Master Thesis

Taking the quantum leap: Preparing DNSSEC for Post Quantum Cryptography

Author
G.J. Beernink

Supervisors
M. Müller — University of Twente & SIDN
R. van Rijswijk-Deij — University of Twente & NLnet Labs

February 11, 2022

UNIVERSITY OF TWENTE.

Acknowledgement

This thesis marks the end of my studies at the University of Twente. Already during an earlier research under his guidance, Prof. Roland van Rijswijk-Deij sparked my interest in the Domain Name System. I am grateful he offered me a position afterwards to finish my master's degree in the field of DNS combined with applied cryptography. It turned out this combination of topics suited my interest well. I want to thank him for helping me aim this research and formulate the research questions. Although we did not have many meetings and the ones that we did have were all online, I could always feel his enthusiasm for the DNS ecosystem. Although his extensive, handwritten feedback was not always completely readable, it was definitely appreciated.

My day-to-day supervisor, or should I say two-week-to-two-week supervisor, started with PhD student Müller but my thesis was finished with supervisor Dr. Müller. Thank you, Moritz, for having the weekly, and later biweekly meetings during both my Research Topics and Final Project. In our meetings we understood each other in our discussions and I could always expect a good portion of positively formulated feedback on all draft versions. Additionally, I would like to thank Assistant Professor Florian Hahn for taking place in my graduation committee and the UT LISA for the provided data captures used in this thesis.

Besides the academic guidance, I would also like to thank my friends and housemates for the wanted distraction in a time where not much social activities could be planned. Luckily I could always look forward to the online coffee-dates with my friends or spend time with some housemates when I didn't feel like writing this thesis. I would like to finish this acknowledgement by thanking my girlfriend and family for supporting me during not only my thesis, but during my whole studies.

Abstract

Once a quantum computer is capable of running a cryptanalytic attack against currently used public key cryptographic algorithms, DNSSEC no longer provides DNS with the security that this core Internet Protocol needs. Attacks such as spoofing DNS responses that would lead users to a malicious website are then again possible for adversaries having a functioning quantum computer. While current quantum computers only have a few quantum bits and can not yet perform difficult computations to break modern public key cryptography, progress is made fast and solutions must be designed to keep the Internet a safe and secure place. This is especially important since it is expected that the transition of DNSSEC to more secure signing algorithms will take years. Post-Quantum Cryptography (PQC) must ensure the security of Internet Protocols in the future where quantum computers are available. These PQC algorithms are researched and evaluated by the cryptographic community together with NIST in a competition-like standardization process for quantum-resistant cryptographic algorithms. In this study DNS is researched for its ability to adopt PQC algorithms in its security extension DNSSEC. Limitations exist on several aspects of the DNS protocol that have direct consequences to the adoption of PQC algorithms into DNSSEC. As DNS packets have a limited size, the signatures and public keys that DNSSEC communicates must stay below this size limit. Additionally, PQC algorithms must not cause DNS operators such as resolvers and name servers to experience an extremely high computational overhead on respectively signature verification and zone signing.

This study uses real-world traffic traces from a recursive resolver deployed on a university campus to analyse the impact of PQC algorithms on DNS response sizes. Using signer logs from the Dutch ccTLD `.nl`, consequences of PQC algorithms on DNS name servers that need to periodically sign the zone file are studied. The impact of PQC algorithms on DNS resolvers are studied as well regarding the computational load of verifying PQC signatures. To work around limitations imposed by the DNS maximum response size and to increase the number of algorithms that can be adopted into DNSSEC, an *out-of-band* key exchange is designed and implemented in a resolver. Different versions of the HTTP protocol and the FTP protocol are benchmarked for their performance on the communication of large public keys outside DNS packets using the implemented out-of-band key exchange method. Security considerations regarding out-of-band key exchange and the benchmarked transport protocols are discussed together with the consequences for the already centralizing nature of the Internet.

Based on the results from this study, only the Falcon-512 PQC alternative can be successfully adopted into DNSSEC without resulting in packets growing over the DNS size limit and without significantly increasing the computational load on DNS name servers and resolvers. However, this comes at a cost of an increase in TCP traffic. In this study, a promising approach to enable more PQC algorithms to be adopted into DNSSEC is proposed. Using this approach of out-of-band key exchange, computationally more efficient algorithms such as Rainbow can be implemented at acceptable expense. Post quantum cryptography is fairly new and the standardization process is not yet finished, additionally, algorithms such as Falcon are based on an underlying mechanism that the community is unsure about. Hence, steps must be taken to keep DNS useful in a world where quantum computers are connected to the Internet.

Contents

1 Introduction

Looking at the current progress[1][2][3] that is being made with creating quantum computers, there is a real possibility that a practical quantum computer can be used within the current or beginning of the next decade [4]. These quantum computers can solve some computationally difficult tasks faster than current computers. These tasks include mathematically hard problems such as integer factorization and discrete logarithm problems that are used as cryptographic primitives in signing algorithms. These signing algorithms are used in the Domain Name System (DNS) and its Security Extensions (DNSSEC). If these mathematical problems can be solved by quantum computers, DNS(SEC) is no longer secure and attacks based on spoofing DNS responses that were once done on the DNS infrastructure are again possible (see also Section 2.2). The need to implement cryptographically secure algorithms in DNSSEC is high. The National Institute of Standards and Technology (NIST) has started a competition for the next generation algorithms that are secure against attacks from quantum computers. This set of algorithms is referred to as Post Quantum Cryptography (PQC) and is meant to provide security in a reality where quantum computers exist. However, it is important to research whether DNSSEC can handle these algorithms and can still be useful. In this study, communicational and computational effects that these new PQC algorithms have on DNS traffic and DNS operators such as name servers and resolvers are researched. It is determined in this study that not all PQC finalists currently in the NIST competition can be adopted into DNSSEC without resulting in a decrease in functionality of the DNS service. A promising approach that uses an out-of-band key exchange is implemented in a resolver and it is shown that this can increase the number of PQC algorithms that can be used in DNSSEC. This also includes algorithms that are computationally more efficient than currently used algorithms.

1.1 Thesis outline

In the next section, background information relevant to this study is given. This includes an explanation of how and why DNS works and through what mechanisms security is provided in the DNS infrastructure. Additionally, an explanation of what Post Quantum Cryptography algorithms are and how these are currently being standardized is given. Afterwards, in Section 3, research that relates to this study is elaborated on, including research on the adoption of new algorithms in the DNS infrastructure as well as research of PQC in other Internet Protocols. In Section 4 the problems that arise are defined and research questions are formulated on how to study these problems. In Section 5 these research questions are elaborated on and a methodology is described on how these can be addressed. These research questions are studied and results are given in the following sections, where Section 6 focuses on communicational overhead, Section 7 on computational overhead and Section 8 on a solution for the problems that are found. Finally, a discussion and conclusion, together with work that remains to be done and is a lead for future studies can be found in Sections 9 and 10 respectively.

2 Background

A detailed description of how DNS works is given in Subsection 2.1, explaining the different Resource Records and an example DNS IP-resolution for 'www.example.com'. It also describes what threats DNS faces and why an additional layer of security is necessary. Subsection 2.2 describes how the DNS Security Extensions (DNSSEC) works and how DNSSEC handles the security vulnerabilities of plain DNS. Additionally an elaboration on how DNS achieves message integrity using the *chain of trust* is elaborated on as well as how DNSSEC is related to this study. In Subsection 2.3 limitations and weaknesses of the current implementation of DNSSEC are discussed. At last, in Subsection 2.4, Post-Quantum Cryptography (PQC) is described together with the ongoing standardization process of these.

2.1 DNS

The main responsibility of the Domain Name System (DNS) is resolving domain names or hostnames that humans understand ('www.example.com') to the IP-addresses that computers understand ('`93.184.216.34`'). It provides us with an easy way of accessing the Internet without having to remember the IP-addresses of servers ourselves and can be considered the backbone of the Internet. In this section a more detailed view of the different components of the DNS and how these work together to deliver the service is given.

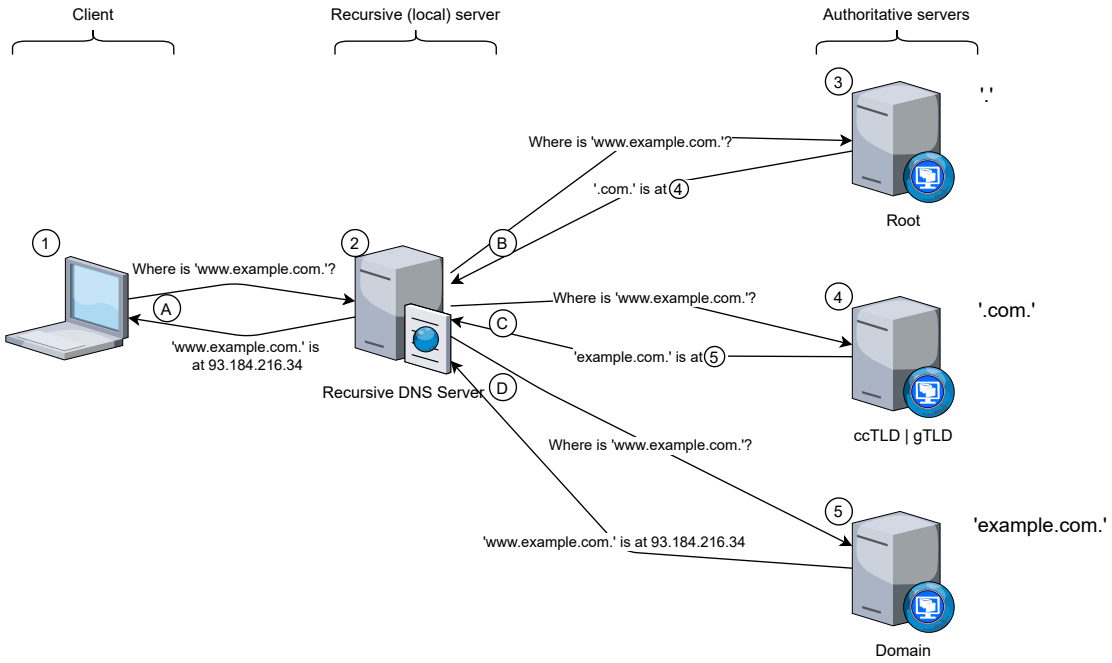


Figure 1: The process of resolving a domain name.

Name Servers and Resource Records
Resolving a domain name to its IP-address generally requires multiple queries to different DNS Name Servers (NS). A name server responds to a query with a Resource Record (RR), this can be a RR with the resolved domain name in a `A` or `AAAA` record or with a `NS` record with a referral to another name server that can further resolve the domain name. These and other DNS RR types are defined in RFC 1035, see [5]. Common RR types that are relevant for this study are listed in Table 1.

To efficiently find the IP-address of a domain name, each part of a domain name is separated by dots, referred to as labels. The domain name 'www.example.com.' has 4 labels, namely 'www', 'example', 'com' and the empty label, called the `root` label, visualized with a period in Figure 1. These labels allow for an efficient resolving strategy as will be explained alongside Figure 1 where an example resolution of 'www.example.com' to its IP address is depicted. In this Figure, machines are illustrated with the numbers (1) to (5) and communication

between these machines with (A) to (D).

The machines and the communication flow between them is as follows:

(1) A **Client** wants to visit 'www.example.com.'. The Client queries a Recursive Resolver in its network to find the corresponding IP-address to this hostname over communication channel (A), the response will follow later in this example run once the resolver has the answer to the query. The Recursive Resolver could be a machine on the same network as the Client, for example a DNS Resolver in a large corporate network. It could also be the Recursive Resolver of the Internet Service Provider itself.

(2) A **Recursive Resolver** performs the domain resolving on behalf of the client. If the resolver has seen the same query before and has cached the result, this can be returned immediately, skipping all further steps. Otherwise, the resolver performs a series of requests to resolve the hostname for the client, starting by querying the server for the `root` label, the Root Server.

(3) The **Root server** is the first of the Authoritative servers that are queried for the authoritative servers of all Top Level Domains (TLD), including Country Code TLDs (ccTLD) and generic TLDs (gTLD). These root servers have a zone file, containing a mapping of all TLDs to the IP-addresses of the servers that can help the Recursive server with the next part of the domain name or hostname. The root servers are responsible, or authoritative, for their `NS` records and do not have `A` or `AAAA` records. Root name servers only refer resolvers to other name servers and do not provide IP addresses of domain names themselves. Our query is answered with a referral to the `.com` authoritative NS. The `root` NS responds with all authoritative name servers for `.com`. There are 13 root servers for `.com`, sorted alphabetically below. The different components of such a response are discussed in Section 2.2.

Response (B):

```
com.                    172800  IN  NS  a.gtld-servers.net.
com.                    172800  IN  NS  b.gtld-servers.net.
com.                    172800  IN  NS  c.gtld-servers.net.
                        ...
com.                    172800  IN  NS  m.gtld-servers.net.
```

(4) **TLD servers**, including ccTLDs and gTLDs, are operated by registries. Most ccTLDs are operated by a manager company in the country itself, gTLDs can be managed by any business [1]. The TLD authoritative servers have yet another zone file with a mapping with more detailed information. In this case it responds to the Recursive DNS server with a list of all name servers that are authoritative for 'example.com.'. What follows as response is a referral to two servers that are authoritative for `example.com`.

Response (C):

```
example.com.            172800  IN  NS  a.iana-servers.net.
example.com.            172800  IN  NS  b.iana-servers.net.
```

(5) Finally the resolver can query the **name server** that is authoritative for 'example.com.' and again queries for 'www.example.com.'. The server is responsible for managing this domain and responds with the proper IP-address `93.184.216.34`. The responses until now were all referrals to other name servers, what follows as response from the `example.com` authoritative server is the following.

Response (D):

```
www.example.com.        86400  IN  A  93.184.216.34
```

The `A` record of this response indicates the IP address of the server. This completes the DNS query resolving for this domain and the resolver now sends the response over to the client.

Response (A):

```
wwww.example.com.       86400  IN  A  93.184.216.34
```

[1] See `https://www.iana.org/domains/root/db` for a list of TLD Managers

RR Type	Description
A	The 32 bit IPv4 address.
AAAA	The 128 bit IPv6 address.
ANY	Query to retrieve all information that is available.
MX	The mail exchanger record.
NS	The authoritative name server.
SOA	The Start of Authority record holds administrative information.
DS	Delegation Signer, identifies one or more keys that the child uses to sign RRs.
DNSKEY	The DNS public key of a name server.
NSEC/NSEC3	Next Secure (3), used to provide authenticated denial of existence of a DNS name.
NSEC3PARAM	Parameters used to calculate a authenticated denial of existence for the NSEC3 type.
RRSet	A set of RR that has the same label, class and type but different data.
RRSIG	RR that holds the signature over a set of DNS records.

Table 1: Most used Resource Record Types used in DNS, including gray shaded DNSSEC RRs.

DNS vulnerabilities
RFC 883 [6] is the first that states the requirements and design of the DNS. It has been updated several times with other RFCs like RFC 1035[5]. However, since DNS was developed without security in mind, it is at its core not secure. Several studies have researched the weaknesses of the DNS protocol [7], [8] and an informational RFC [9] has been written containing a detailed threat analysis of the DNS infrastructure. From these studies it is clear that DNS lacks security and is vulnerable to packet interception which can lead to man-in-the-middle attacks or simple eavesdropping on the communication. The most interesting DNS threat, according to [9] are name chaining attacks, or cache poisoning. With these attacks, adversaries try to get faulty data in a DNS name servers' cache with the consequence that resolvers return an incorrect response, leading the client to a possibly malicious website. Another weakness of DNS from [9] is the possible betrayal by a trusted server. If for whatever reason a DNS resolver or DNS name server that has malicious intentions is queried, the only option for the client that queries is to believe what it receives.

The consequences from these attacks can be severe. A faulty IP-address resolution by the DNS can redirect a user to a fake bank website instead of the legitimate one with all the consequences this entails. In 1997 the first RFC (RFC 2065, [10]) was published to prevent these attacks. This will be discussed in the following Subsection 2.2.

2.2 DNSSEC

To address the above mentioned attacks, a security extension has been created that provides a resolver with a mechanism to trust the data it receives. This extension, called DNS Security Extensions (DNSSEC), guarantees that the data has not been modified by providing origin authentication, data integrity and authenticated denial of existence. It was originally specified in RFCs 4033 [11], 4034 [12] and 4035 [13] and has later been updated in other RFCs [14], [15], [16], [17]. These RFCs have added several resource records to DNS, see Table 1 for an overview of the most used resource records, including the additional new DNSSEC resource records. In this section the purpose of these additional RR are described and the *chain of trust* that is created by this is explained.

Signing and Verifying
The Resource Record Signature (RRSIG) provides the resolver with a way of verifying the integrity of the response of a DNS name server. If all name servers in Figure 1 had enabled DNSSEC, additional RRs would be returned in every response. For the authoritative name servers (3), (4), (5) this would mean that besides the NS or A RR type, also a RRSIG RR type would be included in the response. Such a response then might look as illustrated in Figure 2, in which a query for the A record of the domain 'www.example.com' is made. The RRSIG data contains information on how the signature was created and the signature itself. The value corresponding to the RRSIG type contains, in order, the Type Covered field ('A'), the algorithm that is used ('8',

RSA/SHA-256 [2]), the number of labels in the original domain name ('3') and the original TTL of the queried RR type ('86400'). This TTL field indicates how long the information may be considered valid and can be cached by the resolver for subsequent queries. In this example the resolver could thus cache the result for 86400 seconds (24 hours) before performing the same query again to this name server. The following fields are the expiration and inception dates (20210115201659 and 20201225172223), the Key Tag ('62811') indicating which key pair is used for signing and at last the domain that performed the signing ('example.com') and the signature over the `RRSET` encoded in Base64 ('NHEa. . . ').

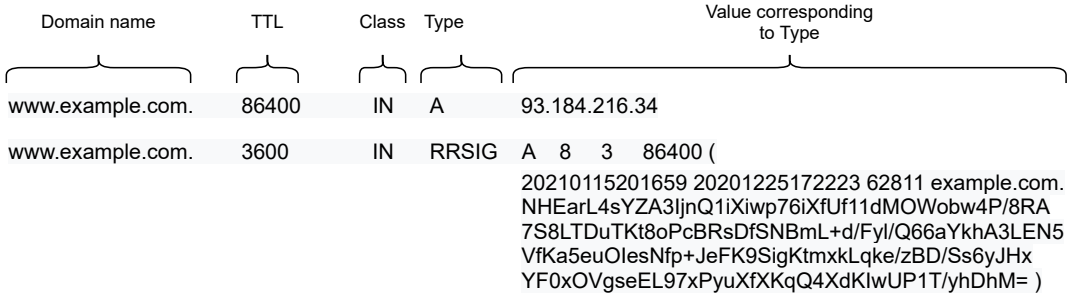


Figure 2: DNS response including `RRSIG`.

Since the DNSSEC infrastructure is based on Public Key Cryptography, before any response can get signed all authoritative DNS name servers that want to sign their responses first have to create a public-private key pair. These keys are referred to as the Zone Signing Keys (ZSK) where the private key signs each `RRSET` in the zone and the public key is used by others to verify the created signature. The public ZSK is made available in the `DNSKEY` RR. Using the three components `RRSET`, `RRSIG` and `DNSKEY` it can be verified that the response came indeed from the owner of the private ZSK. A more detailed view on this mechanism is given in the next paragraph.

The ZSK private and public key are used often in signing and verifying DNS responses, respectively, since for every response its integrity has to be ensured. Signing the zone file containing all `RRSets` has to be done if the zone file is updated, the signatures of the `RRSIG` are expired or the keys are changed. Changing keys is referred to as a *key rollover*. Authoritative servers can also decide to sign DNS responses *on-the-fly*, meaning the `RRSets` are only signed at the moment these are put in a response. The process of signing and verifying thus needs to be fast, especially for DNS servers and resolvers handling large amounts of DNS queries.

Chain of Trust

Verifying that a response came from the owner of a ZSK is of no use if the owner itself cannot be verified or if the ZSK was compromised. In DNSSEC this issue is solved using Delegation Signer (`DS`) records and Key Signing Keys (KSK).

`DS` records contain the (hashed) public keys of its child zones. This way, the public key of a name server can be checked against the (hashed) public key stored in this name servers' parent `DS` record. However, updating these `DS` records at the parents on every ZSK key-rollover of a child results in a tremendous amount of administrative work. To address this, KSKs are added to DNSSEC to reduce the amount of administrative work and communication between name servers.

KSKs are yet another public-private key pair that is meant to sign the public ZSK and the public KSK. These signatures and the public key part of the KSK are also stored in `RRSIG` and `DNSKEY` records respectively. At this point, the public key parts of the ZSK and KSK are public and the signatures over these public keys are also available. Resolvers can then use the public KSK to verify the public ZSK.

Instead of the public ZSK, the (hashed) public `KSK` is kept in a `DS` record by the parent zone. Referring back to the example in Figure 1, this would mean that the authoritative name server (3) has a `DS` record that contains the hashed public KSK used to sign the `.com` zone by name server (4). Name server (4) would in turn make

[2] See `https://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xml` for a list of algorithms and their numerical representation.

a DS record available containing the hashed KSK used to sign the example.com zone file by name server (5). The result of this has been depicted in Figure 3.

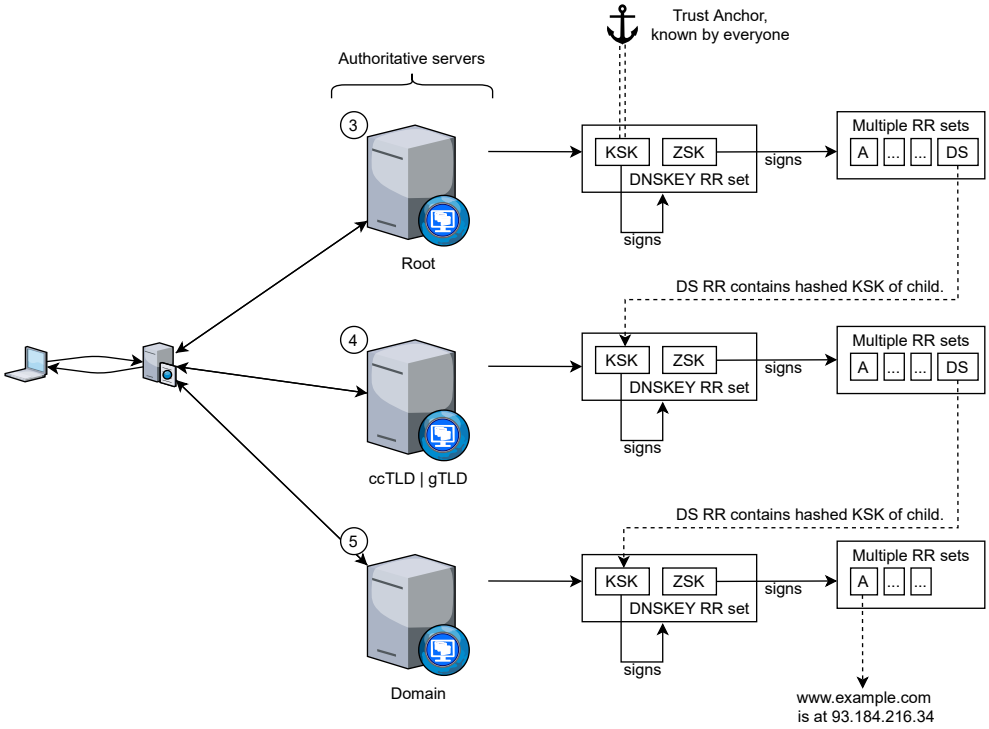


Figure 3: DNSSEC chain of trust from the root to a second-level domain name, based on [18]

Using this structure it is now possible to verify all keys that are used all the way to the root server, creating a chain of trust. Since KSK are generally more secure and do not have to be rolled-over as much as ZSKs, this decreases the amount of administrative work for name servers. As a resolver it is only necessary to store the key of the root server, using its DS records to extend this trust downwards to other name servers.

Purpose of cryptographic algorithms in DNSSEC
As seen above, the DNSSEC infrastructure is complex and the computation and communication overhead depends on the cryptographic algorithm that is chosen to perform the signing and verifying of responses. The chosen cryptographic algorithm has an immediate effect on the size of DNSSEC responses, computation and communication time and thus on the general DNS performance. The goal of this study is to investigate if PQC algorithms in the competition of standardization by NIST can be applied to DNSSEC and what modifications are necessary to adopt PQC algorithms.

Larger signature sizes of PQC algorithms have the consequence that not all of them may fit in one UDP packet, as is discussed in Section 2.3. A DNS query that requires a response to contain several signatures will fail to be sent in one packet if this combination is larger than the allowed size.

Most PQC algorithms have a larger key size than the currently used classical algorithms. This has a direct consequence for the transmission of DNSKEY records, especially for PQC algorithms for which not even one DNSKEY fits in a UDP packet (for example Rainbow, see Table 3). Müller et al. [19] already propose two distinct methods for solving this issue, the first is dividing the key into multiple chunks that do fit in one UDP packet, having the consequence that multiple queries have to be made to fetch each chunk of the complete key. Another method that is proposed works by fetching keys *out-of-band*, i.e. fetching keys from a web server using HTTP. Both methods have their own disadvantages, which need to be investigated.

2.3 DNSSEC limitations & weaknesses

The addition of the Security Extension to DNS gave it the security features that were necessary with the growing use of the Internet and its reliance on DNS. However, there are still several weaknesses to DNSSEC that limit the adoption of new algorithms including the discussed PQC signature schemes. Most notable is the DNS message size limitation and the consequences that arise from this.

The size of DNS packets as originally implemented is limited to only 512 bytes. With the increasing usage of DNSSEC it is not always possible to send the specific DNSSEC additional information like signatures and keys within the limited 512 bytes. To address this, RFC 6891 proposed the Extension Mechanisms for DNS (EDNS(0)) [20]. With EDNS(0), the size of DNS packets could *in theory* be increased to 64 kB. In practice, however, this does not solve the issue of limited packet size. Packets that are too large will often get fragmented since these can exceed the Maximum Transmission Unit (MTU) of the path between the resolver and the authoritative server. These fragmented packets are often blocked by firewalls since these can be used for some types of attacks [21] [22], consequently also blocking fragmented valid DNS packets that cannot be reassembled at the resolver.

In a 2011 study by Weaver et al. [23], the authors show that 9% of clients can not receive fragmented DNS packets over UDP. This results in the DNS resolver retrying the query over a more CPU and memory expensive TCP connection. To avoid fragmentation of packets, van den Broek et al. [24] propose a limit on DNS messages of 1,232 bytes to ensure that there is no fragmentation. A study by Moura et al. [25] in 2020 confirms that most DNS responses (99.99%) for the `.nl` ccTLD are below 1,232 bytes and refers to a comparable result (99.7% below 1,232 bytes) from a Google Public DNS report. The authors of [25] also show that large DNS responses do not occur often and that 75% of the resolvers retry their query over TCP if DNS response packets get fragmented and do not arrive.

This practical limitation of 1,232 bytes and the theoretical limit of DNS messages of 64kB has an effect on several components of DNS, e.g. DNS lookups or zone operations. This is especially relevant when implementing large PQC signature schemes that may have a large signature or large public key that needs to be communicated. It can be seen in Table 3 that this is an issue for most of the algorithms listed there.

2.4 NIST and Post-Quantum Cryptography

The National Institute of Standards and Technology (NIST) is a standardization body in the US. It has initiated a process to research and standardize public key algorithms that are secure against adversaries that have both quantum and current computers. Researchers can propose algorithms for digital signatures, public-key encryption and key establishment to be reviewed by NIST and interested parties from the cryptographic community. NIST initiated the process in 2017 and has already completed two rounds. In these two rounds a total of 67 candidate algorithms have been proposed and evaluated by NIST and experts in the community for their performance and security.

In July 2020 a status report was released by NIST announcing the 15 second round candidate algorithms that are advancing to the next round of the standardization process [26]. These 15 candidates that are advancing to the third round are divided into third-round finalist and third-round alternate candidates. The finalists are considered by NIST to be most promising for implementation in the majority of use cases and are thus most likely to be standardized at the end of the third round, which could take until 2024[3]. The alternate candidates that advance to the next round could be standardized in the future after their security or performance has been improved. NIST will focus first on the finalists.

For the purpose of this study, only the digital signatures algorithms are discussed since these are directly applicable to this study. An overview of the other finalist and alternate candidates in the categories public key encryption and key encapsulation mechanisms can be found in [26].

Post Quantum Cryptography
Post Quantum Cryptography (PQC) is a group of cryptographic schemes that run on classical computers and are secure against an adversary that has access to a quantum computer. Current cryptographic algorithms rely on hard problems from number theory such as integer factorization or the discrete log problem. These problems could be solved in polynomial time using a quantum computer that is powerful enough to run Shor's

[3]For a timeline, see `https://csrc.nist.gov/Projects/post-quantum-cryptography/workshops-and-timeline`

Level	Security Description	NIST FAQ Description
I	At least as hard to break as AES128 (exhaustive key search)	Likely secure for the foreseeable future, unless quantum computers improve faster than is anticipated.
II	At least as hard to break as SHA256 (collision search)	Probably secure for the foreseeable future.
III	At least as hard to break as AES192 (exhaustive key search)	Probably secure for the foreseeable future.
IV	At least as hard to break as SHA384 (collision search)	Likely excessive.
V	At least as hard to break as AES256 (exhaustive key search)	Likely excessive.

Table 2: NIST security levels, adapted from [31] and the NIST PQC FAQ[4]

algorithm [27], thus breaking the security of the cryptographic scheme as a whole. To keep DNSSEC and the Internet secure, algorithms that use hard problems that are not currently known to be solvable by either classical or quantum computers are necessary.

The PQC algorithms that are advancing to the third round of the NIST standardization process can be separated into different families based on the approach these algorithms take. The families of algorithms that are believed to be able to resist an attack with a quantum computer and are used in the NIST digital signature schemes are from the lattice-based cryptography, hash-based cryptography and multivariate cryptography families. Some of the characteristics of these families are shortly summarized below. Other groups that will not be discussed in this paper but are also believed to be able to withstand a quantum computer attack are code-based [28] and supersingular elliptic curve isogeny cryptography [29]. None of the third round finalists or alternate candidate algorithms in the NIST competition are based on these latter two.

- **Lattice-based cryptography** is fast and has been studied a lot in the past years. It uses a Shortest Vector Problem (SVP, [30]) which is NP-hard and currently no quantum algorithm to solve this problem exists. Lattice-based algorithms seem generally suitable to use in DNSSEC since these algorithms generally have small signature and key sizes.
- **Hash-based cryptography** algorithms rely on the collision resistance of the chosen cryptographic hash function. They have small keys, but large signatures and the computational load with signing as well as verifying is high. A big advantage of hash-based signature schemes is that the used hash function can be replaced with a new and secure one if it is found to be broken.
- **Multivariate cryptography** algorithms are very efficient. All computations are done over finite fields and the decryption process only consists of the solutions of linear equations. Multivariate signature algorithms generally have small signature sizes and little computational overhead for verification, however, they do have much larger public key sizes.

NIST requested submitters of algorithms to focus on security level strength categories 1, 2 and 3 [31]. An overview of the security levels that NIST defines can be found in Table 2.

The 3 finalists and 3 alternate candidates for digital signatures are summarized in Table 3 and will be shortly discussed below. All of the algorithms in Table 3 are of Security Level 1.

- **Crystals Dilithium**[32] is one of the two final lattice-based cryptographic signature schemes still in the competition. Dilithium is designed to be simple to implement and it tries to minimize the size of the public key and signature combined, the latter being a good design choice to be used in DNSSEC. However, looking at Table 3 it is clear that both public key and signature size are quite large compared to its main competitor Falcon.
- **Falcon**[33] is the second lattice-based scheme and main competitor of Dilithium. The creators of Falcon have a leading design basis to minimize the public key size and signature size, respectively being 897 and 666 bytes. Compared to Dilithium this is smaller, but has a disadvantage that it is more complex to implement and has less efficient key generation. However, signing and verifying is efficient. NIST expects either Dilithium or Falcon to be standardized at the end of the third round.

[4]See "Which security strength categories will NIST consider for standardization?" at `https://csrc.nist.gov/projects/post-quantum-cryptography/faqs`

- **Rainbow**[34] is the only multivariate finalist that moves on to the next round of the NIST competition. It is based on the Unbalanced Oil-Vinegar (UOV, [35]) signature scheme and has added layers that provides the scheme with more efficiency. Compared to Falcon it has a faster signing process but is slower verifying signatures. In a recent study by Beullens [36] two new attacks on the Rainbow algorithm are discussed. The author proposes new parameter sizes to be used in the algorithm which could protect against these attacks. These new parameter sizes would increase the already large public key size of 158,800 bytes to 203,000 bytes and the signature size of 66 bytes to 71 bytes. The recent found attack and Rainbow having the largest key sizes of all finalists are a major disadvantage of this scheme.
- **GeMSS**[37] is the competitor of Rainbow in the multivariate category. It is not selected as a finalist since GeMSS has larger keys and is difficult to implement on very low-end devices. GeMSS has the smallest signature size of all finalist and alternate candidates with only 35 bytes. However, it also has the biggest public key with 375,212 bytes. Signing times are slow which could pose a problem for DNSSEC if records are signed on-the-fly. Different parameters that are proposed by the authors of GeMSS [37] create RedGeMSS and BlueGeMSS which offer flexibility in performance. In Table 3 the RedGeMSS is used.
- **Picnic**[38] is a hash-based cryptographic scheme. Its signatures are based on zero-knowledge proofs of the knowledge of the private key. It has small key sizes of 16 and 32 bytes but a large signature size of 34000 bytes and a low sign and verify speed. It is selected as an alternate candidate since its security only depends on assumptions about symmetric primitives, which could allow for a very conservative signature standard if that is necessary in the future. However, due to the large signature size and the slow signing and verifying it is likely not suitable to be used in DNSSSEC.
- **Sphincs+**[39] is based on the security of the underlying hash function. It has a small public key of only 32 bytes but a large signature size of 8080 bytes. It is chosen as alternate candidate since a lot of research has already been done on breaking hash based cryptographic algorithms. This could be a very conservative choice for standardization by NIST. However, for DNSSEC, the fact that signing and verifying is slow makes it very likely not suitable to use.

NIST expects either Crystals Dilithium or Falcon to be standardized as the primary PQC scheme for signatures at the end of the third round. Both these schemes are lattice based and are most promising for general-purpose digital signature use according to NIST.

Candidate	Algorithm	Approach	Private key size	Public key size	Signature size	Sign/s	Verify/s	Regarding DNSSEC
Finalist	CRYSTALS-DILITHIUM-II [32]	Lattice	2.8kB	1.2kB	2.0kB	-	-	Large signatures
	FALCON-512 [33]	Lattice	57kB	0.9kB	0.7kB	3,307	20,228	Suitable (with limitations)
	Rainbow-I_a [34]	Multivariate	101kB	158kB	66B	8,332	11,065	Large public key
Alternate	RedGeMSS 128 [37]	Multivariate	16B	375kB	35B	545	10,365	Large public key
	Picnic [38]	Hash	16B	32B	34kB	-	-	Large signatures
	SPHINCS$^+$ – *Haraka* – 128*s* [39]	Hash	64B	32B	8kB	-	-	Large signatures

Table 3: Performance metrics of NIST PQC candidates, adapted from [19]

3 Related work

This section gathers research that relates to this study. First, a study that defines requirements that a PQC algorithm should have to be considered for implementation in DNSSEC is described in Subsection 3.1. Afterwards, studies are discussed on the implementation of non-PQC algorithms in DNSSEC in Subsection 3.2. In Subsection 3.3, studies on the implementation of PQC algorithms in other Internet Protocols are discussed. Finally, in Subsection 3.4, real-world experiments by Cloudflare and Google are discussed and related to this study.

3.1 PQC in DNSSEC

A recent case study by Müller et al. [19] is the first that studies and formulates requirements for Post Quantum (PQ) algorithms and the implementation in DNSSEC. In this work, the authors define a set of requirements that a PQC algorithm has to satisfy before it can be implemented in DNSSEC. These requirements can be used to identify protocols that are currently in the process of being standardised by NIST, as discussed in Section 2.4. The most important requirement for using quantum-safe algorithms in DNSSEC is the signature size, which the authors discuss needs to be below $1,232$ bytes. Besides larger DNS responses being more favored in amplification attacks, a large signature size may have the consequence that packets become fragmented since they exceed the Maximum Transmission Unit (MTU). These fragmented packets may either not arrive at their destination or be used to perform a DNS spoofing attack. With this reasoning the authors claim that signature size is the most crucial requirement.

Another important requirement is the validation speed of signatures. In [19], $1,000$ signatures per second is taken as minimum, given that resolvers nowadays process hundreds of validations per second and DNSSEC adoption will grow in the future. The second to last and last priority requirements are the key size and signing speed, respectively. The key size should best be below 64 kilobytes, although keys that are larger could be accepted conditionally if a mechanism can be implemented in DNSSEC that can efficiently distribute keys. For this, the authors discuss two options that both modify the `DNSKEY` Resource Records. The first solution divides the public key into small chunks that can fit in one DNS packet. The whole key can then be fetched from a name server by performing multiple queries, each for a different chunk of the key. The other solution the authors propose is by providing the key via an out-of-band method. A resolver could fetch the key via a URI provided in the `DNSKEY` RR via an HTTP request.

3.2 New algorithms in DNSSEC

At the moment of this writing, no further research exists on applying PQC algorithms in DNSSEC. There are, however, studies done on other new cryptographic algorithms in DNSSEC. In a paper by van Rijswijk-Deij et al. [40], a switch of algorithms in DNSSEC from RSA to Elliptic Curve Cryptography (ECC) is studied. Since ECC signatures are slower to validate than RSA signatures, an extra workload has to be processed by DNS resolvers. This extra CPU usage could create problems for DNS resolvers.

To determine the number validations that a resolver has to handle, the authors created a model that accurately predicts the validations a DNS resolver has to perform. This model is validated against measurements from four resolvers and is found to be a good predictor of the number of signatures that need to be validated by a resolver given the number of outgoing queries. The authors find that a DNS resolvers can handle the increased computational load of verifying ECC signatures. Even in a worst-case scenario where the adoption of DNSSEC is 100% and the most signature validations have to be performed on a single CPU. This makes a switch from RSA to ECC possible based on only the CPU load. However, there are other problems that need to be tackled. An attack surface that is created with a transition to ECC is a Denial-Of-Service (DOS) attack based on CPU starvation. The authors discuss and test scenarios in which queries can be constructed that would result in such a large amount of signatures that need to be validated that a resolver might not have enough resources to do these computations on-the-fly. The experiments that the authors have done show that a DOS attack based on CPU starvation is a possibility and suggest that existing Response Rate Limiting solutions could be used to prevent this.

Besides a possible DOS opportunity by attackers, other hurdles for adopting ECC are discussed, including the support of signer software and support of resolvers, registries and registrars that is necessary to successfully

switch from RSA to ECC. Although the authors believe these hurdles are quickly solved by the Internet community, these are also applicable to when switching to PQC algorithms in DNSSEC.

In a later research by van Rijswijk-Deij et al. [41] the actual use of an ECC based signature algorithm (ECDSA) is studied. For this study, data sets on the Top-Level Domains (TLDs) `.com`, `.net`, `.org`, `.nl` and `.gov` covering approximately 50% of the whole DNS namespace were used. By analysing DNSSEC Resource Records of the types `RRSIG`, `DNSKEY` and `DS` the authors classified the adoption of ECDSA in DNSSEC in two groups, either a *partial* or *full* adoption. In the latter, the domains include signatures on their DNS responses and have a `DS` record in their parent zone. *partial* adoption does not have this record causing the the *chain of trust* to be broken. In the four years between the standardization of ECDSA in 2012 and the research in [41] from 2016, only 2.3% of `.com` domains are signed using an ECC algorithm. From this we can conclude that adoption of ECDSA by DNS resolvers and the necessary modifications to the software is slow. A work by Müller et al. [42] studies the general process of deploying new algorithms in DNSSEC (*algorithm rollover*) and confirms that adoption of new algorithms at registrars and DNS operators is slow. It took ECDSA more than 10 years to get standardized and gain enough support from registrars, registries and resolvers to be deployed at domain names.

The authors find positive and negative factors that influence the deployment of new algorithms. Most notable is that operators only move to a new algorithm if there is a good incentive for operators to switch, which could either be a financial reward or a performance improvement such as smaller signatures. The authors indicate that a security threat encourages a rollover only if the threat is imminent. Given that most of the PQC algorithms have larger signatures and the threat is *not yet* imminent since quantum computers do currently not exist, other incentives to have operators switch to a PQC algorithm must be thought of. Developing methods to apply PQC to DNSSEC and push adoption for this should be started preferably sooner than later in order to have this implemented before quantum computers are capable of breaking current DNSSEC.

3.3 PQC in Internet Protocols

At this moment, experiments with TLS and SSH protocols focused on supporting PQC algorithms are done. Although the experiments are limited to only these protocols, a large part of encrypted internet traffic is covered with these two.

Sikeridis et al. [43] study how using PQC algorithms in TLS 1.3 would impact the TLS handshake time under realistic network conditions. By implementing different PQC signature algorithms in the TLS 1.3 protocol they find that signature size does impact the total handshake time and that signing is an important factor for the performance of a server. The authors expect that signing performance can be increased by optimizations and hardware acceleration and that the signature and key size then has the most impact on TLS handshakes. An advice is given that for time-sensitive applications Dilithium and Falcon algorithms could best be used since these algorithms had the lowest TLS handshake time. For other applications that do not require full and frequent connection establishment, other PQ algorithms might fit as well as long as they do not cause the server to terminate connections if it is overloaded. Converting these results to this study on DNSSEC, it is clear that we do not need full and frequent connection establishment since DNSSEC is (at first) performed over UDP. In the case a TCP retry is necessary, this is not encrypted and thus no TLS handshake will take place. The focus should therefore not lie on the connection establishment, but on the other factors that have an impact on the performance of PQC algorithms. This is the size of the signatures that are produced and the speed that the server has to handle DNS queries. This is especially the case if the server is signing responses *on-the-fly*. Expensive sign operations then have a large computational overhead, increasing the total query time.

Crockett et al. [44] discuss the challenges of implementing PQ in TLS and SSH, with a focus on implementing hybrid schemes, using a combination of PQ and classical cryptographic schemes. The authors discuss various design considerations that are applicable to SSH and TLS. These internet protocols are both designed with algorithm agility in mind, meaning it is easy to switch algorithms. DNSSEC has achieved algorithm agility only partially, claims [42], making it more complex to switch from one algorithm to another. The primary goal of a hybrid mode as discussed by the authors of [44] is that the security of such a hybrid scheme holds as long as at least one component remains unbroken. Especially in the early years of PQC deployment, confidence in these new PQ algorithms might be low. Having a hybrid scheme that supports both PQ and classical algorithms ensures proper security as long as at least one scheme remains unbroken. The algorithm negotiations used in TLS and SSH that are discussed in [44] do not apply to DNSSEC since there is currently

no negotiation. All algorithms that are used in DNSSEC are defined by a fixed 8 bit number, see Section 2.2. The issue in DNSSEC using such a hybrid mode would be that resource records would need to be signed by both algorithms, increasing the computational overhead. Besides the computational overhead, an additional communication overhead is created since the signatures and keys of both algorithms have to be transmitted to the client, increasing the packet size of the UDP packets.

In a study by Herzberg et al. [45], the authors propose a negotiation protocol that allows the resolver to indicate which cryptographic algorithm it prefers to have the response signed with. Their cipher-suite negotiation protocol uses the EDNS(0) extension mechanism that is already implemented in DNS. With this negotiation protocol, the resolver adds a list of supported ciphers to the query for the name server, which can be used to determine which algorithm to use in the name servers' response. This reduces the communication overhead since now only one algorithm is communicated, instead of sending *all* signatures and keys of different algorithms to the resolver. This negotiation protocol could be applied in a hybrid mode of operation as described by Crockett et al. [44]. Indicating the cryptographic preferences of a resolver could, especially in the early days of PQC deployment, be useful for resolvers that do not trust PQ algorithms for signing responses.

In a research of Paquin et al. [46], the authors develop an experimental framework for measuring the performance of the TLS protocol under a variety of network conditions. The authors focus on hybrid key-exchange in their experiments, using a PQ scheme to ensure a longer period of forward-secrecy while using classical schemes such as ECDH key exchange since trust in the early days of PQC algorithms might be low. However, other experiments focused on PQ authentication are not done using such a hybrid scheme. This choice is made since connections only need to be secure at the time a connection is established, there is no need for extended security over a longer period of time. It can be argued that DNSSEC does not need this extended security as signatures over resource records only need to be validated at the time they are requested. However, since these RRs can be cached for several days, weeks or even months depending on their TTL field, combined with the partial agility that DNSSEC has achieved, it is important to already investigate the use of PQC in DNSSEC. The authors of [46] analyze the impact that PQC algorithms have on the TLS 1.3 handshake completion time and come to the conclusion that on fast internet connections with low packet loss the TLS handshake completion time is dominated by the PQC cryptography, corresponding to findings of [43].

3.4 Real-world PQC experiments

Besides academic studies, experiments with applying PQC in real-world applications are performed by big industry players such as Google and Cloudflare. In a blog post in 2016, Braithwaite[47] describes how Google added a post-quantum key-exchange algorithm, called CECPQ1, to TLS connections from Chrome Canary, the developer version of Googles browser, to Google's servers. With this experiment, Google aimed to get real-world experience with handling all aspects that PQC algorithms require and at the same time it tried to put a focus on an area that Google finds to be important. In [48], Langley describes the results from the experiment and found that the increased message size added latency to the TLS communication, which is especially troublesome for people on slower internet connections.

In a later experiment by Google and Cloudflare in 2019 [49] [50], both client- and server-side data was collected on TLS connections that were using successors of CECPQ1, called CECPQ2 and CECPQ2b. The two algorithms differ in key sizes and computational costs, where CECPQ2 has a larger key size but is computationally faster while CECPQ2b has shorter key sizes but is computationally slower. The goal of this experiment was to determine the performance of TLS 1.3 key-exchange with these PQ algorithms in a real-world setting with millions of devices using Google's Chrome browser. Kwiatkowski and Valenta conclude in [49] that CECPQ2b performs worse than CECPQ2, meaning the small key sizes do not make up the large computational costs that it has. In [51], Langley states that only for the 5% slowest connections the smaller messages of CECPQ2b create an advantage but that the computational advantages of CECPQ2 make it a more attractive choice to use in TLS. Comparing this to the results of the earlier experiment discussed in [48], it could be said that a smaller message size gives only an advantage on slow connections while on faster internet connections the computational load is more important.

Applying the results of the experiments from Google and Cloudflare to a DNSSEC situation, shorter messages would fit better in UDP packets which might reduce the fragmentation of packets resulting in less TCP retries. However, a faster computational solution would increase the number of queries a name server can process in

the case of on-the-fly signing, making it more useful for name servers that are authoritative for large zone files that have to handle a lot of queries. A simple translation from TLS results to the DNSSEC situation is not easily made.

4 Problem Statement and Research Goal

Post quantum cryptographic algorithms currently evaluated by NIST have a significantly larger key size and signature size compared to currently used algorithms in DNSSEC including RSA or ECDSA algorithms. Communicating these larger messages between name servers and resolvers pushes the limits of the DNSSEC protocol. Additionally, the extra computational load of signing DNS zone files for name servers and the verifying load for resolvers may also increase when switching from current algorithms to PQC algorithms in DNSSEC. At the moment, several PQC alternatives are being evaluated by NIST. Irrespective of the outcome of the NIST competition, it remains unclear what PQC alternative will be most suitable to use in DNSSEC. In a worst case scenario, the computational load or communication overhead of large signatures and keys might make DNSSEC not useful and leaves the DNS infrastructure open to attacks by quantum computers.

The following question is formulated to address the above stated problem:

Main RQ: Can DNSSEC still be useful in a quantum world?

In order to find an answer to this research question, several sub research questions need to be answered:

RQ 1: What is the effect of PQC algorithms on the packet size of responses for DNS lookups?

RQ 2: What are the consequences of PQC algorithms for DNS signers and verifiers at computational level?

RQ 3: How can we address the negative effects on DNS operations in order to apply PQC to DNSSEC?

RQ 4: What is the effect of the proposed measures on DNS lookups and zone operations?

These research questions capture the most important aspects of the DNS infrastructure. RQ 1 focuses on the communication aspect and the additional communication overhead that PQC signature schemes would need to communicate the larger signatures and public keys. RQ 2 captures the computational aspect of both the name server and the resolver that, respectively, create the signatures and verify the signatures. The combined knowledge of these two RQ's is then used in RQ 3 that addresses the found issues. This solution is then studied in RQ 4 to determine whether it positively .

By answering the sub research questions and subsequently the main research question of this study, a case can be made for which PQC alternative, if at all, can be adopted by the DNSSEC community and keep DNS secure in the quantum computer age.

5 Research Methodology

Following the research questions that are formulated, a comparison between currently used cryptographic algorithms and new PQC alternatives should be made. This section describes the methodology to answer the research questions. Analysis results and conclusions based on these are gathered in its respective sections later in this thesis.

5.1 RQ 1

First, the effects of PQC alternatives on the communication between name servers and resolvers is studied. DNS lookups mainly use the UDP transport layer for performing a lookup. Although not happening often, a resolvers switches to use TCP in case the UDP DNS lookup fails or if the response is larger than the advertised maximum buffer size [25]. Given the larger signature and key sizes from PQC algorithms, see Table 3, it is expected that DNS responses grow larger when these algorithms are used in DNSSEC. This raises the questions whether a DNS query over UDP can still succeed or that more often a TCP request has to be sent. Additionally, if DNS responses are too large, it does not fit in the DNS protocol. This is due to the fact that there exists a 64kB limit on the size of a DNS message, see Section 2.3.

In order to answer this question, real-world DNS traffic that is collected at a DNS resolver at the University of Twente, see Section 5.5, is used. Every signature and key in this traffic is swapped for PQC signatures and keys to see the consequences if DNS name servers and resolvers would switch to using PQC algorithms instantly. Based on the traffic analysis a conclusion for this research question is drawn in Section 6.

5.2 RQ 2

Where *RQ 1* mainly looked at the communicational aspect of DNS with PQC algorithms, another aspect that needs to be studied is on the sender and receiver side. DNS name servers must be able to handle the DNS operations that have to be performed on signed DNS zones and the resolvers must be able to verify responses quickly once received. Besides the larger signature and key sizes, most PQC algorithms also have an increased computational load. This could lead to resolvers or name servers not being able to serve their clients in time due to the computational overhead.

To study the computational load for DNS name servers, a theoretical approach is taken and a comparison between signing times for classical algorithms and PQC algorithms is made. Properties regarding signing and verifying times can be obtained from the eBACS project[5]. This project benchmarks cryptographic systems and provides performance results for PQ public-key signature algorithms currently evaluated by NIST.

Additionally, a practical approach is taken by analysing the current performance of signing the `.nl` zone file which is managed by SIDN[6]. Together with the data from the eBACS project, a prediction is made on how the .nl domain can be managed using Post-Quantum algorithms.

Since name servers can sign responses in advance or *on-the-fly*, both scenarios have to be taken into account to see what the optimal mode of operation is and how this computational load differs from the current situation where non-PQC algorithms are used.

Where name servers can sign their responses in advance, DNSSEC validating resolvers are limited to perform the validation *on-the-fly*. By estimating the current resolver performance from the obtained dataset, see 5.5, the additional computational overhead for verifying some PQC signatures can be computed. The possible additional verification time must not significantly increase the response time for its clients or result in a Denial of Service since the resolver could get overloaded.

5.3 RQ 3

Based on results from RQ 1 and RQ 2 solutions can be thought of to address the effects that are seen. Literature already mentions solutions for issues that might arise, such as in [19] where the authors indicate that key exchanges cannot fit in a UDP packet and propose alternatives to exchange keys out-of-band. If either of

[5] eBACS: ECRYPT Benchmarking of Cryptographic Systems, see `http://bench.cr.yp.to/`

[6] The Dutch non-profit organisation that manages the .nl domain, see `https://www.sidn.nl/en`

the research questions above might indicate that key exchanges are a negative effect, such solutions might be thought of and be implemented. This idea of out-of-band transmission might even be extended to signatures as well if it turns out these are too large and cause too many issues. Out-of-band key exchange can use a variety of transport protocols, each having its own properties. In this RQ, several network protocols are described and implemented in a DNS resolver to study the performance based on the time it takes to fetch a DNS public key out-of-band.

5.4 RQ 4

Several transport protocols are implemented in a DNS resolver to provide an out-of-band key exchange method. DNS lookups using this method of out-of-band key exchange over different transport protocols are benchmarked on the time it takes to fully resolve a domain name. With these measurements, the effects of different transport protocols on the resolving time are studied. RQ 4 compares these measurements and focuses on security considerations regarding the implemented solutions. Additionally, scalability of the different transport protocols when an out-of-band key exchange method is adopted into the DNS infrastructure is investigated.

5.5 Captured Data & Ethical Considerations

This study uses DNSSEC data captured for a period of 24 hours at a resolver located on the campus of the University of Twente. This resolver is at least used by all research and student housing buildings on the campus. All data that is captured consists solely of DNS responses from authoritative servers arriving at this recursive resolver. This data is free of any personally identifiable information.

DNS Resolver machine
The machine running the DNS software has an Intel® Xeon® Processor E5-2403 CPU running at 1.80 GHz. The DNS software running on this machine is a commercially available solution.

Statistics
An overview of the captured data is given in Table 4 summarizing the packets for different rcodes and query types relevant to this study. Since only responses from name servers to the resolver are captured, the data is limited to the actual packets that are received by the resolver after a successful query, i.e. there is no data for queries that are dropped or whose response did not successfully arrive at the resolver.

Topic	Item	Count	Percentage
Total packets captured		11,640,252	–
rcode	NOERROR	11,232,493	96.50%
	NXDOMAIN	274,994	2.36%
	Other	132,765	1.13%
query type	A	6,226,458	53.49%
	AAAA	3,827,648	32.88%
	DS	705,667	6.06%
	DNSKEY	112,146	0.96%
	Other	768,333	6.61%

Table 4: Relevant statistics for this study of captured data at a resolver at the University of Twente.

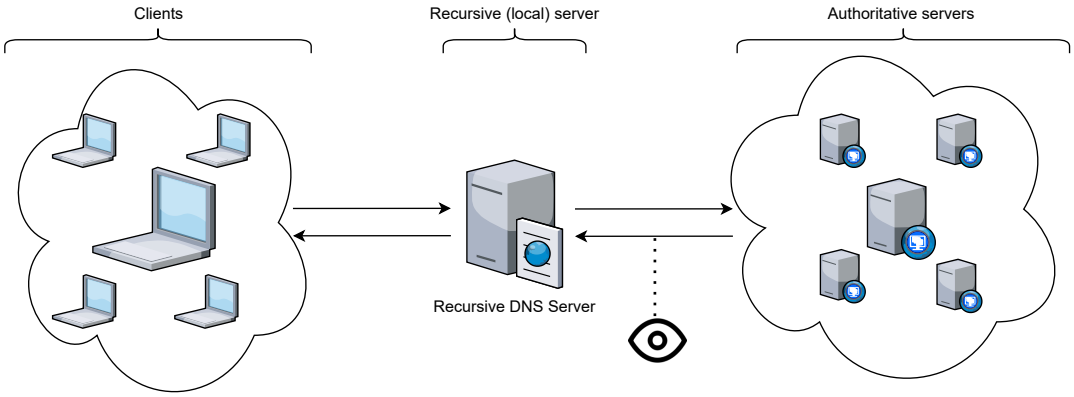


Figure 4: Outline of DNS structure and location of packet capture.

6 Comparison of current and PQC DNSSEC communication

In this section, captured traffic as described in Section 5.5 is analysed. The goal of this analysis is to research the effects of PQC algorithms on the packet size of DNS responses. First, the methods that are used to extract useful information from the packet capture files are described. Then, a comparison is made on how this traffic behaves if current cryptographic algorithms are swapped for several PQC alternatives. The results are discussed afterwards in Subsection 6.5.

6.1 Analysis Setup

The data was delivered in 32 separate PCAP files covering a period of 24 hours of DNS responses arriving from authoritative name servers at the resolver. In Figure 4 the DNS infrastructure is illustrated, the eye symbol indicates the location in the network where the DNS traffic has been captured. For convenience, all 32 separate PCAP files were merged into one single file that is used for the remainder of this study.

In the following paragraphs, each step in the analysis is described. Afterwards, the results from this analysis are discussed.

Information extraction
The PCAP file contains of a lot of redundant information that is not necessary for this study. Extracting useful information from the packet capture is done using TShark. In Appendix A, listings 1 and 2, the TShark commands that are used for respectively UDP and TCP packets are listed. The resulting data are stored in a CSV file that can be easily used for analysis and modification of the traffic. The extracted fields for UDP and TCP DNS responses are listed in Table 5, also an short description of these fields is given in this Table. Most fields are shared between TCP and UDP packets, in the last rows of Table 5 the UDP and TCP specific fields are listed.

Traffic modification
Using the CSV file generated above, it is possible to calculate the size of DNS responses if all DNS keys and signatures are swapped for PQC alternatives. To do this, first the size of the total DNS payload is calculated, after which the size of the keys and signatures are subtracted if these are present. Afterwards, the sizes of cryptographic alternatives as listed in Table 3 can be added to calculate the packet size when PQC cryptography is used to sign DNS zones.

6.2 Results

The findings of the analysis of the captured traffic are given in the following paragraphs. A comparison is made between current traffic and altered traffic where all signatures and keys in are replaced by PQC alternatives currently evaluated by NIST. At the end a conclusion is drawn on the effects of using Post Quantum Cryptographic alternative algorithms in DNSSEC.

UDP and TCP fields	Explanation
dns	Flag to indicate it is a DNS response
frame.time_epoch	Arrival time of packet since epoch
dns.qry.type	Query type
dns.qry.name	Query name
ip.src	IPv4 source IP
ipv6.src	IPv6 source IP
dns.rr.udp_payload_size	Advertised maximum UDP payload size
dns.resp.z.do	DNSSEC OK flag
dns.resp.edns0_version	EDNS(0) version
dns.rrsig.algorithm	Algorithm used to create the signature
dns.rrsig.signature	Hexadecimal representation of the signature
dns.rrsig.type_covered	Fields covered by signature
dns.dnskey.algorithm	DNSSEC public key algorithm
dns.dnskey.flags	DNSSEC key flags to indicate a ZSK
dns.dnskey.public_key	The DNSSEC public key in hexadecimal format
Additional UDP	**Additional TCP**	
udp.payload	tcp.payload	The UDP or TCP payload in the packet, respectively
	tcp.reassembled.data	TCP reassembled data for DNS responses that are sent over multiple packets
	tcp.reassembled.length	Length of reassembled TCP data

Table 5: Fields extracted from UDP and TCP packets.

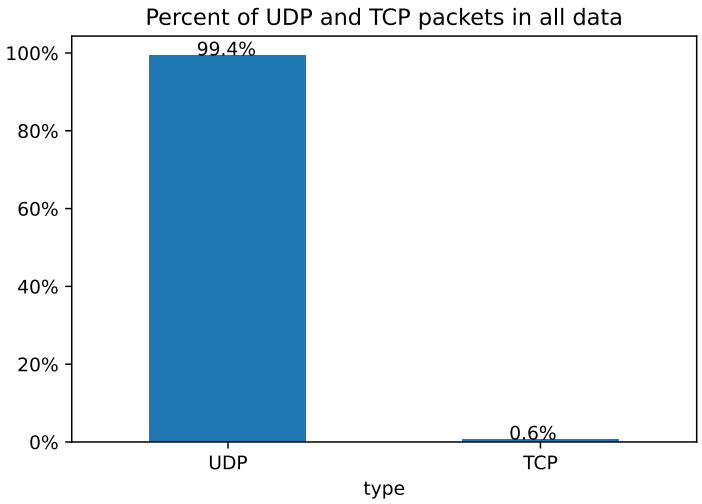


Figure 5: Distribution of traffic over UDP and TCP.

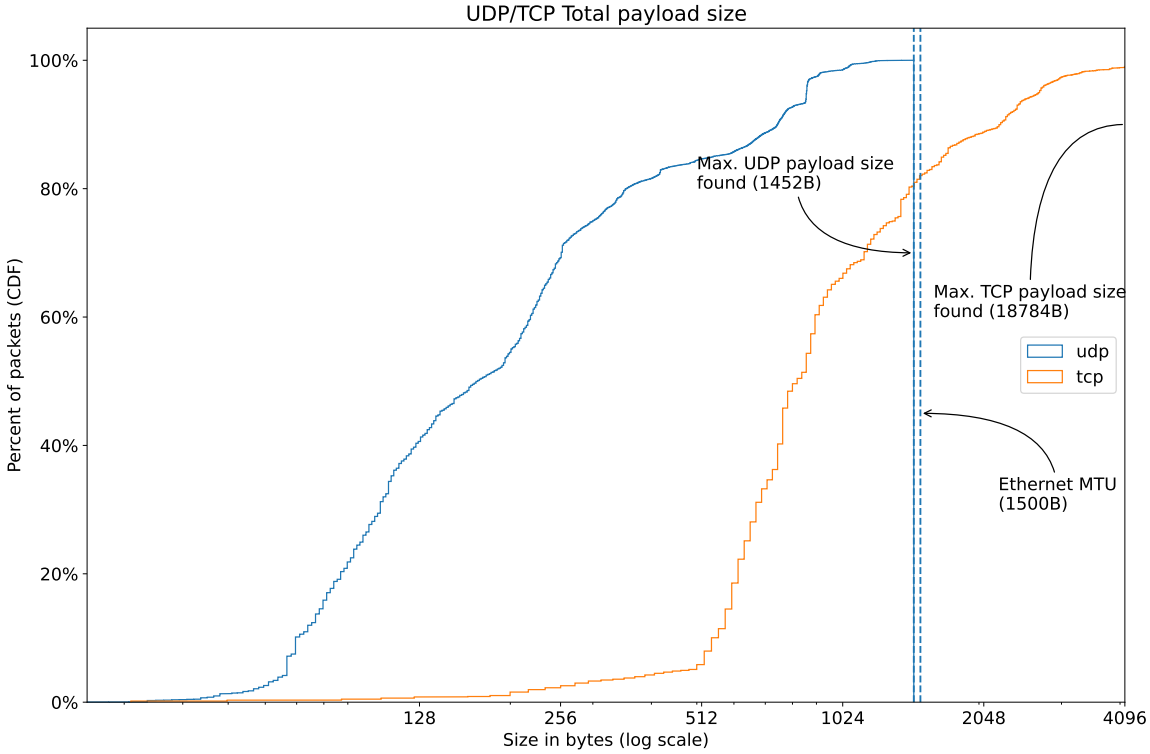


Figure 6: DNS response payload size (CDF, log scale)

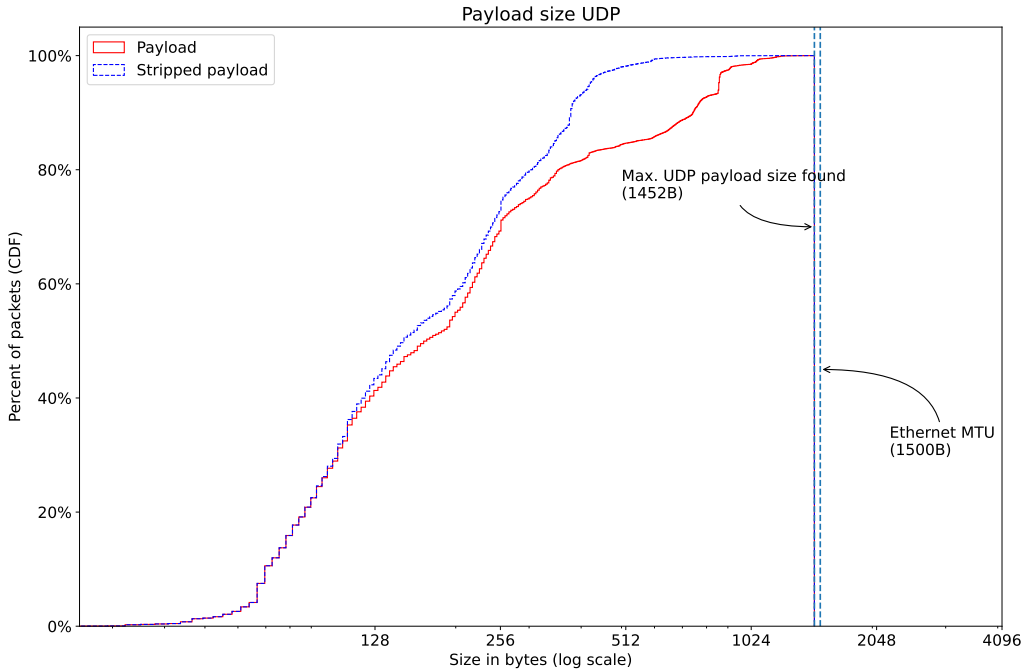
6.2.1 Current traffic analysis

The captured traffic, which contains over 11.6 million packets, consist for over 99% of UDP traffic, see Figure 5. In Figure 6 the payload size of all DNS packets is plotted. From this Figure, we can conclude that all UDP traffic is smaller than the Ethernet MTU of 1500 bytes, the largest payload having a size of 1452 bytes. From this Figure can can also be concluded that TCP packets are generally larger than UDP packets, with a maximum TCP payload size of more than 18KB. The largest TCP packets consist of `MX` and `ANY` query responses.

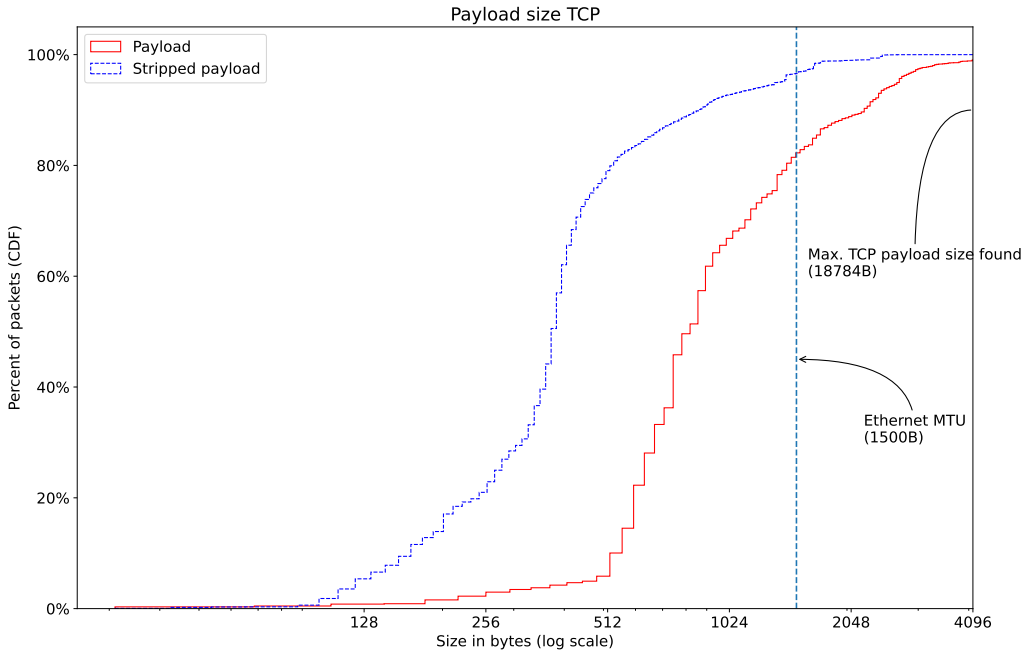
To further understand how this DNS traffic is constructed, in Figure 7 the payload size for TCP as well as for UDP is illustrated in the same way. In both graphs, the payload length is compared to the payload length when stripped of all DNS keys and signatures, revealing the 'plain' DNS response sizes without DNSSEC public keys and signatures. The shift that is visible in this graph is the size difference that DNS keys and signatures create on the responses. The largest shift that is seen here occurs in TCP traffic whereas the shift between the UDP payload and UDP stripped payload is much smaller. This indicates that DNSSEC key and signature material is communicated more over TCP than over a UDP connection.

On further analysis, it is found that less than a percent of all UDP packets contain DNSSEC public keys and only 22% of the UDP packets contain signatures. A comparison with TCP packets is visualized in Figure 8. From this figure, it becomes clear DNS responses over TCP are more likely to contain DNSSEC public keys or signatures. Hence the 'shift' that is visible in Figure 7 is larger for TCP than for UDP. Since the packet capture of network traffic only consists of DNS responses that are successful, we cannot indicate whether these TCP connections are the result of TCP retries. Perhaps, UDP connections are set up first and a TCP retry has to be done since the signatures or keys do not fit in a UDP packet or the MTU size of the ethernet link between resolver and name server is too small.

In Figure 9, the distribution of the 7 most queried records are illustrated. What is visible from this, is that UDP connections are most likely to contain A or AAAA records. TCP packets, on the other hand, are more likely to contain DNSSEC resource records such as DS or DNSKEY records, compared to UDP packets. Given that the traffic consists for over 99% of UDP traffic, it can be said that most DNSKEY and DS records are still communicated over UDP connections. However, if a TCP connection is used, the chances of it containing



(a) UDP payload compared to UDP payload without signatures and keys.



(b) TCP payload compared to TCP payload without signatures and keys.

Figure 7: TCP and UDP traffic stripped of keys and signatures.

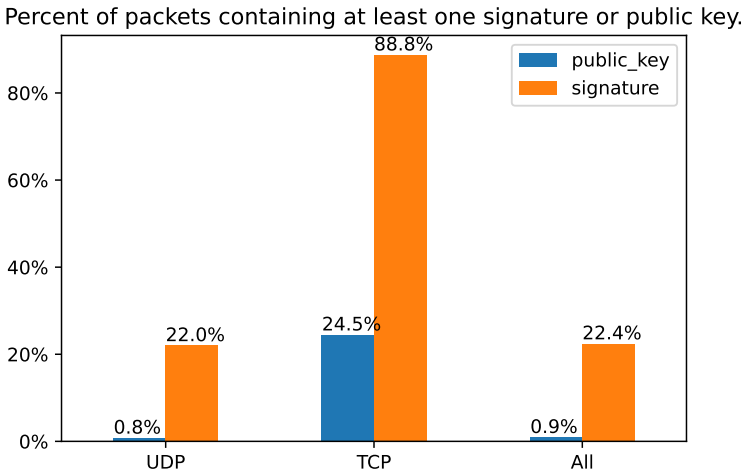


Figure 8: Packets containing DNSSEC public keys and signatures.

Algorithm	Public key size	Signature size
CRYSTALS-DILITHIUM-II	1,200	2,000
FALCON-512	900	700
Rainbow-Ia	158,000	66
RedGeMSS 128	375,000	35
Picnic	32	34,000
SPHINCS	32	8,000

Table 6: Key and signature sizes of PQC algorithms, summary from Table 3

DNSSEC specific records such as DS or DNSKEY records are higher.

The above findings show that TCP connections are more likely to contain DNSSEC-signed responses compared to UDP connections. It may be the case that TCP is used since DNSKEYS are too large to transport over UDP connections, however, this can not be proven with the current dataset since UDP packets that do not arrive at the resolver are not captured. A TCP-retry is therefore not visible.

6.2.2 PQC modified traffic analysis

To simulate an overnight switch to Post Quantum Cryptographic algorithms to sign DNS responses, the DNS public keys and signatures are swapped for its PQC alternatives. The different algorithms currently in the competition from NIST have different properties regarding their public key and signature size. The sizes for public keys and signatures that are used in this study are given in Table 6.

First, the effects of this substitution on existing UDP traffic is discussed, afterwards the consequences for using PQC keys and signatures in the TCP captured traffic are discussed.

UDP Traffic
In Figure 10 the DNS response sizes are visualized as a CDF for different PQC algorithms on UDP traffic. In this graph, the blue line depicts the current size of all UDP packets in the captured traffic. From the figure, it is clear that each of the different PQC algorithms pushes some DNS responses over the default Ethernet MTU of 1500 bytes. DNS responses larger than the Ethernet MTU will often switch to use TCP connections as discussed in Section 2.3.

To compare the performance of different post-quantum algorithms, the percent of UDP packets that would have to be switched to a TCP connection are listed in Table 7. This table lists the percent of packets that exceed the 1500 bytes payload size. The consequence is that this response will have to be communicated again over a TCP connection, since the resolver is configured such that it will not accept packets that are larger in size

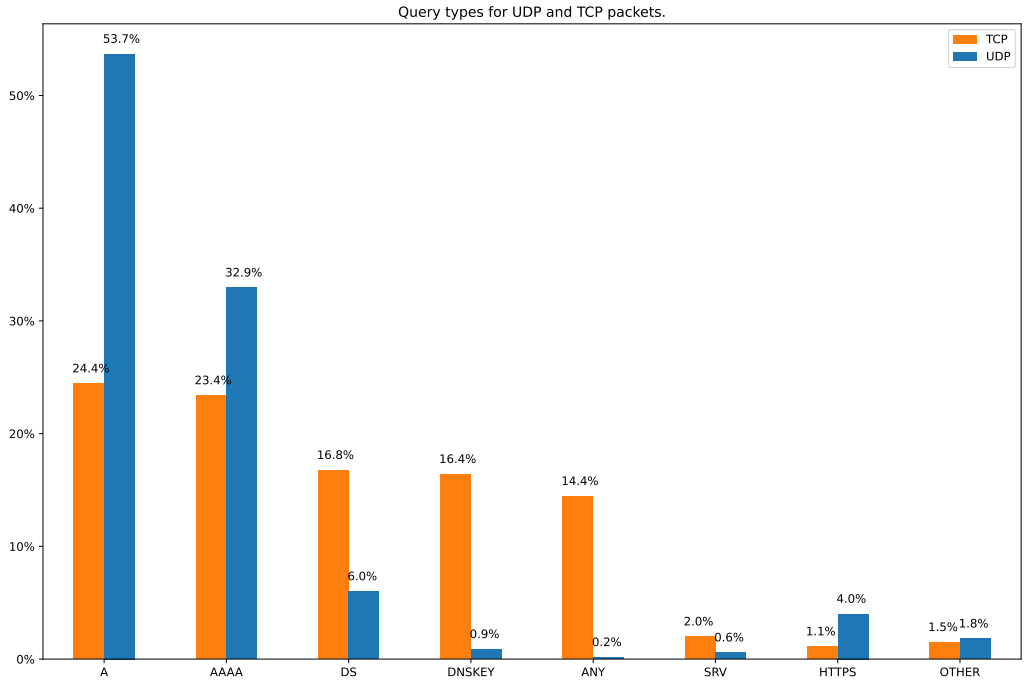


Figure 9: Distribution of query types in TCP and UDP packets.

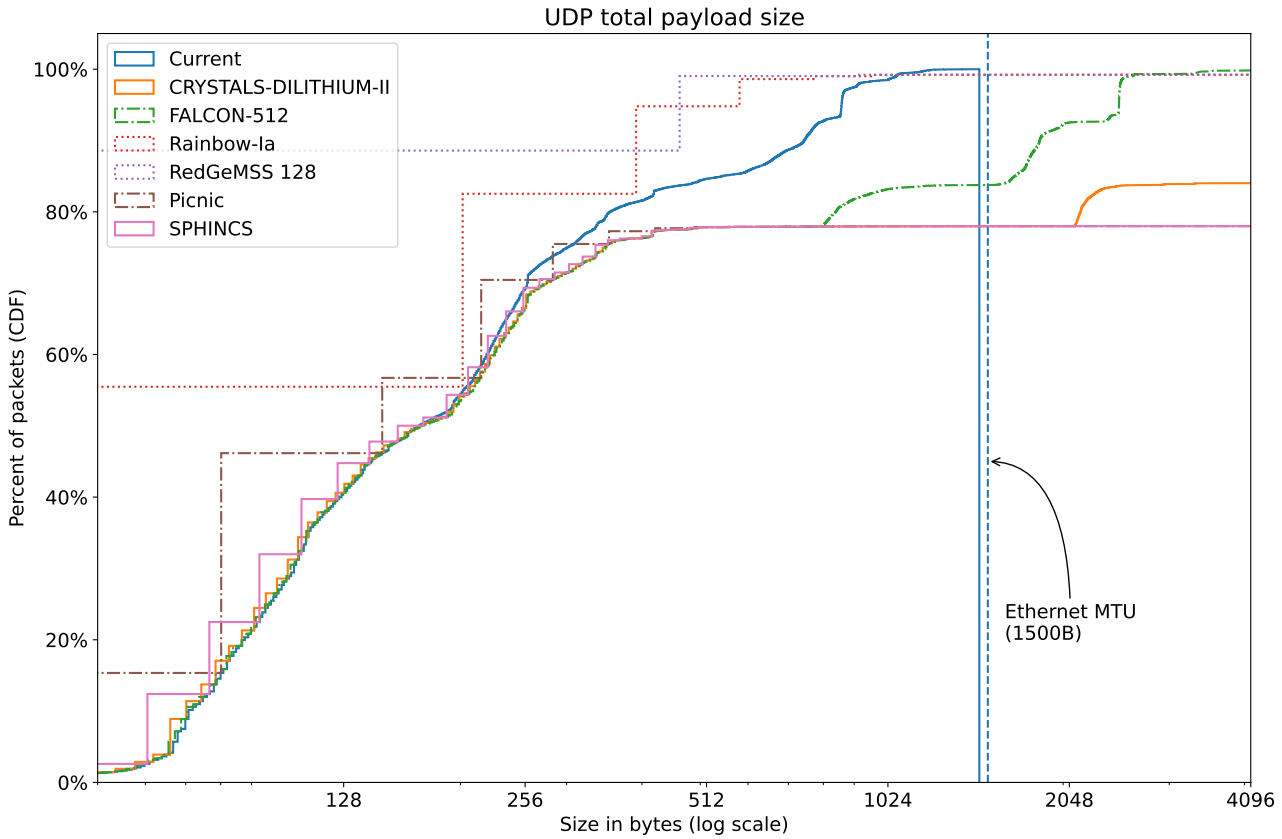


Figure 10: UDP DNS Responses using different PQC algorithms zoomed in.

Algorithm	UDP packets > 1500 bytes	Increase in TCP traffic
CRYSTALS-DILITHIUM-II	22.0%	33.9%
FALCON-512	16.2%	24.7%
Rainbow-Ia	0.8%	0.2%
RedGeMSS 128	0.8%	0.2%
Picnic	22.0%	33.9%
SPHINCS	22.0%	33.9%

Table 7: Effects on existing UDP traffic for different PQC algorithms

Algorithm	UDP packets > 1500 bytes	UDP packets > 64000 bytes	TCP packets > 64000 bytes
CRYSTALS-DILITHIUM-II	22.0%	0.0%	0.02%
FALCON-512	16.2%	0.0%	0.0%
Rainbow-Ia	0.8%	0.8%	24.5%
RedGeMSS 128	0.8%	0.8%	24.5%
Picnic	22.0%	15.5%	80.6%
SPHINCS	22.0%	0.01%	7.5%

Table 8: Effects on UDP and TCP traffic for different PQC algorithms

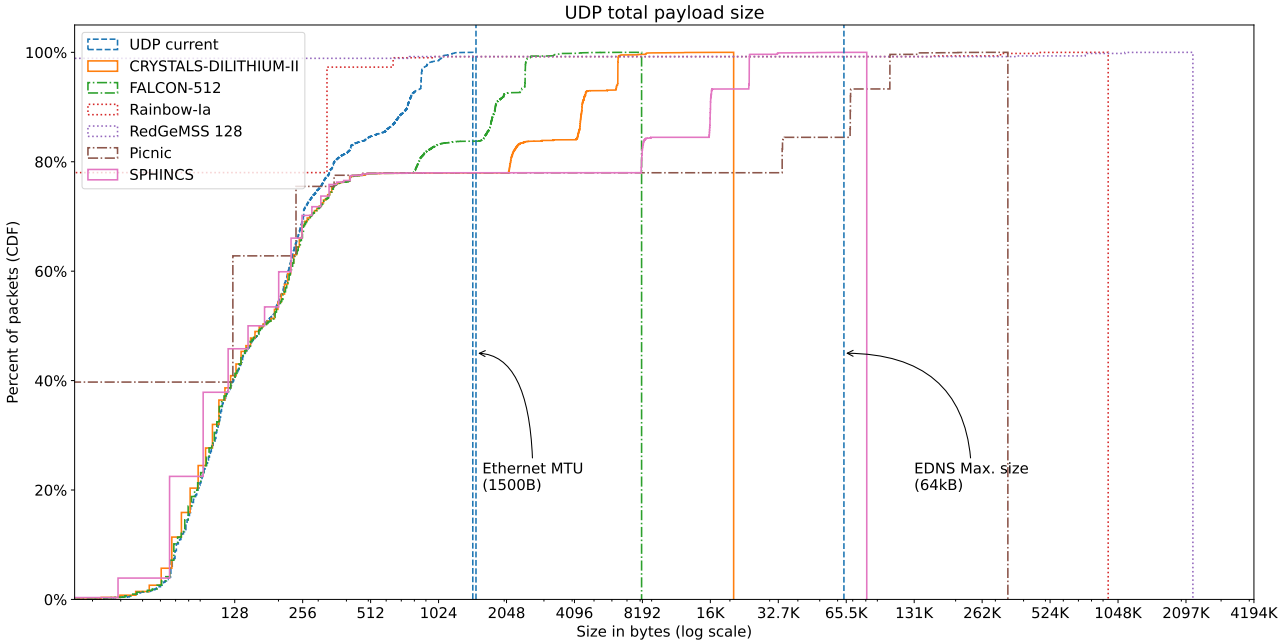
than 1500 bytes. From this table, it could be concluded that Rainbow and RedGeMSS are the best performing algorithms, since the least of the packets would switch to a TCP connection with only 0.8%.

However, when zooming out, some of the largest packets are pushed to over the EDNS maximum size of 64KB for several algorithms. This is visualized in Figure 11a. This is a problem which will be discussed in the next paragraph below.

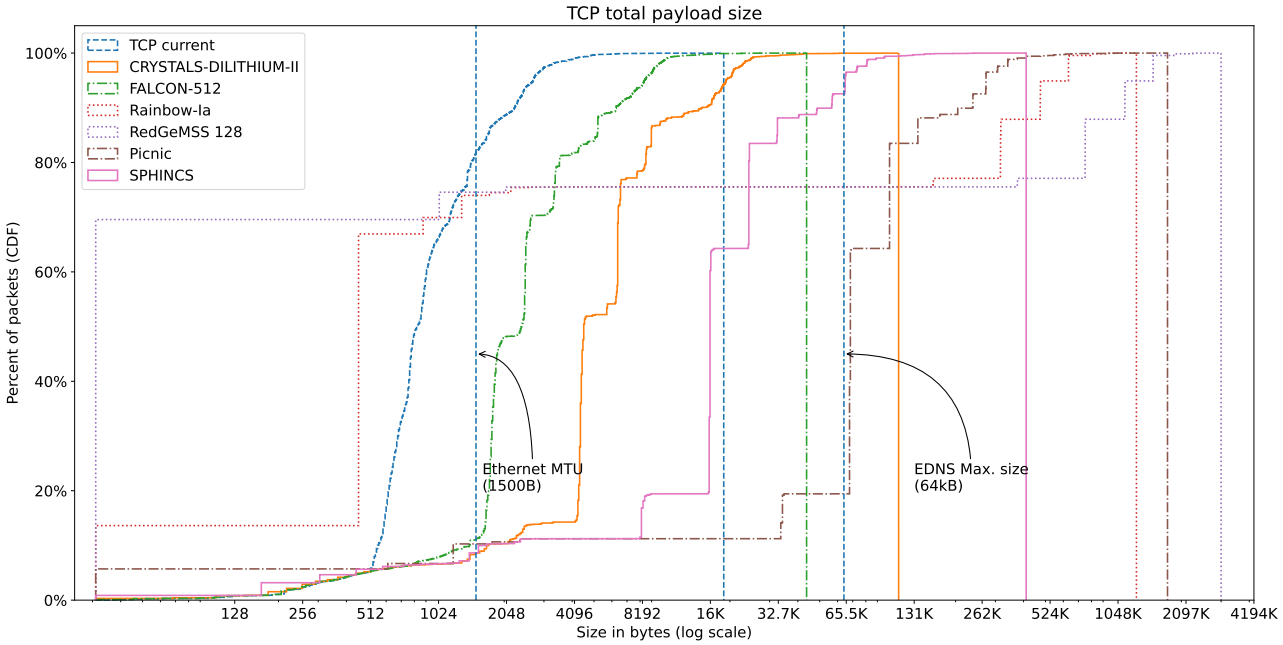
TCP Traffic
The TCP traffic generally involves larger DNS responses compared to UDP traffic because it is used more often for transporting signatures, public keys or other responses that are large in size. This has already been discussed before. It can therefore be expected that the modified TCP traffic is generally of greater size than modified UDP traffic, since signatures and public keys from post-quantum algorithms are larger. In Figure 11b, the packet size as a CDF is plotted for the modified TCP traffic. First, comparing this Figure to the UDP version in Figure 11a, it can be noticed that the TCP traffic does indeed consist of much larger DNS responses than the UDP traffic. Just as UDP responses, also TCP responses grow over the EDNS maximum payload size in Figure 11b. This will be discussed in the following paragraph.

Exceeding EDNS max. size
As discussed in Subsection 2.3, the limit on DNS messages could *in theory* be increased to 64KB. However, as can be seen in Figures 11a and 11b, most algorithms even exceed this limit. This simply means that the DNS response data does not fit in the DNS protocol and can thus not be communicated. Two algorithms that are able to keep current traffic below the EDNS maximum size of 64KB are Falcon-512 and Crystals-Dilithium for UDP traffic. However, for TCP traffic Crystals-Dilithium does also grow over the EDNS limit for at least some packets. Further investigation of these packets shows that these are all attributed to ANY queries to a single domain name. The DNS response for these ANY queries contain 51 signatures and 6 public keys, these are responsible for the increase in size to over $64,000$ bytes. These extreme outliers are deemed not representative for regular internet traffic. When these packets are excluded, also Crystals-Dilithium will stay below the EDNS maximum size limit.

Assuming no significant improvements related to the signature size or public key size of the cryptographic schemes, the only drop-in replacements that can be used are Falcon-512 and Crystals-Dilithium-II. These PQC algorithms do not cause TCP or UDP traffic to grow over the EDNS maximum size.

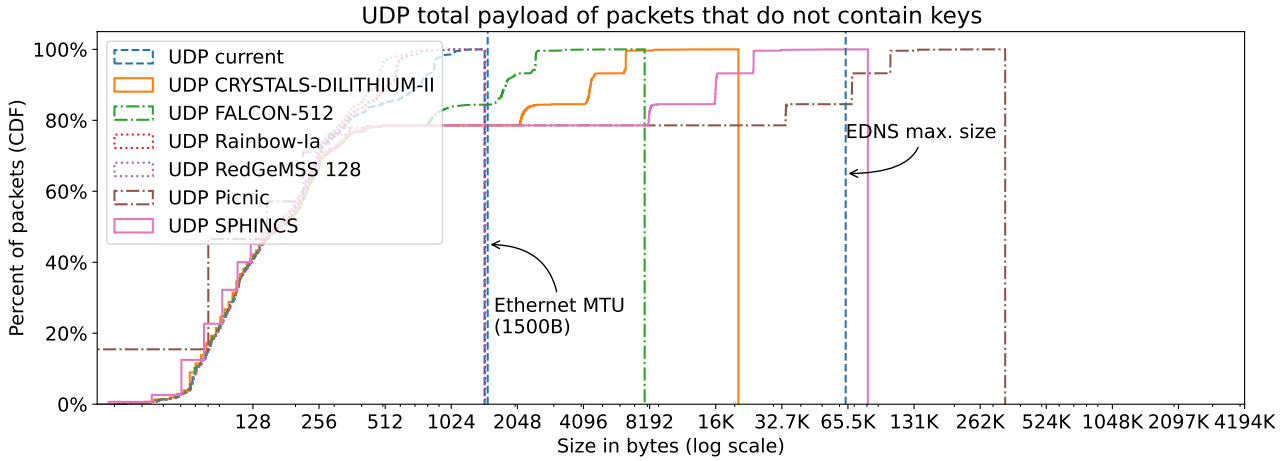


(a) UDP DNS Responses using different PQC algorithms.

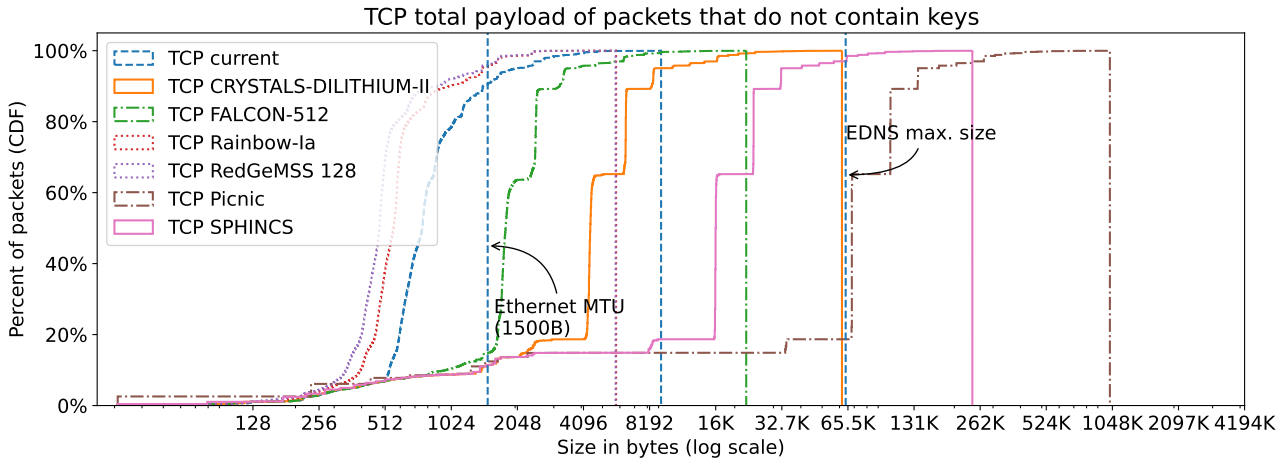


(b) TCP DNS Responses using different PQC algorithms.

Figure 11: TCP and UDP DNS Responses using different PQC algorithms.



(a) UDP traffic that does not contain public keys



(b) TCP traffic that does not contain public keys

Figure 12: Traffic analysis of packets that do not contain DNS public keys

6.3 Filtering traffic

In this section it has become clear that DNS packet sizes might grow larger when new PQC algorithms will be adopted into DNSSEC. Only Falcon-512 is able to keep all traffic below 64,000 bytes, the maximum packet size. Since the proposed algorithms have different properties regarding their signature size and public key size, it is worth researching what exactly is causing the extreme rise in packet size. For a better understanding of the composition of the traffic, several properties of the traffic are investigated. From Table 3 it is already visible that some algorithms have extremely large public keys, like RedGeMSS or Rainbow. These algorithms have public keys that are larger than 64kB. In Figure 12 it now is visible what the consequences are to traffic if these public keys are filtered from the traffic. What can be seen from this, is that the maximum size of the packets drops for most algorithms. Still, Picnic and Sphincs result in the largest packets continuing to exceed the EDNS maximum size of 64000 bytes. This can be attributed to their large signature sizes, having several kB's of signature data. If the need to transport public keys over the DNS infrastructure was not there, more algorithms would be able to be used in DNSSEC. Especially at the start of the PQ age, it might be good to have several alternatives ready to implement in DNS, in case some are found to be broken or not suitable anymore for other reasons.

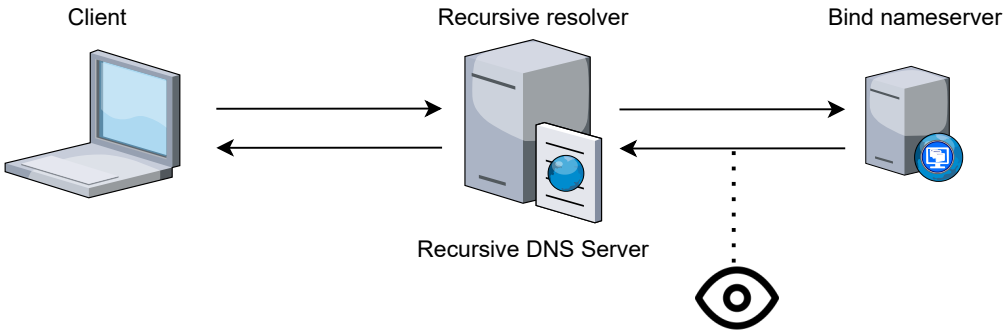


Figure 13: Illustration of the simulated network.

6.4 Benchmarking in practice

In the previous subsection, researching the communication overhead of using PQC algorithms and their signatures in DNSSEC responses has been theoretically approached and analyzed. To test the method that is used and verify the results, a prototype implementation of a popular DNS resolver, Bind[52], is used that has implemented Falcon-512 [53]. The forked Bind 9 software from [52] uses an also forked version of the Open Quantum Safe library in which the Falcon-512 algorithm has been implemented. By creating a simple network using name servers, resolvers and clients, the traffic at the resolver can easily be captured and analyzed as before. The goal is to research the size of DNS responses and see if the theoretically determined results will hold in a simulated experiment. In Subsection 6.4.1 the setup of the experiment is discussed and the outline of the network is illustrated. Afterwards the results are discussed and a conclusion on the accuracy of both theoretical framework and a relation to the experiment is made.

6.4.1 Experiment setup

A template that is already provided by [54] allows for the creation of a network with multiple name servers, resolvers and clients. An illustration of the simple simulated network is depicted in Figure 13. The location of the network capture is the same as how real-world data has been captured, visible in Figure 4. However, since this is a simulated network, the number of name servers and clients that are used is limited. The traffic at the resolver is captured and analyzed as before, revealing the size of the DNS responses that are communicated.

The network that is created using the provided template is simple, using only two name servers, a resolver and a client, as can be seen in Figure 13. One of these name servers is constructed as a root name server, the other as a TLD name server. The name servers and the resolver run the modified version of Bind 9. The DNS name servers signs all traffic using the Falcon-512 algorithm if DNSSEC signed responses are requested by the client. On the resolver, the `tcpdump` utility captures all DNS responses and writes them to a file for later analysis.

The zone files that are used are included in Appendix B. The zonefile of the root name server in Listing 3 shows that the root zone only has a single reference to the top level domain `.tld`. This is the only TLD that is used in this experiment. The zone file of the .tld domain is listed in Listing 4. In this zone file, there are a few entries for some of the most popular RR types. The zone files in the Appendix do not show `DS` or `DNSKEY` records, these are created by the Bind software.

The client runs a simple bash file containing `dig` commands to query information from the name servers through the resolver. To provide an accurate approximation of the distribution of packet sizes of DNS responses in this experiment compared to the captured real world traffic, the bash file containing all dig commands is constructed as follows.

For every DNS response in the original file discussed in Section 5.5, a line is added to the bash file which is executed on the client. These lines all have the following construction:

```
dig [RR type] @172.20.0.2 [+dnssec] [+[no]tcp]
```

In Table 9 the options in brackets are described and an explanation of how and when these are used is given.

Option	Usage
RR type	The Resource Record type that is queried in the original packet capture.
+dnssec	Requests DNSSEC records be sent by setting the DNSSEC OK bit (DO). If signatures are seen in the original packet capture, indicating the client had set the DO bit, the DO bit will be set in this experiment. Otherwise it will default to not request signatures.
+[no]tcp	Do [not] use TCP to query the resolver. When UDP responses grow too large, the request will be retried over TCP. The +notcp flag is set for every UDP response.

Table 9: Flags used in the dig command on the client.

The IP-address (172.20.0.2) is that of the resolver in this network.

6.4.2 Results

In Figure 14 the UDP and TCP DNS response sizes are plotted in the same format as used before in this paper. First, it immediately stands out that the real Falcon traffic in green does not by far reach as far as the predicted traffic in orange in both Figures 14a and 14b. For UDP, this is expected since UDP traffic that exceeds the Ethernet MTU of 1500 bytes will switch to use TCP instead. So it is expected to have no packets of sizes larger than 1500 bytes in Figure 14a. However, further discussing Figure 14a, the real Falcon traffic does not come close to the Ether MTU limit with only a maximum of 797 bytes before switching to TCP. This could be explained since the zone file that is used is not as big as zone files of TLD operators, consequently lowering the variation in content and thus variation in response sizes of the captured traffic.

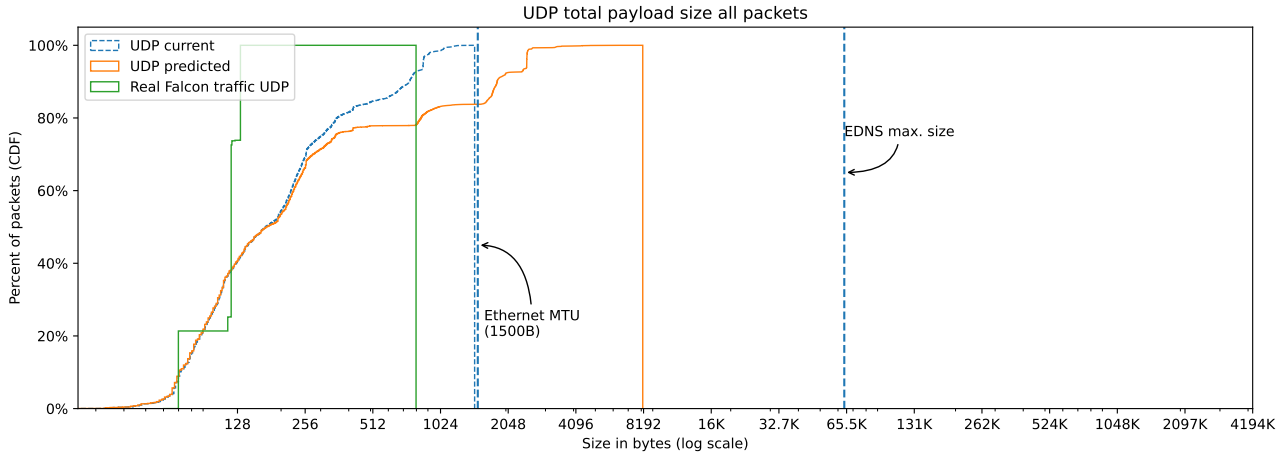
The TCP traffic in Figure 14b has, just as the UDP traffic, not much variation in traffic sizes. This too, is the result of the small zone file and the variety in content that it lacks. The falcon traffic also lacks the tails at the beginning and at the end of the graph. DNS responses below the Ethernet MTU size are communicated over UDP and only have a maximum size of 797 bytes in this experiment. The tail at the other end is not there since there are no packets that large in the traffic, which is expected.

The overall trend in TCP traffic is that it is somewhat larger than current TCP traffic, although it does not seem to follow the predicted TCP traffic. The network that is build and its zone files are simple, without many references or variation. This has as a consequence that lots of the same requests and responses are communicated, resulting in straight lines and large steps, not as smooth as the current or predicted traffic graphs.

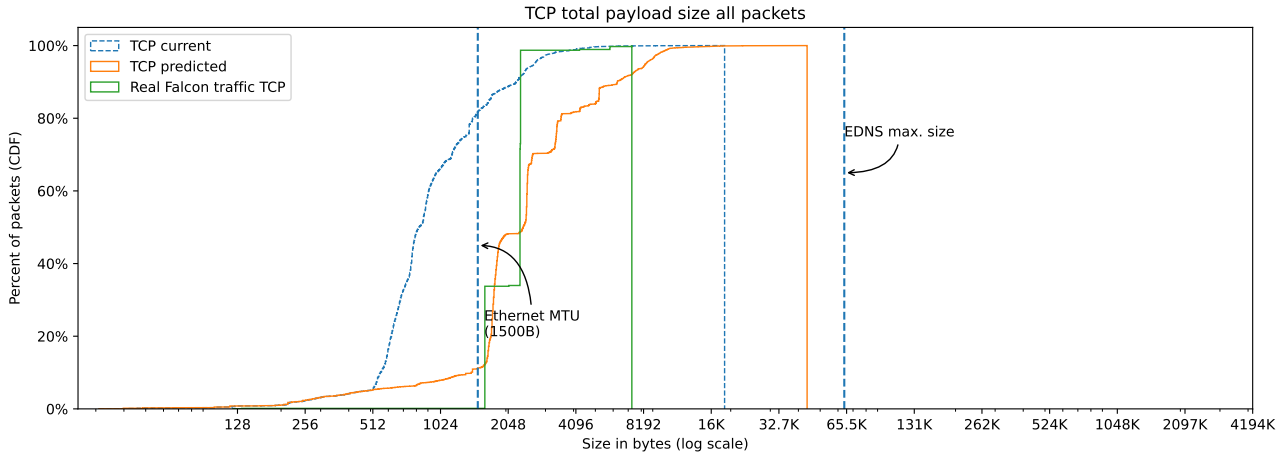
For the purpose of checking the theoretical method and results this suffices to say that it cannot be used to approximate the traffic. A similar experiment could be done using a larger infrastructure with more name servers that have larger zone files. Then the straight lines would likely disappear because the traffic will have a higher variation of packet size. What is shown from this experiment is that the number of signatures and public keys in the packets are the determining factor in packet size, especially for small zone files that have little variation in contents. This is visible by the large steps in the figures. This is expected as Falcon-512 signatures are 700 bytes and its public keys 900 bytes. None of the records in the zone file reach these sizes.

6.5 Concluding communicational overhead

In this section, it is shown that DNS responses grow in size when Post-Quantum Cryptographic algorithms are used instead of currently used algorithms in DNSSEC. In Table 8 some metrics related to traffic size are listed. PQC alternatives that have large signatures or public keys create DNS responses that can grow over the EDNS maximum size. Falcon-512 and Crystals-Dilithium-II are the only two viable algorithms that do not cause DNS responses to grow over the EDNS maximum size of 64kB when extreme outliers in the traffic are excluded. Besides traffic growing over the EDNS limit, UDP packets will also exceed the MTU limit, causing UDP packets to utilize the more resource expensive TCP connections. With no changes to the DNS protocol, only Falcon-512 and Crystals-Dilithium-II can be used based solely on the communication overhead.



(a) Predicted and real Falcon traffic over UDP



(b) Predicted and real Falcon traffic over TCP

Figure 14: TCP and UDP DNS Responses simulated with Falcon-512.

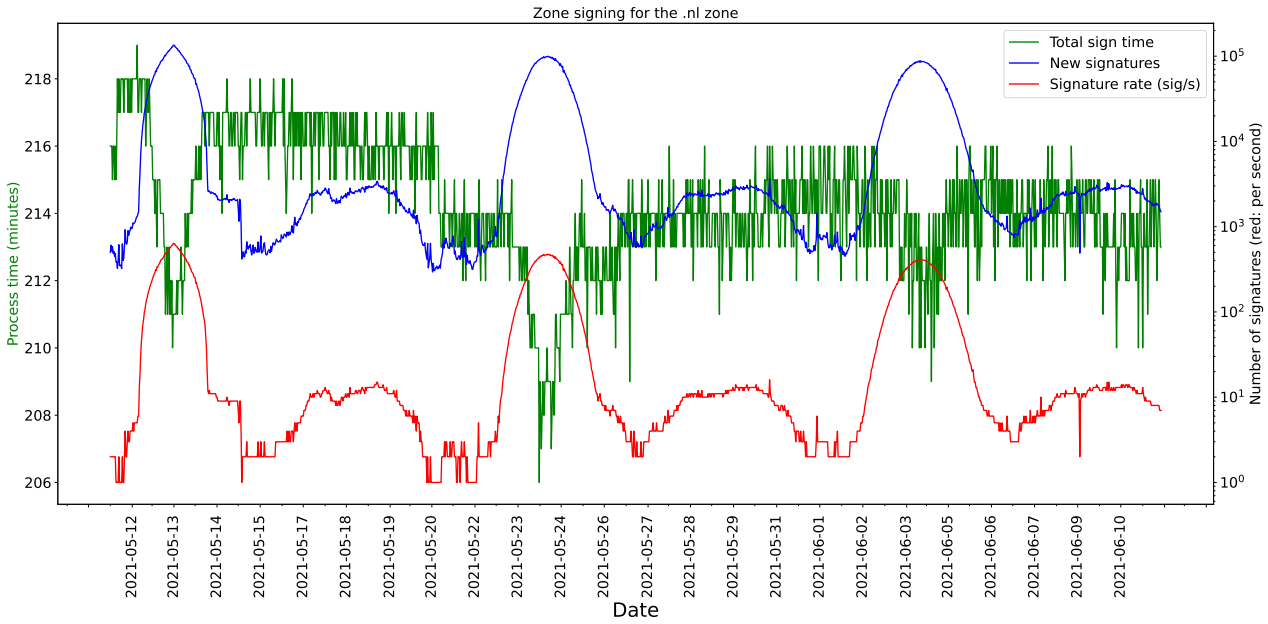


Figure 15: Signing statistics for signing the .nl zone file including sign time.

7 Computational load of PQC on DNS operators

Besides an effect on the communicational overhead discussed in the previous section, adopting new PQC algorithms will also have effects on the computational load on DNS components like name servers and resolvers. To investigate the effects that different PQC alternatives have on the performance of these DNS operators, two analyses are made. First, a data analysis is made on log information that is acquired from SIDN, the organisation that manages the .nl domain and its name servers. This log file contains information on the performance of the SIDN name servers that are responsible for signing the zone file. Second, the already gathered data discussed before and in Subsection 5.5 is analyzed for DNS resolver performance. Using this information, a theoretical estimation of the expected consequences on the computational load is defined for both resolvers and name servers.

7.1 Zone signing

Using a log file containing statistics of the signer part of the name server, the load on the name server is computed for signing the whole .nl zone. In Figure 15 the time that the name server needs to sign the whole zone is depicted in green, together with the signatures that have to be created (blue) and the signature rate in signatures per second (red) indicating at which speed the signatures are created. What stands out is the relation between the signature rate and the number of signatures that have to be created. Once many signatures have to be created, the signature rate rises and if there are less signatures to be created, the speed at which this happens declines. The large peaks occurring every 9 days in the Figure indicate total zone signs where in the order of 10^5 new signature are created. Signing the zone file takes between 208 and 218 seconds depending on how many signatures have to be created.

The process in which signatures are created is part of a process of keeping the zone file up to date. For the .nl zone, SIDN signs its zone file every 30 minutes. In Figure 16, the time it takes to fully run the process of updating the zonefile is plotted, where the blue and red line again correspond to the number of signatures and the signature rate respectively. The green line in this Figure now indicates the total processing time, i.e. the complete time from start to finish of the maintenance process. This includes actions like compiling the zone file, checking validity and moving it around to the right processes and creating the signed zone file, among other things. What is visible in this Figure, is that the total processing time is not related to the number of signatures that have to be placed. The process always takes between 16 and 17 minutes to complete. This indicates that

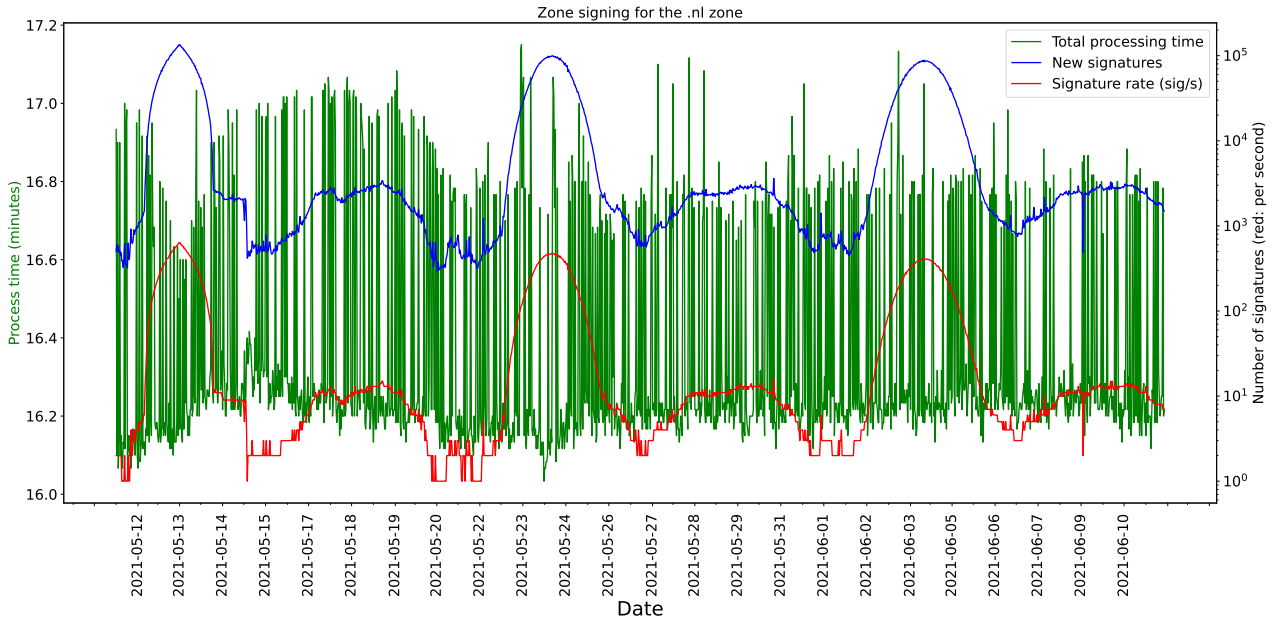


Figure 16: Signing statistics for the .nl zone file including total processing time.

the signing part of the whole process, with only 3.5 minutes, takes little time in the whole process and does not influence the total processing time significantly.

7.1.1 Effect of PQC algorithms on zone signing

To study the effects of different algorithms on the process of signing the zone file, an approach based on the cost in CPU cycles for different algorithms to sign the zone file is taken. Sign and verify cycles from different algorithms are illustrated in the matrix in Figure 17. In this Figure, the time in CPU cycles for both signing and verifying signatures using different algorithms are depicted on respectively the x-axis and y-axis. By calculating the CPU cycles that are currently necessary to sign the signatures in the given time, it is then possible to calculate the CPU cycles that are available to sign the zone. In the next paragraph the method is explained more detailed. Afterwards, the process of validating this model and the results are discussed.

Method
Signing a zone requires that the resolver spends many of its CPU cycles signing every entry in the zone file. It is safe to assume that the current name server infrastructure is handling this well since the zone file is signed successfully every 30 minutes. The current number of CPU cycles that the server is busy creating signatures can be calculated from the SIDN log files. In these log files the new number of signatures that are created and the time it takes to perform this are listed. For clarity this is listed as a formula in Equation 1, for $i = j =$RSA/SHA-256, the results are visible in Figure 18 and will be discussed in the next paragraph.

$$P_{j,t} = \frac{n_t * S_j}{c_t * S_i} \tag{1}$$

Where

n_t	number of signatures that have to be created at time t (from SIDN log file)
c_t	Signatures created per second at time t (from SIDN log file)
S_{algo}	Sign cycles to create one signature using algorithm $algo$
$P_{algo,t}$	Performance for algorithm $algo$ at time t

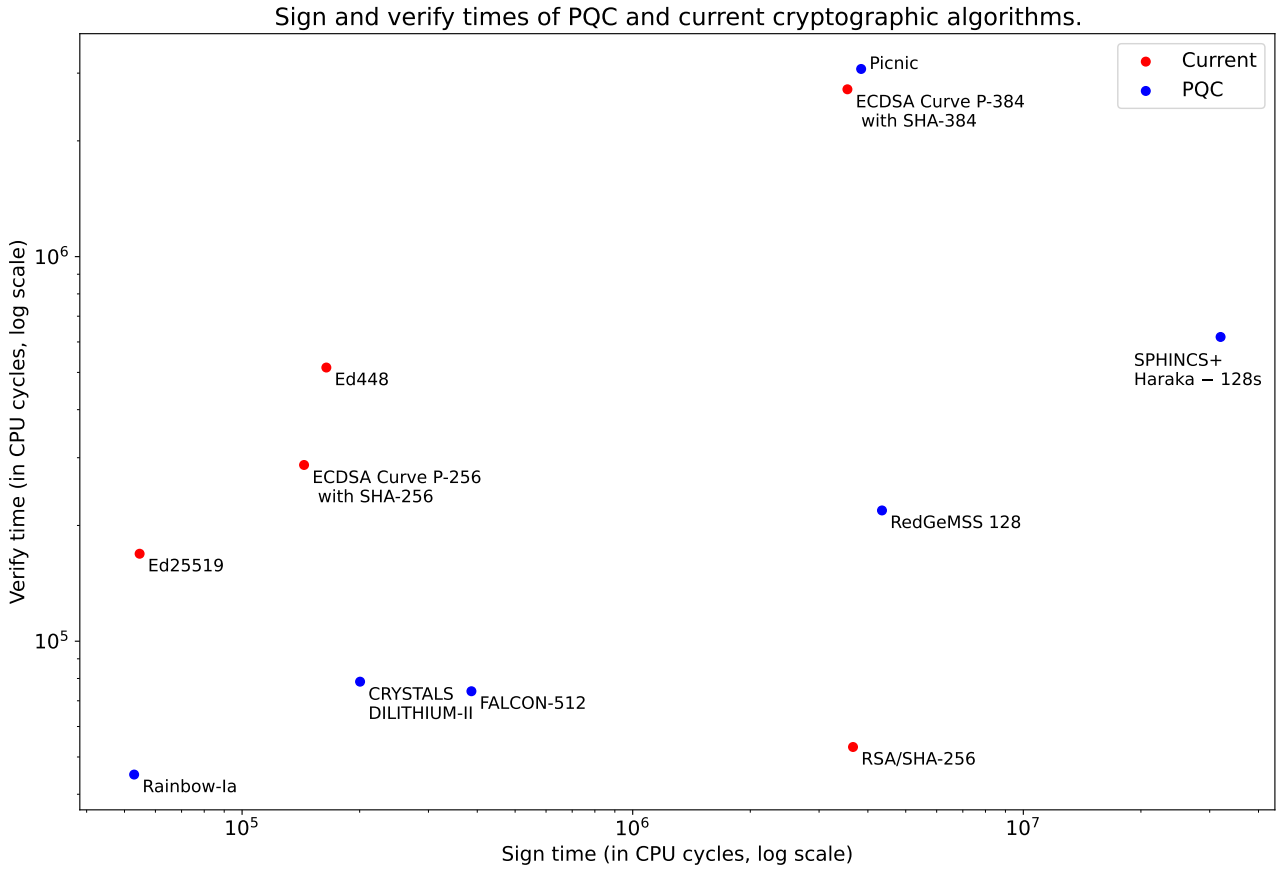


Figure 17: Comparison of signing and verifying times for current and PQC algorithms, data from [55]

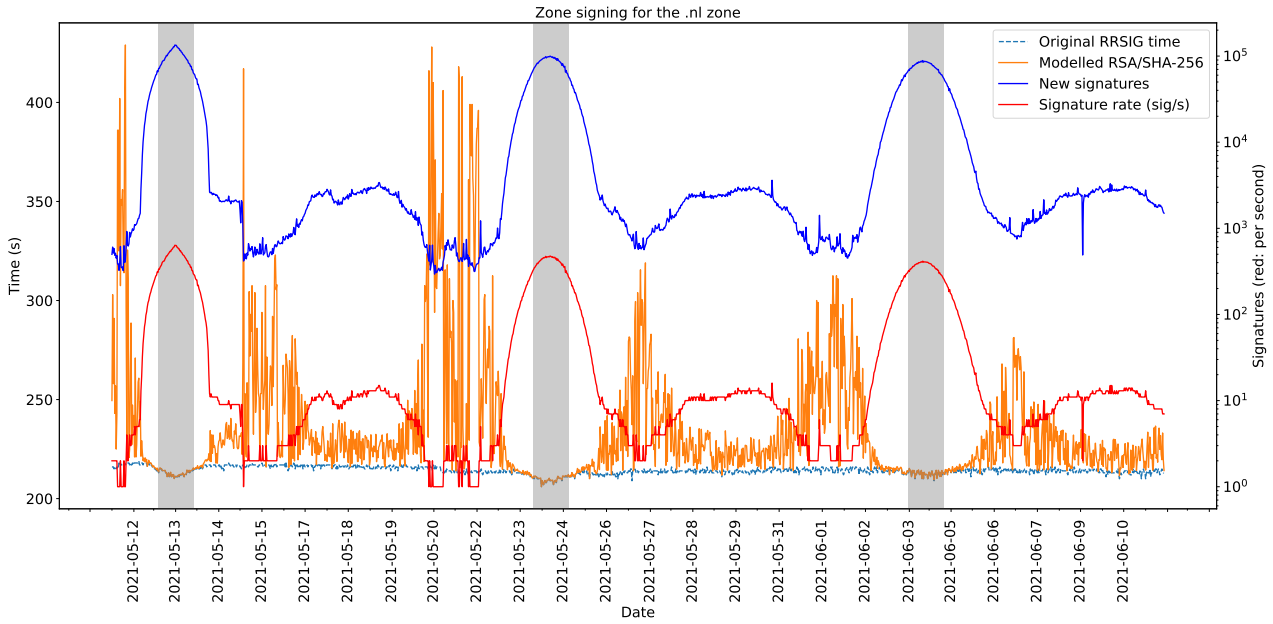


Figure 18: Model for signing the `.nl` zone for RSA/SHA-256

Method verification

To verify the correctness of the method that is used, the modelled RSA/SHA-256 performance that is calculated using the method above can be compared to the actual time it takes to sign the zone file using RSA/SHA-256. In Figure 18 the resulting modelled performance of signing the zone file using RSA/SHA-256 is depicted, together with the performance of the name server retrieved from the SIDN log files. In this Figure, the modelled RSA/SHA-256 performance in orange approaches the actual performance in dashed blue best at peak performances. The most accurate prediction by the model is when the most signatures have to be created and the signature rate is at its highest, indicated with grey areas. At other times, the signature rate is too low to accurately estimate the correct signature time. Since it is not known what the threshold or the reason is for the name server to scale up, it is only possible to give an estimation on the total sign time when the most signatures have to be created. From this, it can be concluded that the model can be used to estimate the duration when most new signatures have to be placed, i.e. when signing the whole zone file. Signing the whole zone file is a worst-case scenario in terms of how many signatures have to be created and consequently how long this might take. Estimating the duration of this worst-case scenario is sufficient to determine what PQC alternatives can be used to sign the zone file. If a worst-case scenario is fast enough, then any other scenario that require less signatures to be created are likely to succeed within the required time as well.

Results

Similar to SHA/RSA-256 in Figure 18, the performance of other algorithms are depicted in Figure 20. Just as in Figure 18, the grey bars indicate where the most new signatures have to be created and where the model is most accurate. For a proper comparison, these moments of signing the whole zone are plotted in Figure 19 for every algorithm. From this, Rainbow performs best with only 0.05 minutes, which is just 3 seconds to sign the whole zone based on the cycles that are necessary. The worst performing algorithm is Sphincs+-Haraka which needs almost 30 minutes to sign the whole `.nl` zone.

Recall that zone signing occurs every 30 minutes and the whole process currently takes at most 17 minutes. The actual signing part of this takes 3 to 4 minutes. Except for Sphincs+-Haraka, which takes almost 30 minutes for just the signing part, all other benchmarked algorithms would be able to replace RSA/SHA-256 and still sign the whole zone in less than 30 minutes. This holds with the assumption that key management and possible calculations as preparation for running algorithms do not take a longer time or are computationally more expensive than the currently used algorithm. Of all PQC algorithms, Rainbow performs best with only 3 seconds to sign the whole `.nl` zone file.

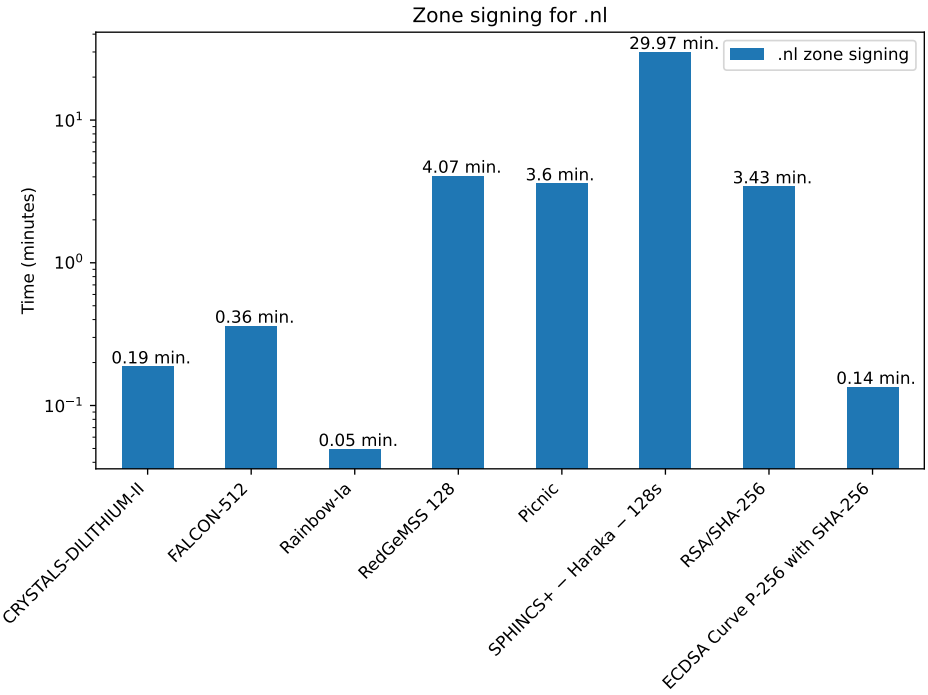


Figure 19: Time necessary for signing the whole `.nl` zone using PQC algorithms.

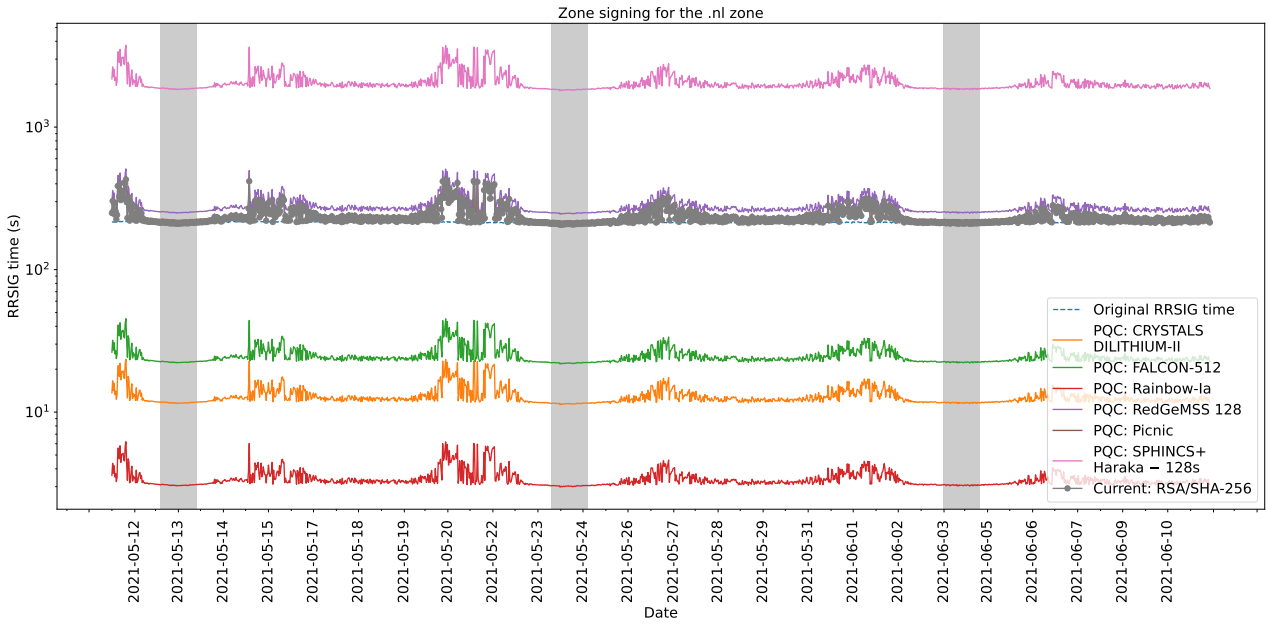


Figure 20: Model for signing the `.nl` zone for PQC alternatives.

Algorithm number	Algorithm name	# of signatures	% of signatures
1	RSA/MD5	753	0,0%
3	DSA/SHA1	762	0,0%
5	RSA/SHA-1	76115	1,4%
6	DSA-NSEC3-SHA1	777	0,0%
7	RSASHA1-NSEC3-SHA1	195672	3,6%
8	RSA/SHA-256	4424634	82,2%
10	RSA/SHA-512	36980	0,7%
12	GOST R 34.10-2001	136	0,0%
13	ECDSA Curve P-256 with SHA-256	637471	11,8%
14	ECDSA Curve P-384 with SHA-384	7793	0,1%
15	Ed25519	2178	0,0%
16	Ed448	724	0,0%
253	private algorithm	1	0,0%

Table 10: Number of signatures for all algorithms in the data capture.

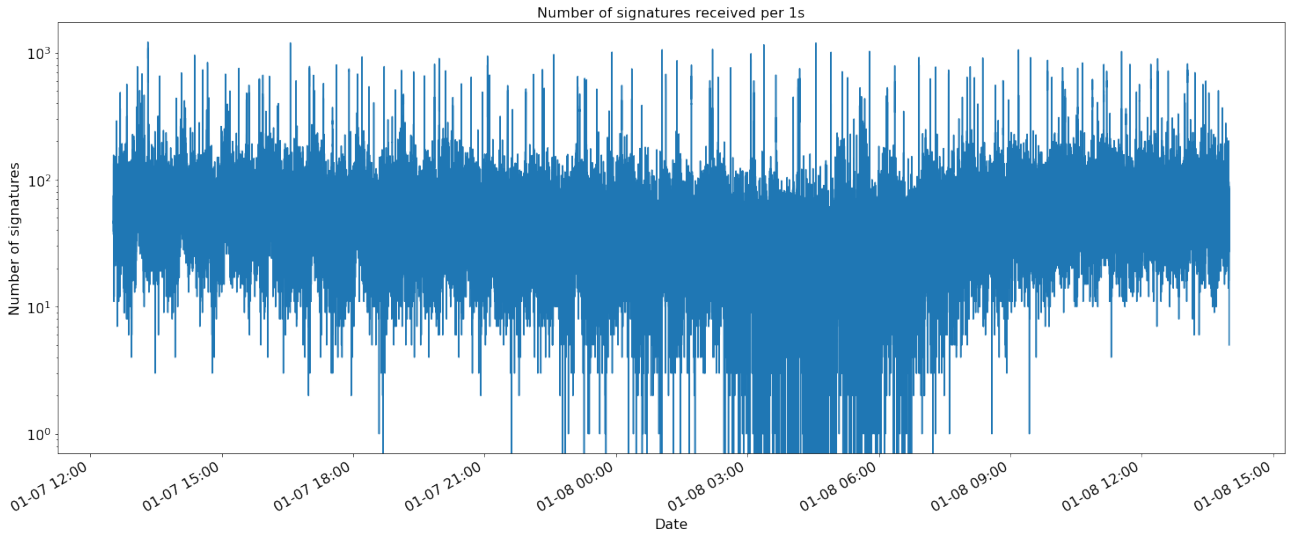
7.2 Resolver signature verification

Signing the zone file at the name server could be really fast using Rainbow, however, if the verification of these signatures takes extremely long for resolvers it would still not be a useful replacement for current algorithms. Large resolvers process thousands of DNS requests per second and if DNSSEC is enabled, these signatures need to be verified *on-the-fly*. To keep DNS resolver operators' operational impact to a minimum, the verification time per request using PQC algorithms must not significantly exceed the time that the resolver is busy verifying signatures using current algorithms.

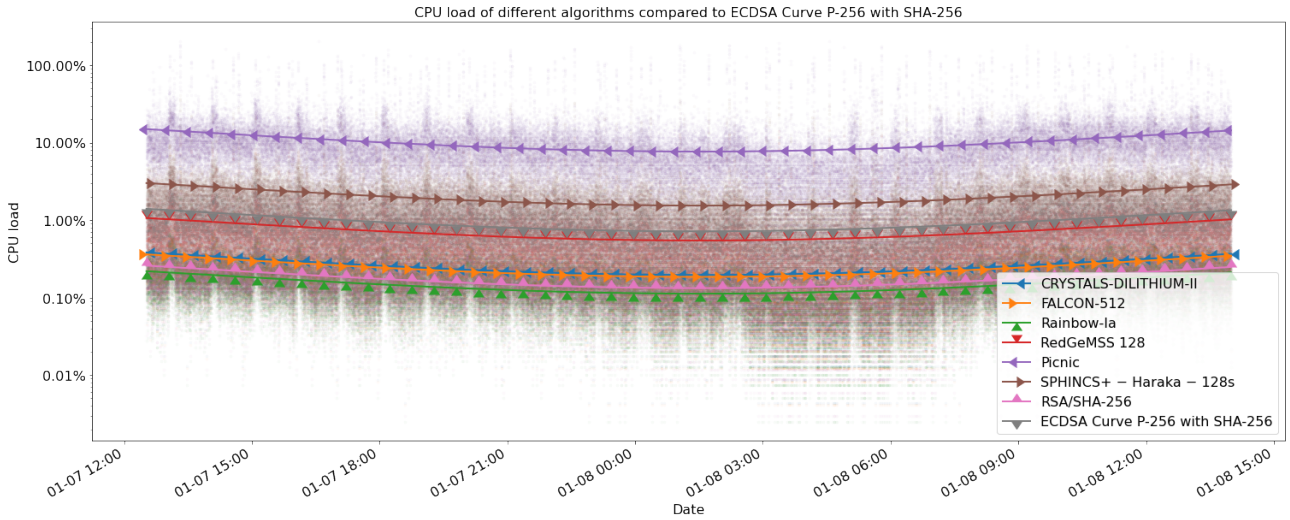
To research the effects of verifying signatures using PQC algorithms on the CPU load of DNSSEC enabled resolvers, a similar method to the one in the previous section is used. Using the verification cycles of different algorithms depicted in Figure 17, the CPU usage is calculated for current and PQC algorithms. This is done by calculating how many cycles are used currently to verify all signatures that arrive at the resolver. By calculating the amount of cycles that are necessary every second and using the system specifications listed in Section 5.5, the CPU load can be calculated. Based on the number of received signatures which is plotted in Figure 21a, the CPU load of different algorithms is depicted in Figure 21b. A best fit line is also drawn in this Figure to compare the performances of different algorithms easily. In these calculations it is assumed that all received signatures are verified. In reality, this is not always the case as signatures can be cached or skipped entirely, as discussed by van Rijswijk-Deij et al. in [40]. The consequence of this simpler approach is that the calculated CPU load is higher than can be expected in reality. To get a more realistic estimation instead, the model from [40] could be used to determine the number of signatures that are verified at the resolver. However, in this study the DNS responses that are captured can not provide a good estimate of the signatures that are verified at the resolver using the model from [40] since the initial DNS queries are not included in the data capture. Hence, the calculations in this section could be seen as worst-case scenario and upper bound to the performance of a resolver in case all signatures are verified.

7.3 Resolver effects

From Figure 21b it can be concluded that there are large differences in verifying performance on the resolver. The Picnic algorithm performs worst with a CPU load that often lies between 10% and 20% according to the best fit line that is drawn in the Figure. For more detailed view of the performances of Picnic and other algorithms, the mean, median and maximum CPU load are listed in Table 11. From this Table, it can be seen that verifying signatures using Picnic results in a maximum CPU load of over 200%. Besides being not able to handle all signature verifications on its maximum capacity, Picnic has the highest mean and median CPU load on average. Putting a load on the CPU for just verifying the signatures of already 10% on average. Since verifying signatures is often not the only task of a DNS resolver and more processes could be running on the same machine, this could mean that the performance of the resolver decreases. This makes it highly likely that many resolvers would have to scale up their computing power to be able to serve the current amount of DNS queries for its connected clients when Picnic as an PQC alternative would be implemented. Two of



(a) Number of signatures arriving at the resolver per second



(b) Computed CPU load of verifying signatures on a DNS resolver.

Figure 21: Resolver load

Algorithm	CPU load mean	CPU load median	CPU load max
CRYSTALS-DILITHIUM-II	0.26 %	0.20 %	5.27 %
FALCON-512	0.24 %	0.19 %	4.98 %
Rainbow-Ia	0.15 %	0.12 %	3.02 %
RedGeMSS 128	0.71 %	0.57 %	14.70 %
Picnic	10.03 %	8.03 %	206.60 %
SPHINCS+ - Haraka - 128s	2.02 %	1.61 %	41.52 %
RSA/SHA-256	0.17 %	0.14 %	3.57 %
ECDSA Curve P-256 with SHA-256	0.94 %	0.75%	19.29 %

Table 11: CPU load of different algorithms on a DNSSEC enabled resolver.

the best performing PQC algorithms compared to RSA are Rainbow and Falcon. Rainbow even results in a small decrease of CPU load. An algorithm that is gaining adoption in DNSSEC is the ECDSA Curve P-256 with SHA-256, or ECDSA-P256 for short. In Table 10 it is visible that almost 12% of all signatures are created using this algorithm. ECDSA-P256 increases the maximum CPU load to almost 20% but this can be handled by resolvers as is discussed by Rijswijk-Deij et al. in [40]. Taking into account that this algorithm is already being deployed and adoption will likely increase, it is clear that also RedGeMSS 128 could be adopted as a PQC alternative since RedGeMSS has an maximum CPU load of under 15%, less than ECDSA-P256.

Results
The currently most used algorithm, RSA/SHA-256, performs well with an average CPU load of 0.17% and only 3.57% on the busiest moments. Additionally, ECDSA-256 already makes up for almost 12 percent of all signatures and although it increases the CPU load compared to RSA/SHA-256, it has been proven to run on a DNS resolver and the adoption will likely increase in the future[40]. A significant increase in computational capacity is needed at the resolver before algorithms Picnic or Sphincs could be used in DNSSEC. The resolver would otherwise not be able to function properly under the currently expected CPU load.

Thus, the best algorithm based on DNS signature verification load on the resolver would have the least CPU load. This makes Rainbow-Ia, Falcon-512 and Crystals-Dilithium-II the most viable candidates since these will not increase the CPU load by much compared to RSA/SHA-256. Also RedGeMSS 128 could be considered since this is performing more efficient compared to the currently growing ECDSA-256 algorithm.

From Figure 8, it is clear that a little over 22% of all traffic is signed. According to a DNSSEC deployment report found at [56], currently 91% of all TLD's are signed in the root zone, but only 5% of all second level domains are using DNSSEC. This leaves enough room for growth and the question then arises whether PQC adoption in DNSSEC would not create another hurdle for the general adoption of DNSSEC. Based on the results found in this study, the Rainbow algorithm performs even better than RSA/SHA-256 with a maximum load on the CPU of only 3%. This would mean that, based only on computational load at the resolver, Rainbow would not result in additional hurdles for the adoption of DNSSEC based on additional computational loads. There could be other barriers for adoption more related to each algorithm, i.e. key management or key rollovers. This is not researched in this study.

7.4 Concluding computational load

In this section, the load of different PQC algorithms on DNS name server and DNS resolver machines is studied. By combining the results of the research on the name server load and resolver load, the best performing algorithms can be chosen that have the least amount of additional computational overhead. Choosing a PQC alternative as successor to current algorithms that does not need name servers or resolvers to increase their computational capacity would be preferred. Other algorithms would cause DNS zone operators and DNS resolver operators to upgrade their machines, this would lower the incentive and speed of adoption of new PQC algorithms.

In Table 12 an overview is given of the results that are discussed in this section. Based on signing performance of a DNS name server of the zone file and the verification performance of a DNS resolver, there is one PQC alternative that performs best, having the least impact on computational load at the name server as well as

Algorithm	.nl zone sign time (seconds)	Max. resolver CPU load
CRYSTALS-DILITHIUM-II	11.3	5.27%
FALCON-512	21.7	4.98%
Rainbow-Ia	3.0	3.02%
RedGeMSS 128	244.4	14.70%
Picnic	216.1	206.60%
SPHINCS	1798.3	41.52%
RSA/SHA-256	206.0	3.57%
ECDSA Curve P-256	8.4	19.29 %

Table 12: Computational load of PQC algorithms on DNS infrastructure

the resolver, namely Rainbow-Ia. Two other PQC algorithms, Crystals-Dilithium and Falcon-512, do perform better than RSA/SHA-256 at signing the .nl zone file, but increase the CPU load on the resolver for verifying signatures, although only slightly. However, this increase is still below the performance of the currently used ECDSA-P256 algorithm.

8 Addressing challenges of PQC + DNSSEC

From the previous sections it can be concluded that there is no one perfect replacement for current algorithms used in DNSSEC. Rainbow performs very good based on the computational load at both resolver and name server, but its public keys are too large to communicate over the existing DNS infrastructure. Crystals-Dilithium and Falcon-512 perform average on the computational side but a large shift in UDP traffic to TCP traffic is expected based on the communication overhead discussed in Section 6.

In this section, out-of-band key exchange methods are researched. An out-of-band key exchange is an exchange of keys outside of the current communication channel, in this case outside of the DNS infrastructure. For example, an out-of-band key exchange over an HTTP connection. At the moment, all DNS related material like signatures and public keys are transported over the DNS infrastructure. However, if large DNSKEYS could be placed outside this DNS system, it could make other cryptographic alternatives that have larger signatures or keys more easily adoptable into DNSSEC. This could be a good alternative for algorithms like Rainbow or RedGeMSS that have public keys of respectively 158 kB and 375 kB. If other aspects of these cryptographic algorithms are favorable over other algorithms, an out-of-band key exchange method could resolve the issues regarding the communication of large public keys.

8.1 Problem statement

As shown in the previous sections, Rainbow is a preferred PQC alternative to adopt in DNSSEC for its low CPU usage and fast zone signing. If its large public keys can be communicated outside of the DNS infrastructure, a decrease in packet sizes can be seen in Figure 12. To compare this to the situation depicted in Figure 11, the largest DNS responses would grow to over 1048 kB for the Rainbow algorithm if keys are included in the DNS responses. If these public keys are not communicated in DNS responses, Figure 12b shows that the packet sizes stay below 8192 bytes. This is smaller than the maximum observed message sizes in the current TCP traffic. Then Rainbow performs very well, sometimes even better than current algorithms. Together with the computational load that is extremely low, its only disadvantage are the big keys that have to be communicated.

To research what the effects are of using out-of-band key exchange in DNS resolving, a DNS resolver written by the NLnet Labs team, Unbound [57], is adapted to perform out-of-band key exchange using different transport protocols. Unbound is chosen for this study as it can only be used as a resolver, unlike Bind [58] it has no features to act as an authoritative server. The lack of this feature likely decreased the complexity of the software, making it easier to develop in Unbound. Additionally, its developers can relatively easy be reached for any questions through the contacts that the authors' supervisors have with its developers.

In a presentation for DNS-OARC 33 by Petr Špaček [59], several resolvers are benchmarked using a new measurement tool. This presentation shows that Unbound performs well and can handle the most clients of all other tested resolvers. Additionally, compared to the other resolvers that are experimented with, Unbound also performs best when using other protocols than UDP and has a low performance penalty compared to its UDP performance. However, the embedded Python program that is added in this study to provide out-of-band functionality causes such a large overhead compared to the low-level C implementation of Unbound itself that these benefits of Unbound over other resolvers can be disregarded. Any performance speed that is gained as a result of using Unbound are minimal. As such, similar results are expected if the Python implementation is added to other resolvers. Using Unbound in this study instead of other DNS resolver software will thus likely not have an effect of the conclusions of this study.

In the following subsections the Unbound software is described and the changes to the software that are necessary to test this are outlined. Afterwards, the different methods of key exchange and the way these are measured are discussed. At last, these methods are compared and a concluding answer will be given if out-of-band key exchange is a viable alternative to exchanging DNSKEYS.

Since resolvers using out-of-band key exchange would have to perform more work to fetch all necessary information, it is expected that the time it takes to resolve a domain name to an IP address increases. So, a viable alternative is allowed to increase the domain resolving process, however, it should not increase the process of resolving a domain name significantly. This study defines a maximum allowed overhead by looking at the page load time of web pages and the role that DNS plays. Based on [60] and [61] a web page should

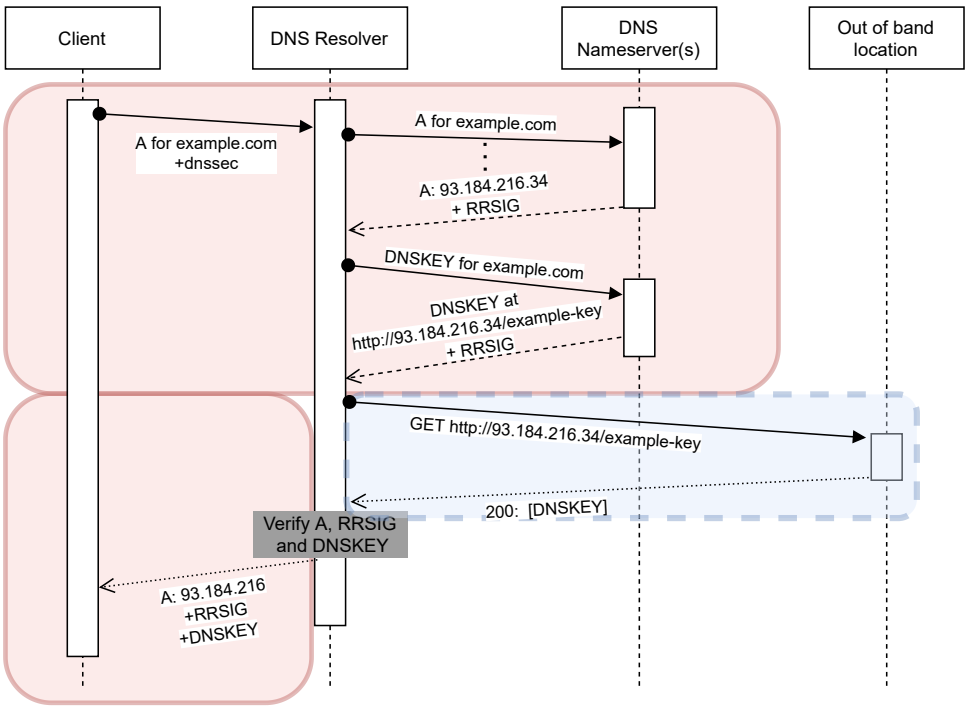


Figure 22: Outline of an out-of-band key exchange using a combination of DNS (red, solid line) and HTTP (blue, dashed line) protocols.

load within 2 or 3 seconds, and the goal should be that even with an out-of-band key exchange this threshold is not exceeded. Additionally, [61] states that Google includes the page load time in determining the relevancy in terms of position in the search results, i.e. extremely long loading times will result in a lower place in the Google search results. A study by Deloitte [62] shows that a delay of 1 second already causes users to lose focus in the task they were performing and beyond 10 seconds users get frustrated and are likely to abandon the site. Additionally, Stadnik and Nowak show in [63] that e-commerce websites lose turnover if the page load time increases. DNS resolving has a direct consequence on the page load time and an increased DNS resolving time will lead to a higher general latency since content cannot be fetched from a server only after the DNS resolving has been completed.

[61] states that the average page load time for results on the first page of Google is 1.65 seconds. For a first-page site to be able to deliver content within 2 seconds, the additional delay that could be added is only 0.35 seconds to still be able to have a page load time of at most 2 seconds. An increase in page load time more than 2 seconds would likely causes large webshops or other popular sites to not use an out-of-band solution since this would risk a decrease in turnover. This would in turn limit the adoption of proper security measures on the Internet if an out-of-band key exchange is deemed necessary. Consequently, an out-of-band key exchange that does not increase the total DNS resolving time with more than 350 ms compared to the current time is deemed acceptable.

8.2 Experiment setup

In this Subsection the DNS resolver Unbound is further described and the experiments and the setup will be discussed. First, the outline of an out-of-band key exchange is further elaborated on, afterwards the inner workings of the Unbound software are outlined together with the changes in the software that are necessary for this experiment. Then, different transport protocols that are used in this experiment are described.

8.2.1 Out-of-band key exchange

As mentioned before, an out-of-band key exchange is an exchange of key material outside of the current communication channel. In this case, it can be defined as an exchange of a DNSKEY that is not using DNS

packets to transfer the public key. In Figure 22 a diagram illustrating the requests that are made over the Internet is shown in the case an out-of-band key exchange is used over HTTP. For clarity, the protocols that are used are indicated with different colors. In the red solid line squares the current DNS protocol is used, in the blue dashed square the HTTP protocol is used. In this figure, an `A` record is requested for the domain `example.com` by the client. The resolver performs all necessary requests to get this information from different DNS name servers and will also request a `DNSKEY` record to verify the responses. It is in this DNSKEY response from the name server that a deviation from the normal domain resolving procedure is seen first. The DNS response does not contain the actual public key but a location where the key can be fetched from. In this case it can be fetched with a simple HTTP GET request from the HTTP server that is provided. When the DNSKEY has been successfully fetched from the server, the normal DNS resolving procedure can be continued.

If a DNSKEY would fit, this could of course be included in the DNS response. This way, a DNSKEY response can contain either the DNSKEY, or a location where to retrieve the key in case the key is too large for a DNS response. This can be indicated with a different algorithm number in the response header[64]. This algorithm number can indicate whether or not an additional request, and over what transport protocol is necessary to fetch the actual DNS public key. A response could thus contain a HTTP location, to which a simple GET request would fetch the key, as is depicted in Figure 22. These DNSKEY referrals to other locations would also need to be signed, just as other DNSKEY RR's, in order to create a signature over the complete DNSKEY RRSet. Additionally, signing these DNSKEY referrals also prevents attacks on this new structure such as DOS attacks based on sending broken links or denying public keys exist for a domain. For example, if a DNSKEY referral to a HTTP server is not signed, it could be altered by a threat actor to change or delete the URL in the response possibly leading to a DOS attack on the altered URL.

Different methods can be used to communicate a key over the Internet, in the following subsections the different transport protocols used in this experiment are described. First, the design and implementation of out-of-band key exchange in Unbound is elaborated on.

8.2.2 Designing and implementing out-of-band key exchange in Unbound

Unbound is an open-source, recursive DNS resolver that supports DNSSEC, it is developed by NLNet Labs [57]. Unbound keeps track of the progress of resolving a domain name in a state machine like construction in memory. For every request from a client, a structure is made that contains the progress of resolving the requested domain. Each response from a DNS name server can then advance a query state to the next, until the domain has been resolved to its IP address or a failure occurs. By using the state machine that Unbound creates, it is possible to detect whenever a DNSKEY state is reached and a DNSKEY has to be requested from a name server. This mechanism is then used to initiate an out-of-band key exchange to a preconfigured server using an already set transport protocol.

To test several transport protocols, ease the implementation of these and be able to quickly switch between them, a single adapter function is made in the Unbound source code that will run a given function from a Python file. This construction allows for a quick implementation in a scripting language that is easier to code than the C language that Unbound is written in. Different transport protocols can then more easily be implemented in Python and only the function call has to be adapted in the Unbound resolver to switch between different transport protocols.

Since there will be no name server on the Internet responding with a DNSKEY response that contains a location where to fetch it, this is simulated inside the Unbound resolver by triggering an out-of-band key exchange whenever a DNSKEY query is made. To benchmark different transport protocols, a worst-case scenario in which every DNS public key has to be fetched using an out-of-band method is simulated, corresponding to a situation in which every DNSKEY response would be too large.

Designing the experiment in this way will likely result in a bit of additional overhead since a python environment has to be build for every query. However, for developing and experimenting, this proved to be faster to develop and easier to experiment with different methods of out-of-band key exchange compared to developing all methods in the C language. Since all out-of-band key exchange methods implemented in Python are having the same additional overhead, this would have no effect on the comparison between the different methods. The modified Unbound resolver software has been published at Github [7].

[7] See `https://github.com/GijsBeernink/unbound`

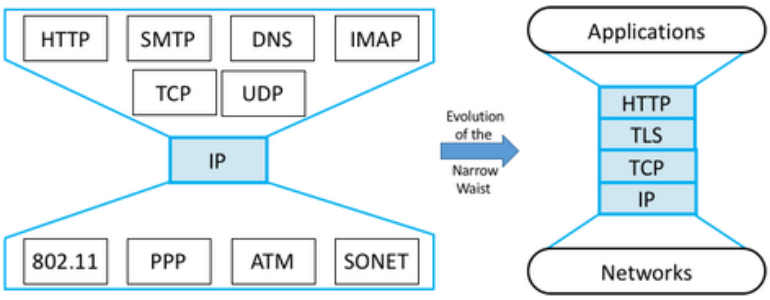


Figure 23: Evolution of the narrow waist, from [65]

8.2.3 HTTP out-of-band

HTTP has positioned itself in the past decade as the new narrow waist of the Internet hourglass model, as depicted in Figure 23. It is a well-known protocol and a large part of Internet traffic uses HTTP for communication. An HTTP out-of-band key exchange, as seen in the example in Figure 22, fetches the public key from a given HTTP server. This HTTP server could be located on premise, along the name server or be hosted somewhere else.

In this experiment, the most used HTTP version, namely HTTP/1.1 and the most recent version, HTTP/3, are tested for performance. HTTP/1.1 is the most used version on the Internet and uses the TCP protocol on the network layer. HTTP/3 uses a newer transport protocol, namely Quick UDP Internet Connections (QUIC)[66]. QUIC is a recently standardized transport protocol which utilizes the UDP protocol and tries to reduce connection latency and quicker transport of data. Since large resolvers might send a lot of HTTP traffic towards the HTTP server of a large name server, using a protocol that handles these requests fast is preferred. The difference in HTTP/1.1 and HTTP/3 is interesting because of these different transport protocols. Recent research already shows that HTTP/3 performance is especially significant in situations with high latency or poor bandwidth [67]. Since DNS public keys could be located in geographical locations far away, high latency could be countered with a protocol such as HTTP/3 over QUIC.

By using QUIC for HTTP/3 connections, all traffic over these connections are secured since TLS is adopted into the QUIC protocol by default. Since this is not the case for a plain HTTP/1.1 connection, also HTTP/1.1 over TLS (HTTPS) is experimented with in this research. Section 8.5 will discuss the security of these connections and the necessity of a TLS secured connection for a public key retrieval further. An observant reader will notice that HTTP/2 is skipped in this experiment. Differences between HTTP/1.1 and HTTP/2 performance are less significant for this experiment since HTTP/2 mostly improves speed of sending multiple resources using connection multiplexing [68], while in this study the data will solely consist of a single file. Therefore, it is not included in this research.

8.2.4 FTP out-of-band

Although HTTP is a very versatile protocol that can be used for file transfers and is positioning itself as the new waist of the Internet [65], another well-known protocol is studied for its performance as out-of-band key exchange protocol as well. FTP is an old and standardized file transfer protocol designed to transfer files over a network. Including this outdated protocol in this study is done to not enforce the already growing trend on the Internet to rely on HTTP for an increasing number of applications. Additionally, FTP is used as a means to benchmark the performance of HTTP against. In general, it is not advised to still use FTP for the transfer of files since it is not encrypted and more secure alternatives exist. These secure alternatives, such as SCP, SFTP and FTPS, all have the disadvantage of using credentials, which in this context can be considered as unwanted. Exchanging credentials to all resolvers to fetch public keys would cause additional overhead on the performance of the benchmarked solutions. In this study, a comparison to the not-secured FTP protocol can be interesting since FTP does not add headers to responses, as HTTP does, reducing the actual data that is transported. Additionally, in the context of this study, a secure connection is not necessary as the transported public key is signed. Any integrity issues regarding this connection can be detected when the signature over the public key is verified. While it is not advised in this research to use a not secure FTP connection, in the

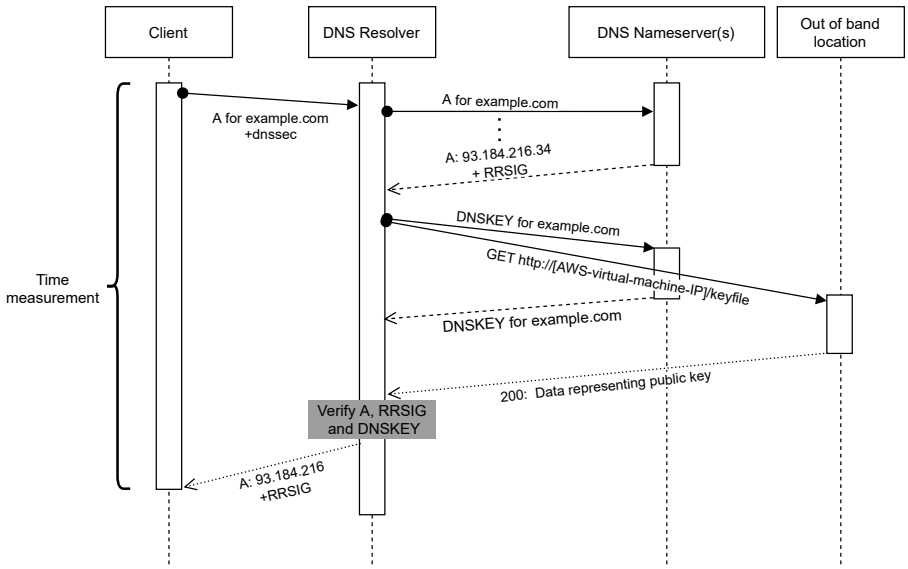


Figure 24: Implementation of out-of-band key exchange in Unbound.

context of this study it could be argued that a plain FTP connection can be used. Further security issues are discussed in Section 8.5.

8.2.5 Fetch key in chunks

Another option to transport large public keys could be to fetch a large public key in multiple smaller chunks which do fit in DNS packets. An implementation of this could either issue multiple requests to fetch all chunks of a public key or send one request and receive a stream of all chunks in smaller DNS packets. Each of these implementation has its own drawbacks, where the first would result in large amounts of Internet usage and load on a resolver as well as a name server, including additional delay for the many Round Trip Times for all requests that are created. The second implementation would require the servers to keep track of received packets and communicate for re-transmission of lost packets, besides using some sort of sequence numbering for reconstructing the public key from the received chunks. This already tends towards using a TCP connection and there already exist multiple transport protocols, such as the ones experimented in this research, that solve these issues. These disadvantages are the reason this study focuses on out-of-band key exchange methods. Analysing this approach to fetch large public keys in smaller chunks remains future work.

8.3 Experiment

Different transport protocols are used to fetch a large file of 150,000 bytes of random data, representing a large public key. Every out-of-band option discussed above is used in a run of 1000 dig commands to fetch the IP address of `utwente.nl` with the DNSSEC option enabled, requiring the Unbound resolver to fetch all keys and check all signatures leading up to this domain. The time it takes for a dig command to return an IP address is measured. Given that the cache on the Unbound resolver is disabled, the whole path from root name server to the `utwente.nl` name server is queried and verified for every dig command. For the used query, namely `utwente.nl`, first the root name server is contacted and a key is fetched, then the `.nl` name server is contacted, and at last the `utwente.nl` name server is queried. This results in 3 DNSKEY requests on every dig request. The consequence is that there will be 3 out-of-band key exchange requests as well.

A worst case scenario is created by disabling the cache and constantly querying and verifying every name server from root to the queried domain name. This process of 1000 dig commands is repeated 5 times to average out peak performances and other irrelevant artifacts. To keep the impact on the network and DNS resolvers low, only one request per second is made and after every run a delay of several minutes is started before the next run of 1000 dig commands is started.

Measurements\Runs	Run 1	Run 2	Run 3	Run 4	Run 5	Average
Measurement 1	0.927	1.385	1.184	1.199	0.773	1.0936
Measurement 2	0.701	0.797	1.186	1.279	1.086	1.0098
Measurement 3	0.718	1.052	1.021	0.778	1.216	0.9570

Table 13: Calculating the average of every n^{th} measurement from every run.

All experiments in this research use a PC located at the University of Twente dorms using Unbound as resolver and AWS virtual machines [69] located in two different geographical locations as either HTTP or FTP servers. All four protocols are used to fetch keys from the AWS locations Ohio and Frankfurt, measuring performance for servers geographically far away and close by respectively. Additionally, an experiment without any out-of-band key transportation protocols enabled is performed as reference performance. This reference experiment will deploy a Python environment but will immediately return a stub value without performing any other actions, resulting in a fairer comparison since the differences that are seen in the next section are a result of the used transport protocols. To find an estimate of the additional delay an out-of-band key exchange generates to resolving a domain name, this reference experiment can be used.

Experimental setup
The procedure depicted in Figure 22 for an out-of-band key exchange assumes that a DNS name server does respond with a URI on where to find the actual public key. Since DNS name servers at the moment do not behave in this manner and modifying and deploying an authoritative name server does not fit in the time constraints of this research, another solution to study this is implemented. As discussed in earlier sections, the Unbound software keeps track of a state machine construction for resolving a DNS query and an out-of-band request is started whenever a DNSKEY query is made. The consequence of this is that this experiment does not entirely model and measure the real-world impact that an out-of-band key exchange would make accurately. Figure 24 shows the order in which actions are executed by the modified resolver software. In this figure it can be seen that an out-of-band key exchange is started at the moment a DNSKEY query is made, as discussed before. The consequence is that all measurements do not include the round trip time of the DNSKEY query that is resolved. However, since all studied transport protocols have the same advantage, the comparison between these does not differ.

8.4 Results

Time measurements for different key transport protocols are plotted in Figure 25. Every data point in these graphs is the average of every n^{th} measurement in 5 individual runs of the same experiment. This is done to average out positive and negative peaks in performance on a single DNS query. To make this more clear, the first measurements of these 5 runs and their average are listed in Table 13. In the last column the average is calculated over the 5 runs, this Average column is plotted in Figure 25. To improve readability and smooth out the lines a moving average with a window of 20 is used.

From these figures, it is visible that for all researched out-of-band key exchange methods the location of the server gives a significant overhead. Servers located further away can increase the time to fully resolve a domain name. Fetching a large key from geographically far away locations increases the resolving time for these methods of transportation. To express the results in the increase that can be expected compared to the reference experiment that is discussed earlier in this section, Table 14 lists the median and change with respect to the reference implementation over the median for all locations and protocols used. Especially for the FTP protocol, where the difference of fetching a key from a close location like Frankfurt to a far away location like Ohio is seen very well. Using FTP to fetch a public key from Frankfurt would increase the DNS resolving time with 52% whereas fetching a key from a more remote location like Ohio almost reaches a 700% increase, as can be seen from Table 14. It is no surprise that every method that is tested does increase the resolving time, since an additional request is made besides the current normal DNS resolving procedure. The results that are seen for the tested transport protocols are further described in the following subsections.

Location	Protocol	Median (seconds)	Change w.r.t. reference	
Frankfurt	HTTP/1.1	0.777	177 ms	30.4 %
	HTTPS	0.777	177 ms	30.4 %
	HTTP/3	1.324	724 ms	122.2 %
	FTP	0.867	267 ms	45.6 %
Ohio	HTTP/1.1	2.019	1419 ms	238.9 %
	HTTPS	2.286	1686 ms	283.7 %
	HTTP/3	4.980	4380 ms	735.9 %
	FTP	4.651	4051 ms	680.6 %
Reference measurement		0.600	-	-

Table 14: Median resolving time using different out-of-band key exchange protocols.

Protocol	Increase Frankfurt-Ohio
HTTP/1.1	160 %
HTTPS	194 %
HTTP/3	276 %
FTP	436 %

Table 15: Increase for the same protocol between Frankfurt and Ohio locations.

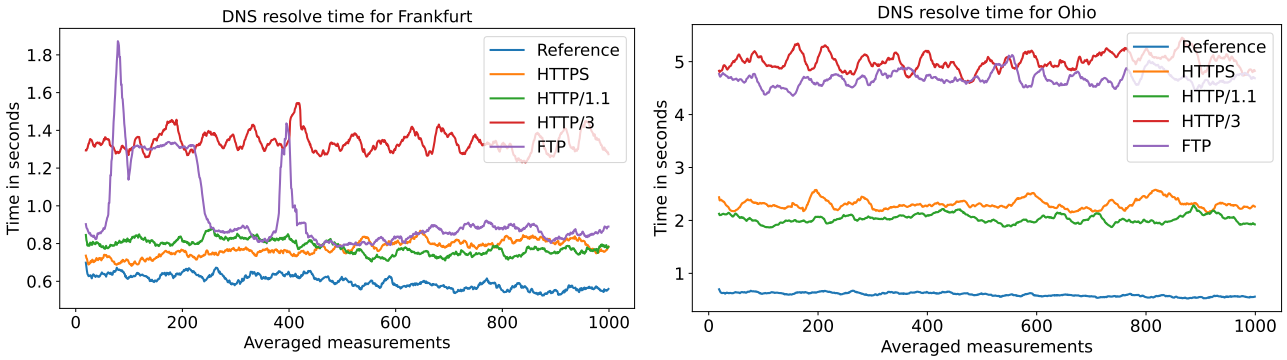
8.4.1 HTTP performance

Besides the location of a HTTP server that has a large impact on the resolving time, also the difference between HTTPS and HTTP/3 is remarkable. In this subsection an explanation for this behaviour is formulated. From Figures 25a and 25b, together with the data from Table 14 it is clear that HTTP/3 performs worse than HTTPS in this experiment based on the time it takes to resolve a domain name. Where HTTPS creates a delay of 2 seconds for far away locations, using HTTP/3 this can grow towards 5 seconds to complete. This is a contradictory finding compared to research from Trevisan et al. in [67] who concluded HTTP/3 to be faster than its predecessors, especially where a high latency is involved. The cause for these contradictory results in this study is most likely linked to the HTTP/3 implementation in Python and the libraries that are used for this. HTTPS and FTP connections can easily be created using standard libraries available in the Python environment and have been optimized over the years, however, this is not yet the case with the more recent HTTP/3 stack. The Python environment does not yet support the new HTTP/3 and QUIC protocol in its standard libraries. Since there are no standard libraries available to create a HTTP/3 server and client, in this experiment the `aioquic` [70] Python library is used. This library is used in other research as well, namely in [71] and [72]. `Aioquic` provides a minimal implementation of TLS 1.3, QUIC and an HTTP/3 stack. An additional benefit is that this library already provides an example server and client, which only needed small modifications to make it suitable for this experiment. Differences that are measured between HTTPS and the HTTP/3 implementation are most likely an artifact of the differences in the optimization of the libraries that highly impacts the performance. Assuming the HTTP/3 implementation can be optimized and perhaps written in a less abstract and more low-level language such as C, it might perform according to measurements of other research and outperform or at least near the performance of HTTPS.

From Table 14 it is also visible that the additional TLS layer for HTTPS does not increase the resolving time for locations that are close by. For locations further away, such as Ohio, the distance to communicate over creates a small overhead compared to a plain HTTP/1.1 connection.

8.4.2 FTP performance

Connections to FTP servers located close by are significantly faster than connections to FTP server geographically far away. In Figure 25a using FTP connections to fetch a large public key from a close location like Frankfurt slightly increases the resolving time of this query. For connections to servers that are far away, the time it takes drastically increases to almost 5 times as much, as can be seen in Figure 25b. Compared to the HTTP implementations, the overhead grows faster for remote locations, to over 436% compared to locations geographically close by, as is listed in Table 15. This makes FTP the least suitable to transfer files over the Internet over large distances. In this experiment, it outperforms the HTTP/3 protocol, however this is most



(a) Out-of-band key exchange for Frankfurt server.

(b) Out-of-band key exchange for Ohio server.

Figure 25: Domain resolving using different out-of-band key exchange protocols, grouped by location.

likely due to the inefficient implementation of the HTTP/3 protocol.

8.5 Security, Scalability and Reliability

Now that results have been described, the consequences for several aspects of the DNS infrastructure are analysed. In this section, the security impact, scalability issues and the reliability of DNS operators to switch to use out-of-band key exchange is described. First, security notions that might be adopted into such a system are discussed, afterwards the scalability of an out-of-band key exchange and the reliability on the proposed measures are discussed.

8.5.1 Security

Security considerations of the implemented solution are discussed using the CIA triad (Confidentiality, Integrity, Availability) with respect to the DNS environment. In this context, *confidentiality* focuses on the (lack of) privacy measures that are taken when resolving a domain name and *integrity* in the DNS environment addresses the verify-ability of the name server and ensuring that a DNS response has not been altered in transit. With the term *availability* the reachability of DNS services such as a resolver and name server and the valid DNS response from these is addressed.

Confidentiality
From DNS perspective, confidentiality is currently not available in the DNS infrastructure. Queries can be intercepted and logged by DNS operators or other parties that are able to monitor this traffic. There are several standards being developed to address privacy issues that currently exist in the DNS ecosystem, such as DNS over TLS (DoT) or DNS over HTTPS (DoH). These are not further discussed in this research, more information on these standards can be found in [73] on DoT and [74] on DoH.

Since the current infrastructure does not offer confidentiality, also the implemented out-of-band key exchange is leaking information. By contacting a HTTP server to fetch a public key, a party that is monitoring the network can infer to what name server this public key belongs to. Hence, the implemented solutions do not offer total confidentiality of the exchanged context. When privacy is addressed in the DNS ecosystem, either by DoH, DoT or another solution, then the leaking of public keys by an out-of-band request can also cause privacy issues. Depending on the design of the solutions for confidentiality in DNS, an out-of-band exchange could be adapted to allow for confidentiality also at this part of the process. For example, multiple public keys of different name servers could be hosted at a single server, fetching these over an encrypted connection would disable the ability of a party monitoring the network to infer the key that is requested and for what service this public key is.

Integrity
Integrity in the DNS ecosystem is guaranteed by DNSSEC, this makes it possible to verify that a response to a query came from a legitimate source and that the response has not been modified in transit. To enable an

out-of-band key exchange, it is necessary to extend the integrity that DNSSEC offers to the out-of-band key exchange protocol as well, to ensure that the chain of trust is not broken. It is therefore important that the researched out-of-band protocol also allows for the verification of the server as well as guaranteeing that the contents of a response cannot change in transit. Using an HTTPS or HTTP/3 connection to fetch the contents, this is enabled due to the use of TLS in these protocols. TLS allows for authentication of servers to each other by exchanging digital certificates and encrypts the connection such that contents cannot be altered unknowingly. Ït would then be necessary for the resolvers and name servers to know or trust each others certificate, or to have a trust anchor such as Certificate Authorities that are used in browsers.

However, since the response of a DNS name server containing the referral to the out-of-band public key location is signed, it can be verified once the key has been communicated. A decision to trust this response can then be made. Additionally, if a public key is changed during transit, the signatures from the DNS responses do not match and a valid DS record cannot be found. This already allows for the detection of a change in contents while fetching a public key. Combining this, also with an unsecured connection it is possible to guarantee integrity of a message. A big disadvantage of using an unsecured connection, besides privacy issues, is that it opens up a new attack possibility for a malicious entity. By altering the public key in transit, the signatures cannot be verified and thus the domain name resolving cannot complete. This could be used in a Denial of Service attack on a resolver, which would not be able to perform DNSSEC validation anymore. However, this could also be reached by simply dropping these public keys from the traffic, without needing to alter the contents of it. This, in turn, is already an existing problem in the current DNS protocol, where a dropped DNSKEY response would lead to a failure. An additional scenario that elaborates on this idea is discussed in the next paragraph on Availability.

In this study, an AWS virtual machine is used for which the IP address of this machine is directly used in the out-of-band key exchange protocol. In a real world implementation, it can be expected that a DNS response does not contain an IP address to directly contact, but yet another domain name where the public key can be retrieved from. This domain name must in turn be resolved to get the DNS public key. To prevent creating circular references when trying to get a public key, a similar solution to currently used DNS glue records could be added to a DNSKEY response. Currently, DNS glue records make sure that querying a domain does not return a name server at the same domain for which the same queries can be created to resolve this domain. Glue records already indicate at which IP address the name servers can be found.

Implementing such a solution in an out-of-band key exchange protocol implies that the resolver does not validate the NS resource record of the name server. To investigate whether the chain of trust is not broken and how this is solved currently, the current implementation at the Unbound resolver is consulted at [75]. The above discussed issue is referred to by the Unbound documentation as `harden-referral-path`, which is by default disabled. This means that a DNS glue response is not validated for its signatures by requesting the NS record set for this name server together with its signatures. Considering the chain of trust discussed in previous sections, this is not broken since eventually the public key is retrieved and the signatures can be verified. It could well be the case that a response is altered by a malicious party, however, then the retrieved key would not be able to verify the already received signatures resulting in a resolving failure. Hence, the chain of trust is not broken and an additional attack vector is not created by not verifying the domain name of the HTTP or FTP server. Additionally, the Unbound documentation reasons that verifying a NS RR set is by default disabled *"because it burdens the authority servers, and it is not RFC standard, and could lead to performance problems because of the extra query load that is generated."* [75].

Availability
The DNS infrastructure has been build very robust, having multiple name servers available on different geographical locations to create a resilient infrastructure. By employing an out-of-band key exchange protocol, this infrastructure should not be weakened by a single point of failure if a DNSKEY cannot be fetched. So, similarly to multiple DNS name servers on different locations, the same should be done for out-of-band key servers. Using anycasting it is possible to create multiple key servers where out-of-band key can be fetched from. However, DNS operators are specialized in operating a DNS server and might not have the technical abilities to run and maintain several anycasted HTTP servers. This is where Content Delivery Networks (CDN) will most likely be a solution, DNS operators could then outsource their HTTP anycasted infrastructure to a CDN that hosts these servers. This enables fast configuration and updating the keys with easy in case a key rollover has been planned. Additionally, if outsourced correctly, possibly to multiple companies, a high up-time

can be guaranteed.

Another aspect of availability that has to be taken into account is the availability of the resolver. If a resolver has to fetch multiple keys of a key-providing server for many different queries, it is important that the resolver keeps functioning and can handle this. Keys for large zone files do not have to be queried often, since these can be cached at the resolver, just as is happening currently. However, an attack could be mounted to bring a resolver to a stop. Assuming a malicious party has an own authoritative name server, it could publish a (location of a) public key with an extremely low TTL. This would mean that a resolver cannot cache it efficiently. If this malicious party provides an out-of-band location that is not reachable or has very limited bandwidth, the resolver will try to fetch a key from this location for every DNS query this malicious party than could make. It is likely that a resolver can, at that point, no longer serve other queries since it is busy fetching a key over a limited connection.

In general, a volume-based DoS attack would be easier to perform since a resolver must perform more work for the same amount of queries, i.e. make an additional request to fetch a public key. Using efficient caching mechanisms this could be limited, but in an attack such as described above, the availability of a DNS resolver might reduce. This is a known problem in the DNS ecosystem, where mitigation for DNS attacks are still being researched [76]. The implementation of an out-of-band key exchange protocol into DNS would create an additional attack surface that has to be resolved.

8.5.2 Scalability

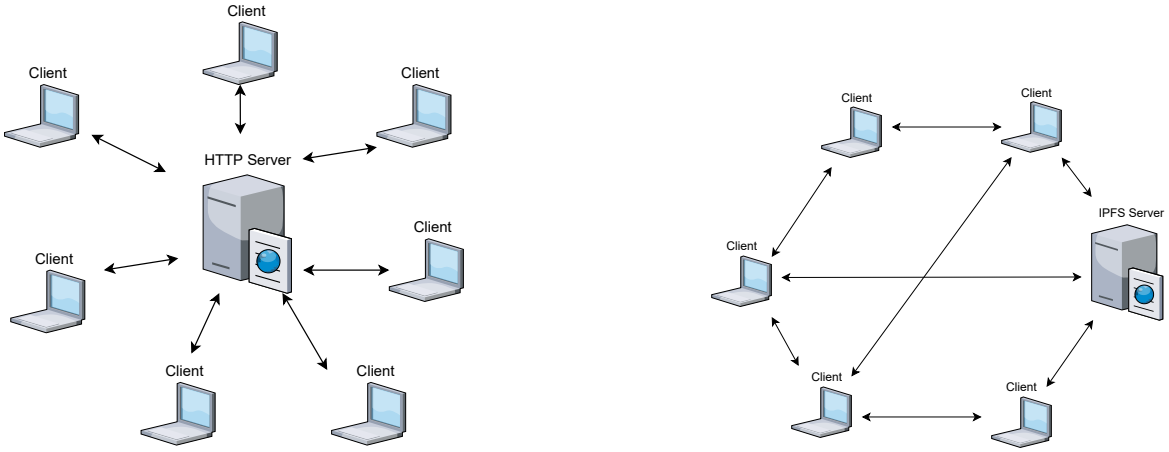
In a global network that the whole world uses, DNS has to be able to scale to allow for all its clients to receive proper domain resolving service. By adding an out-of-band key exchange protocol into DNS, it is also important that this part scales as well and does not become a bottleneck in the already complex DNS ecosystem. By its design, DNS is already highly scalable, where each organization can be responsible for its own content. An out-of-band solution must adhere to this concept as well. To ensure this, in the previous section, several remarks have been made already to provide availability and protect against an additional attack surface. CDNs would most likely be used to make all necessary key material available to a DNS resolver and its clients. The costs of using a CDN for hosting a few kilobytes of public key material is not extremely high [77]. This would most likely outweigh the costs and time to setup an environment for DNS operators themselves at the name server location and preferably at other locations for redundancy.

8.5.3 Reliability

In the previous section on scalability, an already emerging problem on the Internet is further enforced. The Internet and most of its services are already heavily centralised, Wang et al. describe in [78] that DNS services already rely heavily on CDN's. Web services use these parties as well to host its content. Given the recent crash at CDN provider Fastly, described in [79], which caused a large part of the world to not be able to browse many websites anymore, it is clear that the total reliance on several central parties is a risk. On the one end it is necessary to cut costs and make it easily available and configurable to all clients, on the other end, it is centralizing the Internet even more.

A more distributed protocol for hosting and exchanging files already exist, in the form of a peer-to-peer network. A solution to the centralisation problem that would theoretically be possible to scale up easily is based on the InterPlanetary File System (IPFS). IPFS is a protocol for storing data in a distributed network [80]. It uses peer-to-peer connections to share information based on the hash value of the content. In Figure 26 the difference between a HTTP and IPFS network infrastructure is depicted.

Since content is distributed in a peer-to-peer fashion and peers cache content for a while before it is garbage collected, the most popular and widely requested content will remain in the cache and will become more distributed over many peers in the network. In the case of this research, a public key that is requested very often by different resolvers, e.g. a DNS public key for the root name server or other large domains, will get more distributed over the network. Since many DNS resolvers will request this key and cache it for some time, this will likely locate the public key physically closer to other resolvers that yet have to fetch this key for the first time, or if the content is garbage collected. This will likely result in faster responses, since a public key will get geographically closer to the requesting resolver.



(a) HTTP server setting

(b) IPFS server setting

Figure 26: Comparison between HTTP and IPFS connections.

Using such a protocol would contribute to a less centralized Internet and could make DNS less reliable on other parties to host the necessary servers.

8.6 Concluding

In this section, different out-of-band protocols have been described and implemented in a DNS resolver. Although there does not yet exist a DNS name server on the Internet that responds with applicable responses, through the conducted experiments an estimation of the added resolving time has been found if this is implemented. Based on the results that are found, an HTTP out-of-band protocol would likely increase the DNS resolving time with 30% for a connection using HTTP/1.1 secured with TLS. The overhead can be decreased if solutions are directly implemented into a resolver in a more low level language such as C, instead of embedding Python programs in the resolver. The implemented protocols are further discussed on a security level and the effects on DNS operators and the (further) centralization of the Internet are discussed. Additionally, an alternative protocol is presented with a theoretical hypothesis on the performance of this protocol in the DNS infrastructure.

9 Discussion & Future Work

Results of all analyses are summarized in Table 16, together with the public key size and signature size of the PQC alternatives. The green coloured cells indicate properties performing similarly good or better compared to the currently used algorithm RSA/SHA-256. The performed analyses and experiments give a broad view of the performance of different PQC algorithms if these are to be adopted by DNSSEC. What is important, is that the analyses assume an adoption into DNSSEC with minimal changes to the DNSSEC protocol, i.e. adopting PQC alternatives just as a regular algorithm rollover. If public keys or signatures of these PQC alternatives create DNS packets larger than 64kB over TCP, a response cannot be send. This implies that using the Picnic algorithm, more than 96% of all DNS responses do not arrive at the resolver. Based on these results Falcon-512 is preferred here since none of the responses grow to over this maximum EDNS size.

The Falcon-512 algorithm is the only algorithm of all alternatives that can be adopted into DNSSEC without issues regarding communicational overhead or computational overhead. Falcon is the only algorithm that does not cause DNS responses to grow over the EDNS maximum size, making it the only protocol that could theoretically be deployed overnight without DNS losing functionality. Regarding the signing and verifying performance, Falcon performs really well as it can sign the whole `.nl` zone file in less time than is currently done using RSA/SHA-256. The increase in resolver load for verifying signatures does increase, however, since this is limited to less than 2%, it could be said that this is an acceptable increase and a very minimal upgrade to resolver machines might be necessary in some situations where a resolver is already performing at near 100% CPU load.

A disadvantage of Falcon-512 is the expected increase in TCP traffic. Since around 16.2% of all UDP packets grow over 1500 bytes in size, a TCP retry will occur for these packets. The consequence is that there will likely be an increase of almost 25% in TCP traffic. Comparing this to the Rainbow PQC alternative this will only cause a 0.2% increase in TCP traffic, however, an out-of-band key exchange method must be used since almost 25% of TCP packets grow over the EDNS maximum size. Whether an out-of-band key exchange method outweighs an increase of 25% in TCP traffic for Falcon-512, is left as future work.

However, research into PQC algorithms and the competition started by NIST are fairly new and no algorithm has even been standardized yet. It is likely we will see an improvement on currently researched algorithms or other PQC algorithms being developed in the future, also after the NIST competition has been finished. Chen et al. mention in [81] that there is no proof for the security of lattice based schemes, which Falcon is. So it is possible that developments in cryptanalysis might be able to break Falcon and other lattice based cryptography in the future. While for the moment lattice based cryptography can be considered secure, Chen et al. mention that multivariate cryptography does have a more proven track record given historical performance. Algorithms based on multivariate cryptography do require larger public keys. Considering the fact that PQC algorithms are relatively new and have not yet been standardized, it is good to already have an alternative process to exchange large public keys outside of DNS responses, should security issues regarding Falcon or other lattice based algorithms do arise in the future. This study proves that i) it is possible to exchange large public keys outside of DNS responses and ii) the provided solution does not increase the time to resolve a domain over an established threshold value of 350ms. An out-of-band key exchange is not necessary to be able to adopt the Falcon-512 PQC algorithm in DNSSEC. However, future work to measure performance with more real-world experiments or larger data sets to create an efficient out-of-band protocol is necessary to adopt other well performing algorithms such as Rainbow into DNSSEC.

Measurements on the out-of-band key exchange methods described in this paper have been performed assuming a worst-case scenario in which every DNSKEY request triggers an out-of-band key exchange request. A production implementation of out-of-band key exchange would most likely cache public keys, such that out-of-band key exchanges do not happen as often as in this study. It can be expected that the number of out-of-band key exchanges that are necessary will then decrease, especially for popular domains that are often queried. Additionally, the implementation that is created is not efficient since it utilizes a Python program embedded in the resolver. Implementing a more efficient solution likely decreases the resolving time even more and the results in this paper can thus be interpreted as an upper-bound of the performance of out-of-band key exchange.

Analysed traffic in this study is limited to traces captured at a university campus DNS resolver. This resulted

Algorithm	Public key size	Signature size	Packets > 64kB	.nl zone sign time (seconds)	Max. resolver CPU load
CRYSTALS-DILITHIUM-II	1.2kB	2.0kB	0.02%	11.3	5.27%
FALCON-512	0.9kB	0.7kB	0.0%	21.7	4.98%
Rainbow-Ia	158kB	66B	25.3 %	3.0	3.02%
RedGeMSS 128	375kB	35B	25.3 %	244.4	14.70%
Picnic	32B	34kB	96.1 %	216.1	206.60%
SPHINCS	32B	8kB	7.5 %	1798.3	41.52%
RSA/SHA-256	256B	256B	-	206.0	3.57%

Table 16: Summary of previous sections.

in almost 12 million DNS response packets generated predominantly by staff and students at the university. The diversity of the population that generated the internet traffic is thus limited. The packets were captured in a period of 24 hours which captures a single business day, but can contain internet trends of the population on that specific day. A study using a larger dataset of, for example, an ISP resolver could find differences in communicational overhead. For example, depending on the population that uses a resolver and the sites they visit, a shift in UDP versus TCP traffic could be created or the amount of DNS public keys that are requested could be different. This could result in a slight deviation from the communicational overhead that is found in this study.

Additionally, a log file from the Dutch ccTLD is used to calculate the effects of signing a zone file with PQC algorithms. While the Dutch ccTLD contains a large number of entries that have to be signed, the log file is not created for the use of this study and this could limit the accuracy of the results. A study that has more zone server data available or can perform measurements using an actual zone file will be able to create a more accurate view of the effect of PQC algorithms on zone servers' performance. However, since there is a large difference in performance as is depicted in Figure 19, using the less-accurate log file available to this study already indicates which algorithms might not be suitable to use in DNSSEC.

As discussed above, the communicational or computational overhead that is found in this study can deviate from future studies that have access to a larger dataset. However, the out-of-band key exchange protocol that is implemented in this study is not related to the used datasets in this study, so a similar implementation will most likely lead to similar results.

In Section 8.1 an additional delay of at most 350 ms has been discussed to be acceptable for an out-of-band key exchange. Looking at the results, the additional delay of resolving a domain name in a worst-case scenario will for most protocols stay below 350ms if key servers are located close by the resolvers. For key server farther away the additional delay is too large to be acceptable, other considerations regarding far away servers have already been discussed in this section. Methods that use HTTP/1.1, HTTPS or FTP could be used if placed close by clients using the current implementation. Considering the optimizations that could still be researched and implemented, it is well possible to have an out-of-band key exchange method in DNSSEC.

10 Conclusion

In this study, several aspects of the adoption of Post Quantum Cryptography algorithms in DNSSEC are studied. Quantum-resistant public-key cryptographic algorithms submitted to the NIST PQC competition are researched for their suitability in DNSSEC. First, using a theoretical approach, consequences of communicating PQC signatures and public keys in DNS response packets are studied. By modifying real-world DNS response packets and replacing existing signatures and public keys with PQC alternatives, the effect of these on the communicational overhead is analysed. From this it is concluded that several PQC alternatives exceed the maximum packet size that DNS responses are allowed to have and only Falcon-512 can be used as an alternative without changing the DNSSEC protocol. Second, possible issues regarding the computational overhead on both zone servers and resolvers are researched. The studied PQC alternatives have different signature signing and verifying properties. These have a direct effect on the workload of zone servers for signing a zone file and on resolvers that need to verify signatures. Based on log files of the Dutch ccTLD zone server the effects of using PQC alternatives are analysed. Data from the resolver located at the university campus is used to predict the additional workload for a DNS resolver. Based on these analysis, Rainbow-Ia performs best of all researched PQC alternatives and has a better performance on both zone signing and computational workload on a resolver compared to the most used algorithm currently, RSA/SHA-256. At last, to address high communicational overhead, an out-of-band key exchange method is implemented in the Unbound DNS resolver to benchmark different methods of out-of-band key exchange. Additionally, security considerations regarding using such a solution in the DNS ecosystem are discussed. Falcon-512 and Crystals-Dilithium are the algorithms currently most suitable to be adopted into DNSSEC, however arguments are made to keep studying an alternative method to communicate cryptographic material out-of-band as lattice based cryptography might be proven unsuitable to be used. This study has demonstrated that while DNSSEC has options to adopt PQC algorithms and has the *potential* to stay useful in the near future, additional research must ensure that DNSSEC *actually* stays useful in a quantum world.

A Code

Relevant code that is used in this study and referred to from the text is gathered here. The modified Unbound resolver source code used in this study can be found at `https://github.com/GijsBeernink/unbound`.

Listing 1: TShark command for UDP CSV construction

```
tshark -r $file -T fields > csv/udp/$file.csv -2 -R udp.port==53 \
  -e dns \
  -e dns.qry.type  \
  -e dns.rr.udp_payload_size  \
  -e udp.payload  \
  -e dns.resp.z.do  \
  -e dns.resp.edns0_version  \
  -e dns.rrsig.algorithm \
  -e dns.rrsig.signature  \
  -e dns.rrsig.type_covered  \
  -e dns.dnskey.algorithm  \
  -e dns.dnskey.flags \
  -e dns.dnskey.public_key  \
  -E separator=';' -E header=y
```

Listing 2: TShark command for TCP CSV construction

```
tshark -r $file -T fields > csv/tcp/$file.csv -2 -R tcp.port==53 \
  -e dns \
  -e dns.qry.type \
  -e dns.rr.udp_payload_size \
  -e tcp.payload \
  -e tcp.reassembled.data \
  -e tcp.reassembled.length \
  -e dns.resp.z.do \
  -e dns.resp.edns0_version \
  -e dns.rrsig.algorithm \
  -e dns.rrsig.signature \
  -e dns.rrsig.type_covered  \
  -e dns.dnskey.algorithm \
  -e dns.dnskey.flags \
  -e dns.dnskey.public_key \
  -E separator=';' -E header=y
```

B Zone files

Zone files used in this researched are listed below.

Listing 3: Zone file of the root name server

```
$ORIGIN .
$TTL    604800

.       IN      SOA     ns1.root. hostmaster.root. (
                              3       ; Serial
                         604800       ; Refresh
                          86400       ; Retry
                        2419200       ; Expire
                         604800 )     ; Negative Cache TTL
```

```
; name servers - NS records
.               IN      NS      ns1.root.
tld.            IN      NS      ns1.tld.


; name servers - A records
ns1.root.       IN      A       172.20.0.3
ns1.tld.        IN      A       172.20.0.4
```

Listing 4: Zone file of the .tld name server

```
$TTL    604800
.       IN      SOA     ns1.tld. hostmaster.tld. (
                                6       ; Serial
                           604800       ; Refresh
                            86400       ; Retry
                          2419200       ; Expire
                           604800 )     ; Negative Cache TTL

; name servers - NS records
.       IN      NS      ns1.tld.
; name servers - A records
ns1     IN      A       172.20.0.4

test.tld.       IN      A       42.42.42.42
test.tld.       IN      AAAA    2001:db8:3333:4444:5555:6666:7777:8888
test.tld.       IN      MX      10 mail.tld.
mail.tld.       IN      A       42.43.44.45
test.tld.       IN      TXT     "This is a TXT record."
```

References

[1] R. Waters, "Quantum advantage is the next goal in the race for a new computer age," *Financial Times*, Jan. 2022. [Online]. Available: https://www.ft.com/content/e70fa0ce-d792-4bc2-b535-e29969098dc5

[2] M. Sparkes, "2022 preview: Quantum computers may finally become useful tools," Dec. 2021. [Online]. Available: https://www.newscientist.com/article/mg25233661-700-2022-preview-quantum-computers-may-finally-become-useful-tools/

[3] ScienceDaily, "Important milestone in the creation of a quantum computer," Jan. 2021. [Online]. Available: https://www.sciencedaily.com/releases/2020/12/201228101807.htm

[4] M. Mosca, "Cybersecurity in an era with quantum computers: will we be ready?" Tech. Rep. 1075, 2015. [Online]. Available: http://eprint.iacr.org/2015/1075

[5] P. Mockapetris, "RFC1035 - DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION." [Online]. Available: https://www.ietf.org/rfc/rfc1035.html

[6] P. V. Mockapetris, "Domain names - concepts and facilities." [Online]. Available: https://tools.ietf.org/html/rfc1034

[7] Security Associates Institute, "Attacking the DNS Protocol," *DNS Security*, p. 11, Oct. 2003.

[8] S. Ariyapperuma and C. J. Mitchell, "Security vulnerabilities in DNS and DNSSEC," in *The Second International Conference on Availability, Reliability and Security (ARES'07)*. Vienna, Austria: IEEE, 2007, pp. 335–342. [Online]. Available: http://ieeexplore.ieee.org/document/4159821/

[9] D. Atkins and R. Austein, "RFC3833 - Threat Analysis of the Domain Name System (DNS)," Aug. 2004. [Online]. Available: https://tools.ietf.org/html/rfc3833

[10] C. W. Kaufman, "Domain Name System Security Extensions," Jan. 1997. [Online]. Available: https://tools.ietf.org/html/rfc2065

[11] M. Larson, D. Massey, S. Rose, R. Arends, and R. Austein, "DNS Security Introduction and Requirements," Mar. 2005. [Online]. Available: https://tools.ietf.org/html/rfc4033

[12] ——, "Resource Records for the DNS Security Extensions," Mar. 2005. [Online]. Available: https://tools.ietf.org/html/rfc4034

[13] ——, "Protocol Modifications for the DNS Security Extensions," Mar. 2005. [Online]. Available: https://tools.ietf.org/html/rfc4035

[14] J. Ihren and S. Weiler, "Minimally Covering NSEC Records and DNSSEC On-line Signing," Apr. 2006. [Online]. Available: https://tools.ietf.org/html/rfc4470

[15] O. M. Kolkman, "DNSSEC Operational Practices," Sep. 2006. [Online]. Available: https://tools.ietf.org/html/rfc4641

[16] D. Blacka, B. Laurie, G. Sisson, and R. Arends, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence," Mar. 2008. [Online]. Available: https://tools.ietf.org/html/rfc5155

[17] P. Hoffman, "Cryptographic Algorithm Identifier Allocation for DNSSEC," Nov. 2010. [Online]. Available: https://tools.ietf.org/html/rfc6014

[18] R. van Rijswijk-Deij, A. Sperotto, and A. Pras, "Making the Case for Elliptic Curves in DNSSEC," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 5, pp. 13–19, Sep. 2015. [Online]. Available: https://dl.acm.org/doi/10.1145/2831347.2831350

[19] M. Müller, J. de Jong, M. van Heesch, B. Overeinder, and R. van Rijswijk-Deij, "Retrofitting post-quantum cryptography in internet protocols: a case study of DNSSEC," *ACM SIGCOMM Computer Communication Review*, vol. 50, no. 4, pp. 49–57, Oct. 2020. [Online]. Available: https://doi.org/10.1145/3431832.3431838

[20] J. Damas, M. Graff, and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))," Apr. 2013. [Online]. Available: https://tools.ietf.org/html/rfc6891

[21] C. Kaufman, R. Perlman, and B. Sommerfeld, "DoS protection for UDP-based protocols," in *Proceedings of the 10th ACM conference on Computer and communications security*, ser. CCS '03. New York, NY, USA: Association for Computing Machinery, Oct. 2003, pp. 2–7. [Online]. Available: https://doi.org/10.1145/948109.948113

[22] A. Herzberg and H. Shulman, "Fragmentation Considered Poisonous, or: One-domain-to-rule-them-all.org," in *2013 IEEE Conference on Communications and Network Security (CNS)*, Oct. 2013, pp. 224–232.

[23] N. Weaver, C. Kreibich, B. Nechaev, and V. Paxson, "Implications of Netalyzr's DNS Measurements," p. 8, 2011.

[24] G. V. D. Broek, R. V. Rijswijk-Deij, A. Sperotto, and A. Pras, "DNSSEC meets real world: dealing with unreachability caused by fragmentation," *IEEE Communications Magazine*, vol. 52, no. 4, pp. 154–160, Apr. 2014, conference Name: IEEE Communications Magazine.

[25] G. C. M. Moura, M. Muller, M. Davids, M. Wullink, and C. Hesselman, "Fragmentation, truncation, and timeouts: are large DNS messages falling to bits?" p. 18, Nov. 2020.

[26] D. Moody, G. Alagic, D. C. Apon, D. A. Cooper, Q. H. Dang, J. M. Kelsey, Y.-K. Liu, C. A. Miller, R. C. Peralta, R. A. Perlner, A. Y. Robinson, D. C. Smith-Tone, and J. Alperin-Sheriff, "Status report on the second round of the NIST post-quantum cryptography standardization process," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep. NIST IR 8309, Jul. 2020. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8309.pdf

[27] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, Nov. 1994, pp. 124–134.

[28] R. Overbeck and N. Sendrier, "Code-based cryptography," in *Post-Quantum Cryptography*, D. J. Bernstein, J. Buchmann, and E. Dahmen, Eds. Berlin, Heidelberg: Springer, 2009, pp. 95–145. [Online]. Available: https://doi.org/10.1007/978-3-540-88702-7_4

[29] L. D. Feo, D. Jao, and J. Plût, "Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies," 2011. [Online]. Available: http://eprint.iacr.org/2011/506

[30] D. Micciancio, "Shortest Vector Problem," in *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg, Ed. Boston, MA: Springer US, 2005, pp. 569–570. [Online]. Available: https://doi.org/10.1007/0-387-23483-7_392

[31] D. F. Moody, "Let's Get Ready to Rumble- The NIST PQC "Competition"," Apr. 2018. [Online]. Available: https://csrc.nist.gov/CSRC/media/Presentations/Let-s-Get-Ready-to-Rumble-The-NIST-PQC-Competiti/images-media/PQCrypto-April2018_Moody.pdf

[32] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Dilithium," p. 32, Mar. 2019.

[33] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, "Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU," p. 67, Jan. 2020.

[34] J. Ding, M.-S. Chen, A. Petzoldt, D. Schmidt, and B.-Y. Yang, "rainbow-pars.pdf," Sep. 2020. [Online]. Available: https://troll.iis.sinica.edu.tw/by-publ/recent/rainbow-pars.pdf

[35] A. Kipnis, J. Patarin, and L. Goubin, "Unbalanced Oil and Vinegar Signature Schemes," in *Advances in Cryptology — EUROCRYPT '99*, ser. Lecture Notes in Computer Science, J. Stern, Ed. Berlin, Heidelberg: Springer, 1999, pp. 206–222.

[36] W. Beullens, "Improved cryptanalysis of UOV and Rainbow," vol. 2020, p. 26, Oct. 2020. [Online]. Available: ia.cr/2020/1343

[37] A. Casanova, J.-C. Faugère, G. Macario-Rat, J. Patarin, L. Perret, and J. Ryckeghem, "GeMSS: A Great Multivariate Short Signature," Apr. 2020.

[38] M. Chase, D. Derler, S. Goldfeder, J. Katz, V. Kolesnikov, C. Orlandi, and S. Ramacher, "The Picnic Signature Scheme Design Document," Apr. 2020. [Online]. Available: https://github.com/microsoft/Picnic

[39] D. Bernstein, C. Dobraunig, M. Eichlseder, S. Fluhrer, S.-L. Gazdag, A. Hülsing, P. Kampanakis, S. Kölbl, T. Lange, M. M. Lauridsen, F. Mendel, R. Niederhagen, C. Rechberger, J. Rijneveld, and P. Schwabe, "sphincs+-specification.pdf," Nov. 2017. [Online]. Available: https://sphincs.org/data/sphincs+-specification.pdf

[40] R. van Rijswijk-Deij, K. Hageman, A. Sperotto, and A. Pras, "The Performance Impact of Elliptic Curve Cryptography on DNSSEC Validation," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 738–750, Apr. 2017, conference Name: IEEE/ACM Transactions on Networking.

[41] R. v. Rijswijk-Deij, M. Jonker, and A. Sperotto, "On the adoption of the elliptic curve digital signature algorithm (ECDSA) in DNSSEC," in *2016 12th International Conference on Network and Service Management (CNSM)*, Oct. 2016, pp. 258–262, iSSN: 2165-963X.

[42] M. Müller, W. Toorop, T. Chung, J. Jansen, and R. v. Rijswijk-Deij, "The Reality of Algorithm Agility: Studying the DNSSEC Algorithm Life-Cycle," in *IMC '20: Proceedings of the 20th ACM Internet Measurement Conference*. ACM Publishing, Oct. 2020, pp. 295–308. [Online]. Available: https://research.utwente.nl/en/publications/the-reality-of-algorithm-agility-studying-the-dnssec-algorithm-li

[43] D. Sikeridis, P. Kampanakis, and M. Devetsikiotis, "Post-Quantum Authentication in TLS 1.3: A Performance Study," in *Proceedings 2020 Network and Distributed System Security Symposium*. San Diego, CA: Internet Society, 2020. [Online]. Available: https://www.ndss-symposium.org/wp-content/uploads/2020/02/24203.pdf

[44] E. Crockett, C. Paquin, and D. Stebila, "Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH," *Cryptology ePrint Archive, Report 2019/858*, Jul. 2019. [Online]. Available: https://eprint.iacr.org/2019/858

[45] A. Herzberg, H. Shulman, and B. Crispo, "Less is more: cipher-suite negotiation for DNSSEC," Dec. 2014.

[46] C. Paquin, D. Stebila, and G. Tamvada, "Benchmarking Post-quantum Cryptography in TLS," in *Post-Quantum Cryptography*, ser. Lecture Notes in Computer Science, J. Ding and J.-P. Tillich, Eds. Cham: Springer International Publishing, 2020, pp. 72–91.

[47] M. Braithwaite, "Experimenting with Post-Quantum Cryptography," Jul. 2016. [Online]. Available: https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html

[48] Adam Langley, "ImperialViolet - CECPQ1 results," Nov. 2016. [Online]. Available: https://www.imperialviolet.org/2016/11/28/cecpq1.html

[49] Kris Kwiatkowski and Luke Valenta, "TLS Post-Quantum Experiment," Oct. 2019. [Online]. Available: https://blog.cloudflare.com/the-tls-post-quantum-experiment/

[50] Kris Kwiatkowski, "Towards Post-Quantum Cryptography in TLS," Jun. 2019. [Online]. Available: https://blog.cloudflare.com/towards-post-quantum-cryptography-in-tls/

[51] Adam Langley, "ImperialViolet - Real-world measurements of structured-lattices and supersingular isogenies in TLS," Oct. 2019. [Online]. Available: https://www.imperialviolet.org/2019/10/30/pqsivssl.html

[52] J. Goertzen, "BIND 9," Aug. 2021, original-date: 2021-08-08T19:13:53Z. [Online]. Available: https://github.com/Martyrshot/OQS-bind

[53] ——, "liboqs," Oct. 2021, original-date: 2021-06-29T21:42:26Z. [Online]. Available: https://github.com/Martyrshot/liboqs

[54] ——, "OQS-bind test network," Aug. 2021, original-date: 2021-08-09T04:57:55Z. [Online]. Available: https://github.com/Martyrshot/OQS-Bind-testing-env

[55] D. J. Bernstein and T. Lange, "eBACS: ECRYPT Benchmarking of Cryptographic Systems." [Online]. Available: https://bench.cr.yp.to/ebats.html

[56] R. Lamb, "DNSSEC Deployment http://rick.eng.br/dnssecstat/," Nov. 2021. [Online]. Available: https://rick.eng.br/dnssecstat/

[57] NLNet Labs, "Unbound." [Online]. Available: https://nlnetlabs.nl/projects/unbound/about/

[58] Internet Systems Consortium, "BIND 9." [Online]. Available: https://www.isc.org/bind/

[59] Petr Špaček, "OARC 33," Sep. 2020. [Online]. Available: https://indico.dns-oarc.net/event/34/contributions/782/

[60] Brian Dean, "We Analyzed 5.2 Million Webpages. Here's What We Learned About PageSpeed," Oct. 2019. [Online]. Available: https://backlinko.com/page-speed-stats

[61] N. Schäferhoff, "30+ Website Load Time Statistics," Mar. 2021. [Online]. Available: https://websitesetup.org/news/website-load-time-statistics/

[62] Deloitte Ireland, "Milliseconds Make Millions." [Online]. Available: https://www2.deloitte.com/ie/en/pages/consulting/articles/milliseconds-make-millions.html

[63] W. Stadnik and Z. Nowak, "The Impact of Web Pages' Load Time on the Conversion Rate of an E-Commerce Platform," in *Information Systems Architecture and Technology: Proceedings of 38th International Conference on Information Systems Architecture and Technology – ISAT 2017*, ser. Advances in Intelligent Systems and Computing, L. Borzemski, J. Świątek, and Z. Wilimowska, Eds. Cham: Springer International Publishing, 2018, pp. 336–345.

[64] IANA, "IANA — Technical requirements for authoritative name servers," Feb. 2020. [Online]. Available: https://www.iana.org/help/nameserver-requirements

[65] L. Peterson, "HTTP is the New Narrow Waist," Mar. 2019. [Online]. Available: http://www.systemsapproach.org/1/post/2019/03/http-is-the-new-narrow-waist.html

[66] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," Internet Engineering Task Force, Request for Comments RFC 9000, May 2021, num Pages: 151. [Online]. Available: https://datatracker.ietf.org/doc/rfc9000

[67] M. Trevisan, I. Drago, and A. S. Khatouni, "Measuring HTTP/3: Adoption and Performance," *2021 19th Mediterranean Communication and Computer Networking Conference (MedComNet)*, pp. 1–8, Jun. 2021, arXiv: 2102.12358. [Online]. Available: http://arxiv.org/abs/2102.12358

[68] Cloudflare, "HTTP/2 vs. HTTP/1.1: How do they affect web performance?" [Online]. Available: https://www.cloudflare.com/learning/performance/http2-vs-http1.1/

[69] I. Amazon Web Services, "Amazon EC2." [Online]. Available: https://aws.amazon.com/ec2/

[70] J. Lainé, "aioquic — aioquic documentation." [Online]. Available: https://aioquic.readthedocs.io/en/latest/#

[71] R. Marx, J. Herbots, W. Lamotte, and P. Quax, "Same Standards, Different Decisions: A Study of QUIC and HTTP/3 Implementation Diversity," in *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC*. Virtual Event USA: ACM, Aug. 2020, pp. 14–20. [Online]. Available: https://dl.acm.org/doi/10.1145/3405796.3405828

[72] J. Dizdarević and A. Jukan, "Experimental Benchmarking of HTTP/QUIC Protocol in IoT Cloud/Edge Continuum," in *ICC 2021 - IEEE International Conference on Communications*, Jun. 2021, pp. 1–6, iSSN: 1938-1883.

[73] R. Houser, Z. Li, C. Cotton, and H. Wang, "An investigation on information leakage of DNS over TLS," in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, ser. CoNEXT '19. New York, NY, USA: Association for Computing Machinery, Dec. 2019, pp. 123–137. [Online]. Available: https://doi.org/10.1145/3359989.3365429

[74] T. Böttger, F. Cuadrado, G. Antichi, E. L. Fernandes, G. Tyson, I. Castro, and S. Uhlig, "An Empirical Study of the Cost of DNS-over-HTTPS," in *Proceedings of the Internet Measurement Conference*, ser. IMC '19. New York, NY, USA: Association for Computing Machinery, Oct. 2019, pp. 15–21. [Online]. Available: https://doi.org/10.1145/3355369.3355575

[75] NLNet Labs, "unbound.conf(5) — Unbound documentation," Dec. 2021. [Online]. Available: https://unbound.docs.nlnetlabs.nl/en/latest/manpages/unbound.conf.html?highlight=harden-referral-path#unbound-conf-5

[76] L. Fang, H. Wu, K. Qian, W. Wang, and L. Han, "A Comprehensive Analysis of DDoS attacks based on DNS," *Journal of Physics: Conference Series*, vol. 2024, no. 1, p. 012027, Sep. 2021, publisher: IOP Publishing. [Online]. Available: https://doi.org/10.1088/1742-6596/2024/1/012027

[77] M. Williams and D. Athow, "Best CDN providers of 2021 to speed up any website," Dec. 2021. [Online]. Available: https://www.techradar.com/news/best-cdn-providers

[78] S. Wang, K. MacMillan, B. Schaffner, N. Feamster, and M. Chetty, "A First Look at the Consolidation of DNS and Web Hosting Providers," *arXiv:2110.15345 [cs]*, Oct. 2021, arXiv: 2110.15345. [Online]. Available: http://arxiv.org/abs/2110.15345

[79] P. Haskell-Dowland, "Fastly global internet outage: why did so many sites go down — and what is a CDN, anyway?" [Online]. Available: http://theconversation.com/fastly-global-internet-outage-why-did-so-many-sites-go-down-and-what-is-a-cdn-anyway-162371

[80] "IPFS Powers the Distributed Web." [Online]. Available: https://ipfs.io/

[81] L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone, "Report on Post-Quantum Cryptography," National Institute of Standards and Technology, Tech. Rep. NIST IR 8105, Apr. 2016. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8105.pdf