw. gemäß § 161 Abs. 1 StGB bis zu einem Jahr Freiheitsstrafe oder Geldstrafe bei fahrlässiger Begehung.

I hereby affirm in lieu of an oath that I have completed this thesis with the above title independently and without unauthorized assistance from third parties (in particular academic ghostwriting). I have not used any other sources or aids than those indicated; this includes in particular AI-systems, software and services for language, text, and media production. In the event that the work is submitted in different formats (e.g. electronically, printed, plotted, on a data carrier), I declare that all the submitted versions are fully identical. I have not previously submitted this work, either in the same or a similar form to any examining authority.

I am aware of the criminal liability of a false Affirmation in lieu of an oath, namely the threat of punishment according to § 156 StGB up to three years imprisonment or fine for intentional committal of the offence or according to § 161 Abs. 1 StGB up to one year imprisonment or fine if committed by negliance.*

**Please be aware that solely the German version of the Affirmation in lieu of an oath ("Eidesstattliche Versicherung") is the official and legally binding version.*

______________________________ ______________________________
Ort, Datum/ place, date Unterschrift/ signature

Belehrung
Official notification

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer von einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 156 StGB (German Criminal Code): False declaration in lieu of oath

Whoever falsely makes a declaration in lieu of an oath before an authority which is competent to administer such declarations or falsely testifies whilst referring to such a declaration incurs a penalty of imprisonment for a term not exceeding three years or a fine.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

§ 161 StGB (German Criminal Code): Negligent false oath; negligent false declaration in lieu of oath

(1) Whoever commits one of the offences referred to in sections 154 to 156 by negligence incurs a penalty of imprisonment for a term not exceeding one year or a fine.

(2) No penalty is incurred if the offender corrects the false statement in time. The provisions of section 158 (2) and (3) apply accordingly.

Abstract

Some Advice. Think of the abstract as a short version of your thesis. Motivate the topic of your thesis, and give a brief summary of its contents. Keep in mind that the abstract (and the remainder of your thesis) should be comprehensible for fellow students of yours. It is often expected that abstracts do not exceed one page.

Contents

1 Introduction

Writing an Introduction. Introductions are often regarded as the "hardest part" when it comes to writing a thesis. You can use the following questions as a golden thread:

- Why is the topic of your thesis of particular interest? Why is it interesting to investigate this topic today?
- What are interesting problems and why are they interesting?
- Are there simple or naïve approaches to solve those problems? Why do they fail in practice?
- What are the goals of your thesis?
- What is the current state of the art?
- Did you contribute to the state of the art? How?
- Is there any related work not covered by the previous questions? Which? Why are those works not applicable to your thesis?
- How is your thesis structured?

Do not be afraid of writing too much. In my opinion, a good introduction is at least 3–4 pages long, sometimes even longer. For example, the introduction of my PhD thesis is 13 pages long, including a broad research motivation, several conceptual approaches to my research, the formulation of research questions, the state of the art, how my thesis advanced the state of the art, and related work. Of course, we do not expect a 13 page introduction in a bachelor or master thesis but we encourage you to invest some time when writing it. By the way, a well-written introduction is a great outline for a talk about your thesis.

2 Related Works

There already is a broad range of research regarding post-quantum TLS.

3 Preliminary

In following I will discuss the TLS 1.3 handshake and used components to give a solid understanding of mechanisms and schemes included in the proposed formula and the calculator using it .

3.1 TLS1.3

The TLS 1.3 Handshake is the backbone of todays communication via the internet, as it enables us to establish a secure channle over a unsecured medium. To achive this, client and host agree on a shared secret within one round trip, as the following figure describes:

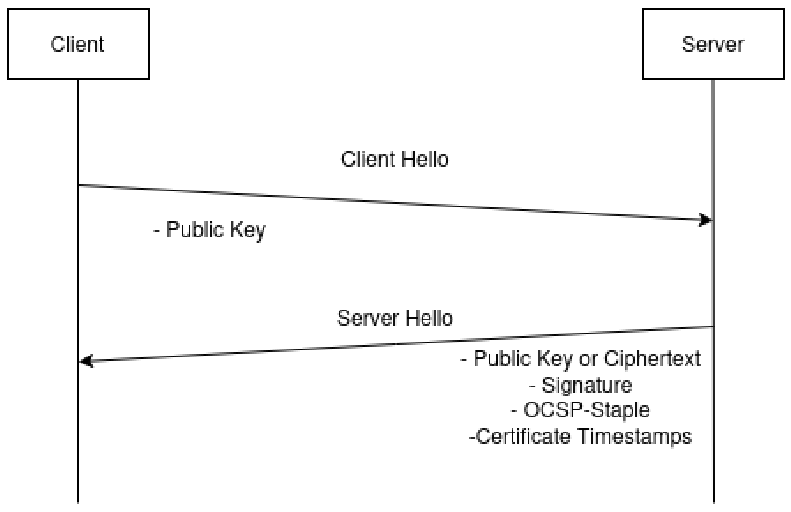

Figure 3.1: TLS 1.3 Handshake, reduced to transmitted cryptoobjects

During the two messages sent, `Client Hello` and `Server Hello`, no secret is transmitted and all necessary information is exchanged to agree on a shared secret. This shared secret is used to symmetrically encrypt any further messages sent. In real-world situations these messages contain a lot addtitonal information, but for the proposed use case the transmitted cryptographic objects as shown in the figure are fully sufficient.

3.1.1 Key-Exchange

There are three different available approaches to agree on a shared secret. Public key infrastructure(hiermit bin ich nicht wirklich zufrieden, ich weiss aber nicht wie ichs anders betiteln soll), reffered to as `PKI`, and Key-Encapsulation Mechanisms, reffered to as `KEM`. The thrid method abailable are hybrid approaches, which combine KEMs and PKI.
The key-exchange is based on assymetric cryptography, in which a uniqe pair of `public key` and `private key` is generated. A message m encrypted using the private key sk can only be decrypted using the public key pk. The way encryption and decryption work and how pk and sk are generated depend on the used scheme.

PKI

Using PKI, client and server agree on a shared secret by simply exchanging their public keys, from which the shared secret can be derivated. During the handshake both `Client Hello` and `Server Hello` will include the public key. The authenticity of the servers public key is backed by the servers certificated, which will be discussed later.

KEM

KEMs work similar to PKI with the difference that the servers public key is encapsulated in a ciphertext by the KEMs encapsulation algorithm. The `Client Hello` will include the public key, and the `Server Hello` will include the generated ciphertext, which will be decapsuated with the accourding algorithm on the clients side to recieve the servers public key. The public keys authenticity is again backed by its certificate.

Hybrid

Hybrid key exchange utilizes a combination of PKI and KEM, thus the `Client Hello` will include the public key of the used PKI scheme and the public key of the used KEM scheme. The `Server Hello` will include the public key of the used PKI and the ciphertext containing the encapsulated public key of the used KEM scheme. With both messages containing two cryptohraphic objects, these messages average size is larger than using a single approach. The advantage of hybrid approaches is increased forward security, as the newer KEMs are additionally secured by long tested PKI, which especially mitigates flaws in the implementations of KEMs that were not yet discovered.

3.1.2 Signatures

As already mentioned, signatures are used to validate the authenticity of a server. Signatures are encrypted using the servers private key and are issued by a certificate authority(CA) for a limited period of time, after which they need to be renewed. Using the servers private key to encrypt the signature ensures that the recieved public key is owned by the desired server, as only the public key of the key pair can be used to decrypt the signature. The servers signature is send within the `Server Hello` message.

3.1.3 Extensions

Three TLS extensions which impact the security of the handshake and transmit some sort of cryptographic object are availabe for created configurations.

Encrypted Client Hello

Encrypted client hello (`ECH`) encrypts, as suggested by its name, the client hello message, adding an additional layer of security to the key-exchange. With ECH enabled the public key within the `Client Hello` doubles in size.

OCSP-stapeling

With OCSP-stapeling enabled, the server attaches a signed OCSP-response to the `Server Hello` to validate its certificate. This enables the client to validate the servers certificate without sending a request to the according CA, saving an additional roundtrip.

cerificate transparency

Within the certificate transparency extensions package is a record of n past certificates of the respective server. This extension is mandatory in most modern web browsers as it prevents potential attackers from impersonating servers.

3.2 PQC

In this section we discuss threats imposed on the TLS 1.3 Handshake by post-quantum computers and schemes which are able to mitigate posed threats. Schemes which are vulnaerable to these threats will be referd to as `classic cryptography`.

3.2.1 Shors algorithm

Shors algorithm is the major threat which the Handshake is faced with. Using Shors algorithm potential attackers gain the ability to factor large numbers in polynomial time, with its complexity being exponentially more efficient than Quadratic Sieve for example. This especially poses a thread to the widely used RSA algorithm, as its security is solely based on the difficulty of factorizing large integers.

3.2.2 ML-KEM

ML-KEM, short for `Module-Lattice-Based Key-Encapsulation Mechanism` and formerly known as CRYSTALS-Kyber, is a post-quantum key-exchange algorithm which is already standardized by NIST. Within the standardized submission are three parametersets, 512,768 and 1024, ranging from NIST-Level 1 to 5[].

3.2.3 HQC

HQC, short for `Hamming Quasi-Cyclic`, is another post-quantum key-exchang algorithm which is already standardized by NIST. It is a code based Key-Encapsulation Mechanism based on the hardness of solving the Quasi-Cyclic Syndrom Decoding[]. Similar to ML-KEM its standardized with three parametersets, HQC-1, HQC-3 and HQC-5, again ranging from NIST-Levels 1 to 5.

3.2.4 Hybrid usage

In addition to purely post-quantum key-exchange there also are hybrid solutions, which combine on of the proposed post-quantum algorithms with algorithms from classic cryptography such as RSA or ECDHE.[Explanation or graphic?] This drastically reduces the attack surface outside of mathematically breaking the encryption as schemes as RSA and ECDHE are used for a such a long period of time that most if not all exploits in their implementation and appliance are already fixed.

3.3 NIST-Levels

Any post-quantum scheme standardized by NIST is classified in security strength categories, or NIST-Levels, rangeing from 1 to 5. With the uncertainties of yet to be discoverd quantum attacks and the limited ability to predict performance metrics for future quantum computers, these categories are defined by reference primitves rather than bits of security. These will serve as the base of a wide variety of metrics relevant in practical security. [Erklärung der NIST-Level von der Nist Website]

3.4 Illustrated components

Next we discuss how the proposed components will be illustrated and used with the formula which calculates the size of the handshakes cryptographic objects. The selection of at least a key-exchange scheme or signature scheme is mandatory. Usage of available extensions is optional.

Key-Exchange

Depending on the selected key-exchange method a combination of public key *pk* and ciphertext *ct* is used in the calculation, according to the following table:

	`Client Hello`	`Server Hello`
`PKI`	pk	pk
`KEM`	pk	ct
`hybrid`	$pk_{PKI} + pk_{KEM}$	$pk + ct$

Table 3.1: Cryptographic objects used from key-exchange

Signatures

For the selected signature scheme the size of the resulting signature which is part of the `Server Hello` is used in the calculation.

Extensions

Depending on the extension different elements are taken into the calculation:

`ECH` - the additional public key is taken into the calculation by doubeling the size of the public key from the selected key exchange scheme

`OCSP-Stapling` - the signature part of the attatched OCSp-response is used in the calculation, the rest of the response is not used

`Certificate transparency` - the attached certificate chain is used in the calculation, everything else is not used

4 Method

I create an UI based calculater to configure and compare the size of transmitted cryptographic objects of up to 2 different instantiations of the TLS 1.3 handshake. These instantiations consist of key-exchange and used signature schemes as well as different TLS extensions. Available extensions are `OCSP-Stapeling`, `certificate transparency` and `encrypted client hello`. The underlying datasets for key-exchange and signing include different pre- and post-quantum schemes with different parametersets available for each scheme.

classic	post-quantum
DHE	HQC
ECDHE	KYBER

Table 4.1: Aviable key-exchange schemes

For signature schemes, there is a broad spectrum of different post-quantum schemes with different NIST-Status, including on-ramp and not fully proven as secure applications. As this calculator focuses on post-quantum instantiations of the TLS 1.3 handshake, all included schemes from classic cryptography are those which are included in [rfc8446], where the TLS 1.3 handshake is formally defined. Legacy algorithms, even those annoted in [rfc8446], are not included.

4.1 Limitations

The formula which is used by the calculator only includes the size of cryptographic objects during the handshake, stopping at and already excluding the shared private key. Everything aside the cryptographic objects in each payload is not taken into consideration. This includes package information, additional extensions and even headers, even these used in OCSP or Certificate Transparency. The computational effort of used schemes is not taken into consideration either, as results heavily vary outside of benchmark environments. By excluding these factors I ensure the compareability and consitancy of generated results, regardless of connected host or computing machine in real-life scenarios.

schene	status	scheme	status
EdDSA	classic	CROSS	On-ramp
RSA	classic	Feast	On-ramp
DHE	classic	Falcon	t.b.s
UOV	On-ramp	Hawk	On-ramp
SQIsign	On-ramp	Less	On-ramp
SNOVA	On-ramp	MAYO	On-ramp
SLH-DSA	FIPS	ML-DSA	FIPS
SDitH	On-ramp	MQOM	On-ramp
RYDE	On-ramp	Mirath	On-ramp
QR-UOV	On-ramp	PERK	On-ramp

Table 4.2: Aviable signature schemes

4.2 Capabilities

This calculater can be used to quickly compare the size of transmitted cryptographic objects during `client` and `server hello` as well as the total size, without setting up and reconfiguring a dedicated server. These objects are included:

- The used public key, which can also be encrypted if the extension Encrypted Client Hello [] is enabled
- Transmitted ciphertext, which will be used if the key exchange is handled by a KEM[]
- Signatures
- The signature of OCSP-responses, if OCSP-Stapeling is enabled
- the signature of scts, if Certificate Transparency is enabled

4.3 data source

The data used for calculating the size of the key-exchange is sourced from their indidvidual NIST-publications[][][][]. Each signature dataset is sourced from the repository of the "PQ Signatures Zoo" open source project []. By using consistent sources for each dataset I further ensure the compareability of generated results.

4.4 Calculator

The calculator gives you the ability to compare the size of cryptographic objects exchanged within the TLS 1.3 handshake. This includes the key exchange, signatures and OCSP-stapling, Encrypted Client Hello, and Certificate Transparency. The key exchange offers a wide variety of schemes from classical and post-quantum cryptography, as well as hybrid key exchange, which can be freely configured from offered PKI and KEM schemes. Available signature schemes are sourced from the PQ-Zoo; project's GitHub page.

4.4.1 Underlying Equation

The total size is calculated using the following equation:

$$y_1 * a * pk_{client1} + y_2 * a * pk_{client2} + y_3 * pk_{server} + y_4 * ct + y_5 * \sigma + y_6 * \sigma_{ocsp} + y_7 * b * \sigma_{ct} \tag{4.1}$$

$$\begin{aligned} \text{s.t.} \quad pk_{client1}, pk_{client2}, pk_{server}, ct, \sigma, \sigma_{ocsp}, \sigma_{ct} &\in \mathbb{N} && (4.2)\\ y_i \in \{0,1\}\,,\ \forall i &\in \{1,2,3,4,5,6\} && (4.3)\\ a &\in \{1,2\} && (4.4)\\ b &\in \mathbb{N} && (4.5)\\ y_1 + y_4 &\geq 1 && (4.6) \end{aligned}$$

The formula and subjected restrictions are to be understood as follows:

(4.1) calculate total size, considering all aspects of the represented instatiation

(4.2) represents the size of corresponding cryptographic object in bytes. Needs to be a positiv whole number

$pk_{client1}$	Client Public Key
$pk_{client1}$	Second Client Public Key - for hybrid key-exchange
pk_{server}	Server Public Key
ct	Server Ciphertextunderline
σ	Signature
σ_{ocsp}	OCSP-response signature
σ_{ct}	Certificate Transparency signature

(4.3) Represents if component is selected or not.

(4.4) Factor for Client public key. If Encrypted Client Hello is enabled public key size is doubled. Impacts both public keys if hybrid key-exchange is enabled

(4.5) Factor for Certificate Transparency, represents log length.

(4.6) At least key-exchange or signature need to be included

4.4.2 UI

This formula is embedded in the shown in fig x browser based user interface, enabeling users to configure different instatations within the subjected restrictions proposed earlier. The UI shifts selection options based on made inputs, so instantations outside of the given constraints can not be created.

Creating Configurations

1 Select how the key-exchange is handled. All available schemes are within the upper dropdown. Once a scheme is selected, the desired parameters can be selected from the second dropdown. If hybrid key exchange is enabled, there will be two sets of dropdowns instead. The upper set defines the used KEM, the lower set defines the used PKI.

2 Select the signature algorithm. The upper dropdown again includes a list of available schemes, the lower one the possible parameters.

3 Additional extensions can be toggled on and off. If Certificate Transparency is enabled, the length of its backlog can be set

4 By clicking on "show config", the created config will be added to the list of comparable configurations.

Viewing Configurations

Each created config will be added to the list on the ride hand side of the calculator. It is identified by the schemes used for key-exchange and creating the signature. The total size of transmitted crypto objects will also be visible. Configs can be expanded by clicking on them, revealing details of the selected components. These include:

1 The selected key-exchange scheme(s) name, which is a link to some external side with additional information about each scheme, the selected parameters and the corresponding NIST-Level

2 The selected signature scheme's name, which is a link to some external website with additional information about each scheme, the selected parameters and the corresponding NIST-Level

3 Client Hello with the size and type of each transmitted crypto object as well as the total size of all crypto objects within the client hello message.

4 Server Hello with the size and type of each transmitted crypto object as well as the total size of all crypto objects within the server hello message.

5 The combined size of all transmitted crypto objects during the handshake

5 Results

5.1 benchmarking

Table of max and min size for key-exchange and signature, as well as total handshake size and possible nist levels (1,3 and 5 as 2 and 4 are not within my source data) - script is already done, results just need to be written down here

nist level	ke scheme	size	sig scheme	size
pre-quantum	ke scheme	size	sig scheme	size
1				
3				
5				

Table 5.1: Schemes with largest cryptographic objects per level

nist level	ke scheme	size	sig scheme	size
pre-quantum	ke scheme	size	sig scheme	size
1				
3				
5				

Table 5.2: Schemes with smallest cryptographic objects per level

5.2 implications for traffic

regular mtu is around 1500 bytes for ethernet traffic. TCP technically supports up to 65535 bytes, but this value is more theoretical than practical. -> fragmentation is to be considered at certain key or sign sizes -> implications on security and performance ->calculator only shows size of crypto objects, full handshake messages are even larger!

5.3 Use cases

Fast comparision of different configurations with insights ons security and size influence of addons on package size. Does not replace testing within fully configured server setups, but gives a quick comparison of viable configurations!

6 Conclusion

6.1 Implications

6.1.1 Traffic

6.1.2 Packages

7 Conclusion