The Performance Impact of Elliptic Curve Cryptography on DNSSEC for Constrained Hardware Architectures

Breus Blaauwendraad
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
b.blaauwendraad@gmail.com

ABSTRACT

The Domain Name System (DNS) is an Internet protocol, with one of its tasks being the translation and mapping of human memorable domain names to computer usable IP addresses. Due to major security flaws in the DNS, security measures were needed. DNS Security Extensions (DNSSEC) are a suite of extensions that provide authenticity and integrity to DNS responses, using digital signatures. Unfortunately, DNSSEC is not without its flaws as well. Earlier research has shown that many of these issues are caused by the use of RSA as default cryptosystem for DNSSEC. Switching to an Elliptic Curve Cryptography (ECC-)based cryptosystem, will significantly mitigate these problems; however, verification of an ECC-based signature requires much more computing power. Earlier research has shown that for *server-class* architectures this is not an issue. However, local DNS resolving comes with privacy benefits for users. Therefore, we analyse whether the performance impact is problematic for home routers with a constrained hardware architecture. We examine the question "Will switching the cryptosystem of DNSSEC to an ECC-based one lead to performance issues for DNS resolvers on a constrained hardware architecture?" In this research, we construct a DNS scenario test and adopt a model to determine a worst-case value for the amount of signatures a constrained hardware resolver should be able to verify. Subsequently, we benchmark a variety of ECC verification algorithms and implementations to determine whether this amount is obtainable for mentioned architectures. Provided that certain implementation choices are made, we conclude that switching the cryptosystem of DNSSEC to an ECC-based one is feasible for home routers.

Keywords

DNS; DNSSEC; ARM; DNS resolvers; Elliptic Curve Cryptography; ECC Benchmark

1. INTRODUCTION

The Domain Name System (DNS) is one of the most important network protocols in the Internet protocol suite. One of its prime functions is the translation and mapping of human readable domain names (e.g. www.google.com) to computer comprehensible IP addresses (e.g. 216.58.204.4). Due to security flaws found in DNS[2], there was the need for security measures. DNS Security Extensions (DNSSEC) are a suite of extensions that provide authentication and integrity for DNS responses, using digital signatures. In July 2008, Dan Kaminsky uncovered a major security breach in DNS, called cache poisoning[7]. Abusing this flaw makes it possible for an attacker to falsify DNS information at will. The discovery of this flaw significantly accelerated the development and deployment of DNSSEC.

RSA is the default cryptosystem for DNSEC, used for the signing and verification of the digital signatures. It has the disadvantage of making the DNS responses very long, due to the size needed for an RSA digital signature to be secure. Over time, larger RSA keys were prescribed to guarantee security against brute force attacks. As computer power further increases, this issue becomes more severe for the future.

Earlier research has shown that the large signatures are the root of two problems[12]. First of all, due to the large digital signatures, the DNS responses are significantly larger than the initiated queries. This increase in message size can be misused for amplification in DDoS attacks[11].
Second, due to the size of a secure RSA digital signature, it can take more than one Internet packet to send the DNS response. Due to the firewall configurations, up to 10% of Internet hosts are unable to receive fragmented DNSSEC messages. [9, 10].
To achieve the same security in bits for Elliptic Curve Cryptography (ECC), much shorter digital signatures are required[6], as shown in subsection 2.3. The switch of the cryptosystem for the DNSSEC digital signatures from RSA to ECC mitigates both response size issues. In the future, the difference between RSA and ECC signature sizes is likely to increase vastly, as will be shown later in the paper.

However, ECC comes with its own problems. In commonly used scenarios, verification of an ECC-based digital signature requires up to an order of a magnitude more computing power than an RSA-based one.[4]. This could result in a slow Internet connection when the DNS resolver lacks the capacity to verify all the incoming signatures rapidly enough. In the paper which has led to this research[5], the authors ascertained that for resolvers on server-class machines, this will not be the case; however, that does not imply it will not cause performance issues for home routers with a constrained hardware architecture. Since running a DNS resolver on a home router comes with great privacy benefits, as we will explain in the next section, we analyse this topic. We examine it with the following defined question: "Will switching the cryptosystem of DNSSEC to an ECC-based one lead to performance issues for DNS

27^{th} *Twente Student Conference on IT* July 7^{th}, 2017, Enschede, The Netherlands.

resolvers on a constrained hardware architecture?"

We need a reliable estimate of how many signatures a constrained hardware architecture should be able to verify for it to function for DNSSEC. We execute a real life DNS scenario test for this prediction, and adopt a model based on another one presented in an earlier paper [5]. With the result from the scenario test as input for the model, we come to a worst-case scenario of how many signatures the resolver would presumably be able to verify. Subsequently, we benchmark ECC-based validation for home routers by benchmarking various algorithms and implementations, on a representative architecture.

The contribution of this work is the answer to the question whether an ECC-based cryptosystem is, performance wise, suitable for DNSSEC for constrained hardware architectures, such as home routers. Additionally, we contribute two sets of ECC signature verification benchmark results on two different hardware architectures.

1.1 Goals, research questions and approach

The purpose of this work is to determine if switching the cryptosystem used in DNSSEC to an ECC-based one will cause performance problems for constrained hardware resolvers such as home routers. The main question: "will switching the cryptosystem of DNSSEC to an ECC-based one lead to performance issues for DNS resolvers on a constrained hardware architecture?", is split up in two research questions:

RQ1: *How many signatures per second would a home router presumably have to verify in a worst-case scenario?*

Examining this question provides a worst-case scenario of how many signatures a home resolver should be able to verify, to function for DNSSEC. The methods we use to study this question are the adoption of a model based on earlier presented work[5], in combination with a real life DNS scenario test on a home resolver to obtain a representative input for the model.

RQ2: *How do ECC implementations perform on the CPU architectures that are common in home routers?*

After finding out how many ECC signatures a home resolver should be able to verify in a worst-case, we need to determine how many signatures such a resolver is capable to verify. Therefore, we benchmark a representative constrained hardware architecture, similar to the one used for home resolvers, to come to a reliable estimate.

To be future proof, the worst-case parameters are taken, which cannot be outgrown by growth of DNSSEC. Right now, only about 3% all domain names are employing DNSSEC [1][5]. In this model it is assumed to be 100%, including a higher amount of signatures per response than the current average, as discussed in subsection 4.1.

1.2 Paper structure

In section 2, background information on various important topics for this research, such as the DNS, DNSSEC and ECC are provided. Section 3 discusses related work to provide context for our research, and shows the fit with the existing knowledge in the field. In section 4, the research methods we adopt are explained in depth. Section 5 discusses the results obtained, with these research methods. In section 6, various issues are discussed concerning the results and potential weaknesses in the research methods. In section 7 the conclusion of the research is presented. At last, in section 8 future work is proposed.

[1]http://internetsociety.org/deploy360/dnssec/statistics/

2. BACKGROUND

2.1 Domain Name System (DNS)

The Domain Name System is a tree-structured hierarchical system and network protocol. An actual part of the current DNS tree is shown in figure 2.1, to give an illustration of how this works. The root of the DNS tree is managed by the Internet Corporation for Assigned Names and Numbers (ICANN). It authorizes domain name *registries*, who manage Top-level domains (TLD's) and *registrars*, through which domain names may be registered and reassigned. Sub domains can be created and modified by the *registrant*, which is a person or company who registers a domain name.

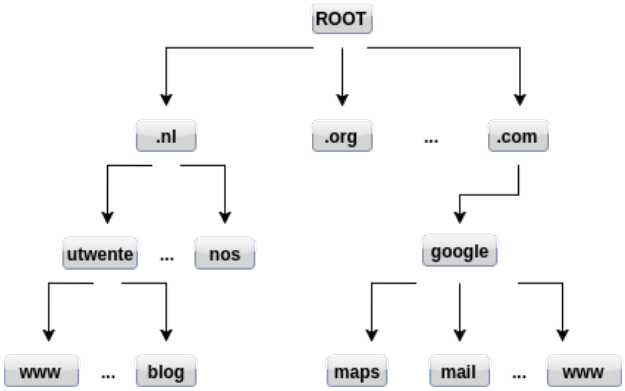

Figure 2.1: Tree shaped hierarchical structure of the DNS

An implementation of the DNS generally consists of three parts: a DNS client, a recursive caching name server (DNS resolver), and the authoritative name servers. The exact implementation can be different in various scenarios, as shown in figure 2.2 and 2.3.

The DNS client and DNS resolver both keep separate caches. When a requested DNS record is in the cache and the Time to live (TTL) has not expired, this record will be used; however, when the demanded record is not in any of both caches, recursive DNS resolving is required. A common scenario of how this could work is illustrated in figure 2.2:

1. A DNS client sends a DNS query, via the router, to the DNS resolver.
2. The DNS resolver requests the IP address of the corresponding TLD from the root server.
3. The root server provides the requested IP address.
4. The resolver requests the IP address of the desired Second-level domain (SLD) from the TLD authoritative name server.
5. The TLD name server provides the requested IP address.
6. The resolver requests the IP address of the sub domain from the SLD authoritative name server
7. The SLD server provides the requested IP address.
8. The DNS resolver provides the response to the requested query to the DNS client.

The scenario as illustrated in 2.2 is, privacy-wise, not optimal for the user. The recursive caching DNS server is generally owned by an ISP, or a large company such as

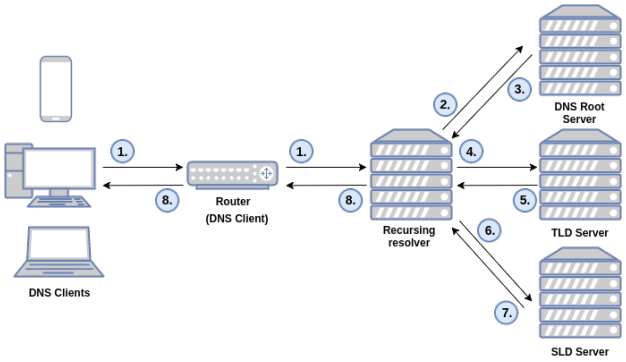

Figure 2.2: Common DNS implementation with external resolver

Google. Every IP address is asked from their servers from which the logs are stored.[2] Essentially, the user gives up the privacy of his web history. An implementation which provides significantly better privacy is presented in figure 2.3. A local device, such as the home router, functions as DNS resolver. Instead of sending the DNS queries via a company owned resolver, the router communicates directly with the authoritative name servers. This research focuses on this kind of DNS resolving; a constrained hardware architecture such as a home router as DNS resolver.

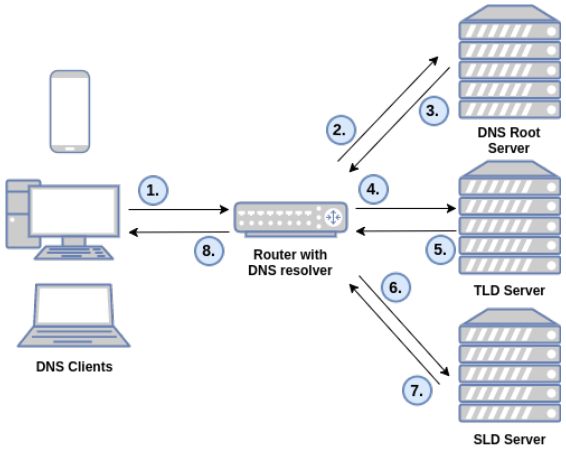

Figure 2.3: DNS implementation with home router as resolver

2.2 DNSSEC

The DNS protocol is vulnerable to certain attacks. DNS cache pollution, also referred to as DNS spoofing, can be used to introduce false DNS data in the resolver's cache. The result of DNS spoofing is, that traffic which was meant to be sent to a certain server, can be send to a server of someone with bad intents. The cause of DNS pollution is that the DNS communication is not authenticated with cryptographic primitives. To solve this problem, the DNS protocol is extended with the DNS Security Extensions (DNSSEC). DNSSEC protects DNS responses using digital signatures, to make it impossible to falsify the responses. Currently, the digital signatures used for DNSSEC are making use of the widely-used RSA cryptosystem. The disadvantages of this cryptosystem for DNSSEC have been extensively addressed in the introduction of this paper[3].

[2]https://developers.google.com/speed/public-dns/privacy/

2.3 Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is a way of public-key encryption based on elliptic curves over finite fields. This section focuses on aspects of elliptic curve cryptography that are relevant for the research in this paper. We simply use it as a tool for encryption, and keep its characteristics in mind. For an in-depth explanation of ECC, see [1].

The main advantage of an ECC-based cryptosystem is that it requires a smaller key, compared to RSA, for the same security in bits. Due to the increase of computer power, RSA needs increasingly larger keys. Non-ECC algorithms, such as RSA, are sub-exponentially growing in key size for a doubling of security in bits. Doubling the number of bits of security, requires more than a doubling of the key lengths. For ECC, this is not the case. In order to double the strength in bits, one needs double the key size. In table 5.3, a comparison between RSA and ECC, on required key length, is shown.

ECC as cryptosystem for DNSSEC could thus mitigate the two problems it currently has with RSA, due to the required key lengths. Due to the difference in key-size growth, the switch to an ECC-based cryptosystem for DNSSEC is becoming increasingly crucial. At this moment, the general consensus is that 128-bit security is enough for personal use[8]. If 256-bit security will be needed, due to growing computer power, the difference between RSA and ECC is a 15360-bit key length versus a 512-bit key length respectively. This shows that the need for a change in the DNSSEC cryptosystem for the future seems unavoidable[8].

Table 2.1: Required key length for different securities in bits for ECC and RSA

Security in bits	RSA Encryption method	RSA key in bits	ECC key in bits
80	3DES (2 keys)	1024	≥ 160
112	3DES	2048	≥ 224
128	AES-128	3072	≥ 256
256	AES-256	15360	≥ 512

3. RELATED WORK

In the paper [12] by van Rijswijk-Deij et al., researchers from the DACS group at the University of Twente presented the idea of ECC as a solution for problems in DNSSEC, caused by RSA. The performance impact of ECC for DNSSEC on DNS resolvers has also been studied. In the paper [5], also by van Rijswijk-Deij et al., the authors found that the performance impact of switching DNSSEC to ECC would not cause problems for DNS resolvers with a common server-class hardware architecture. Furthermore, in said paper, the authors came up with a method to calculate how many signatures a resolver should be able to verify, given a workload of initialized queries. We base our model on this work, to determine the presumable amount of verifications a home router should be able to verify.

Current research does not consider the impact of an ECC-based cryptosystem for constrained hardware architectures. This research specifically focuses on the constrained hardware architectures to determine if the switch of cryptosystems will be a problem for DNS resolvers on home routers, since they generally have a constrained hardware architecture. We thus will find out if the switch to ECC for DNSSEC will cause performance problems for the DNS set-up as shown in figure 2.3.

4. RESEARCH METHODS

This section presents the research methods used in the paper. It discusses the approach taken, to predict the impact an ECC-based cryptosystem has for DNSSEC on resolvers, with a constrained hardware architecture. In order to conclude anything about this impact, we need to determine how many DNS queries such resolvers should be able to handle. It is nearly impossible to execute a representative measurement, since this would require measurements on potentially thousands of DNS resolvers. Not only would this be extremely labour-intensive, but also difficult due to privacy reasons. Therefore, a well-founded worst-case scenario is constructed in section 4.1.
Subsequently, a method to benchmark ECC-based validation is presented in section 4.3.

4.1 First research question

A set of mathematical functions executed on a computer will fluctuate much less than the amount of ECC-verifications needed on constrained hardware architectures to function for DNSSEC. For every network, this number can be significantly different. In houses with ten or more people, much more DNS queries will be initialized than in the home of a single person.
For these reasons, we try to derive a realistic worst-case scenario, and see if this resulting amount can be met in the benchmark tests.

A model is adopted to determine the actual amount of signatures that a hardware architecture should be able to verify per second, to function for DNSSEC. The model is based on another model, proposed in a previous paper [5], in section III B. We use it to accurately predict the number of signatures verifications (S_v) a DNS resolver has to perform, given a certain workload in terms of queries (Q) it sends to the authoritative name servers.

In the earlier work [5], the input parameter Q is the number of queries that a DNS resolver sends upstream to authoritative name servers. Queries from individual clients are not considered. In contrast, in this work, we assume a worst-case approach, where every query from a client leads to a query from the resolver to upstream authoritative name server. In essence this means that we assume that the resolver does not gain any benefits from caching at all. The function of the model is as follows:

$$f : Q \rightarrow S_v \tag{1}$$

Function (1) was built upon different sub-equations[5], which are disregarded for this paper; however, we use the relation equations which consist of the sub-equations and their variables.
The variable R represents the number of responses from authoritative name servers. R_s represents the number of responses with a signature. Responses can have more than one signature, therefore, S are the actual amount of signatures. Not all signatures have to be verified, therefore, S_v represents the amount of signatures that need to be verified.

The authors of [5] examined representative empirical data of these parameters. By plotting these parameters against each other, they found an estimate for the correlation coefficients. The results of their research are linear relationships between all variables.
For the scope of this research, we use the worst-case scenario for the linear models. We avoid many risks by taking a worst-case scenario, because the future of DNSSEC is uncertain and we have no representative data from DNS resolvers on home routers. DNSSEC adoption could, theoretically, grow faster than the speed of DNS resolvers. If it turns out that using ECC for DNSSEC will be feasible in our worst-case scenario, it will likely stay so for the future. The model for the parameters R, R_s, S and S_v is defined as the following set of linear functions $f_1 ... f_4$ below, from the model [5]:

$$f_1 : R = \bar{r}Q + \beta 1 \qquad f_3 : S = \bar{s}R_s + \beta 3$$
$$f_2 : R_s = a_s R + \beta 2 \qquad f_4 : S_v = a_v S + \beta 4$$

with:
$\bar{r}$: the average number of responses per query
a_s: the fraction of responses with signatures
$\bar{s}$: the average number of signatures per response
a_v: the fraction of signatures that is validated

These functions can then be combined to obtain f:

$$f : S_v = aQ + b$$
$$a = a_v \bar{s} a_s \bar{r}$$
$$b = a_v(\bar{s}(a_s\beta_1 + \beta_2) + \beta_3) + \beta_4$$

S_v is an answer to the first key question of our research, the amount of ECC-verifications a constrained hardware architecture would presumably be able to verify, to function for DNSSEC. The parameters in the equations $f_1 ... f_4$ are determined using linear regression over empirical data. However, in this paper, a worst-case scenario is built based on a number of assumptions. The value for β in the linear equations are an artefact of the fact that [5] uses linear regression. Hence that in this paper we can set β to 0, as we build a hypothetical worst-case scenario.

For the other parameters, $\bar{r}$, a_S, $\bar{s}$ and a_v, a couple of assumptions are made for the worst-case values. For $\bar{r}$, the average number of responses from the authoritative server per initiated query, we take 1. This would mean that every query to an authoritative name server gets a response back. Since we want a worst-case scenario, we assume that every response contains signatures. Hence, for the fraction of responses with a signature (a_s) we also take 1. In previous research [5], the makers of the model found that signed responses contain 2.1 signatures per response on average. To avoid risk, we round this number up to 3 and use this for $\bar{s}$. Furthermore, we will assume that all the signatures have to be validated. Thus a_v, the fraction of signatures that have to be validated, is set to 1.

A very important part of DNS resolving is the cache of the resolver. Especially when a lot of DNS requests are made from the clients to the resolver, IP addresses will be still available in the cache; however, we ignore the cache to make our worst-case scenario even stronger. With these values for the parameters, an absolute worst-case situation is established to avoid risks. If it turns out that DNSSEC would work even in this worst-case scenario, it most definitely will in practise. By taking these values for the variables, and ignoring the cache, we can rewrite the function $f : S_v = aQ + b$ with $a = a_v \bar{s} a_s \bar{r}$ and for each $\beta = 0$ as follows:

$$f : S_v = 3Q + b$$
$$f : S_v = 3Q + a_v(\bar{s}(a_s\beta_1 + \beta_2) + \beta_3) + \beta_4$$
$$f : S_v = 3Q + 0$$

Thus, for every query Q the DNS resolver initializes, it requires the capability to validate three signatures. As said earlier, we suppose that every DNS request that comes from the clients, and goes to the resolver, will result in a query to the authoritative name servers (Q).

4.2 Ethical DNS scenario test

We attempt to establish a realistic amount for Q. We do this by configuring a DNS resolver and determine the maximum burst rate in a DNS scenario test. The DNS scenario test will be held in a home environment. A time is chosen for which the most people are connected to the network. The burst rates are probably the highest achievable in that period. Because people outside of our research will get involved, privacy issues should be held into account. We execute our test for a constrained period of time, and notify everyone involved that the DNS traffic on our network is monitored. Visitors can choose whether or not to use our network, providing informed consent if they do. After explaining what would happen with the obtained data, all the guests declared that they did not mind the monitoring.

4.3 Second research question

The question: "How do ECC implementations perform on the CPU architectures that are common in home routers?" is analysed. This is done through extensive benchmarking of various ECC algorithms and implementations on a constrained hardware architecture, similar to the architecture of a basic home router. For comparison, we also benchmark the same ECC algorithms and implementations on a server-class hardware architecture. The switch of the cryptosystem of DNSSEC not causing performance issues for server-class hardware architectures does not imply the same for constrained hardware architectures.

The constrained hardware architecture used for the analysis, is a single core of a Raspberry Pi 3. The CPU of this device is a quad core ARM Cortex-A53, clocked at 1.2GHz . The environment running on this architecture is the operating system (OS) Ubuntu Mate, completely stripped from its graphical user interface. The hardware architecture in combination with the lightweight OS is a representative set-up, comparable to a home router architecture.
For the server-class hardware architecture we use an Asus UX301LA having an Intel 4500U processor clocked at 1.8GHz.

For the ECC algorithms, we first pick one of the most used ECC validation algorithms: ECDSA. Both the 256-bits variant ECDSA P-256, as well as the 384-bit variant, ECDSA P-384 are benchmarked; however, Ed25519 is said to be faster, which could potentially change the answer to the main question. Hence, the curve Ed25519 in the Donna implementation is benchmarked as well. Perhaps Ed25519 is suitable for DNSSEC on constrained hardware architectures, while ECDSA turns out to be too slow. We benchmark the ECDSA algorithms in various versions of OpenSSL, because the implementation will influence the results. We take the, still widely used, legacy version OpenSSL 1.0.1f, the current version in Linux Ubuntu 16, OpenSSL 1.0.2k, and the state-of-the-art version, 1.1.0e.

Furthermore, the 1024, 2048 and 4096 bit-versions of RSA are benchmarked to verify that ECC signature verification indeed requires more CPU power than RSA signature verification. It also shows the relevance of this research question stating that the switch could indeed be problematic because the performance differences are significant.

The average values of the benchmarks are calculated over 100 independent speed tests. The corresponding standard deviations, are calculated over the 100 results. A single speed test consists of a 10-second run with continuous calls to signature verification functions, from which the average number of verifications, per second, is calculated.

5. RESULTS

This part first contains the results of the DNS scenario test executed as described in the previous section. Subsequently, we use this result in the model, constructed in the previous section, to answer our first research question. The section is concluded with the results from extensive benchmarking of ECC validation on two different hardware architectures.

5.1 DNS scenario test

Previously in this paper, we reasoned about a worst-case scenario for the amount of signatures a constrained hardware architecture should be able to verify to function for DNSSEC. Because the amount of DNS queries required per second is user-dependent, we tried to find a moment for which the amount of DNS requests would be as high as possible in our situation.

Tests were conducted in a student dorm, and the data was collected over a period of 24 hours. During this period, the number of occupants using the network peaked at 11 users. All the guests were asked to connect to our network with their smart phones. Besides 9 people using the network via their smart phones, a NAS was functioning and the TV was used for music on various applications via Chrome Cast. Two other students used the Internet on a laptop, until around 21:00 for studies, and later for amusement.
Since DNS monitoring is extremely privacy sensitive, all participants were asked for an informed consent as discussed in subsection 4.2.

Figure 4.1 illustrates the results of the DNS scenario test. It is clear to see that the event started around 20:00, and ended quite early, at 0:30. The next day, three students were working on their computers for their studies; however, as expected, the highest burst rate was already achieved at the time of the event, at around 21:45, with 58 DNS queries in one second. This value is used as input for the model, to find the amount of signatures that a DNS resolver should be able to validate, per second.

5.2 Signature validation model

The model proposed in the research methods section is used to find the amount of signature verifications that need to be achieved by the constrained hardware architecture, to function for DNSSEC. We found the following equation through inductive reasoning, in combination with substitution of worst-case values for the multipliers, in the original model.
All things combined provided the following function, as explained in subsection 4.1.

$$f : S_v = 3Q \tag{2}$$

For the value Q, a maximum burst rate of 58 queries per second is found. Substitution of this value in the equation (2), gives $S_v = 3 * 58 = 174$ signature verifications.
For convenience, and because the maximum burst rate could variate, this is rounded this up to 200 signature verifications per second that have to be achieved, in one second, for the worst-case scenario.

5.3 Benchmarks

Earlier researchers[5] have benchmarked ECC validation on a server-class hardware architecture. Since the benchmarks in [5] were performed, the state of the art in terms of implementations of ECC has improved. Therefore, a full new set of benchmarks is conducted, with the set of algorithms and implementations discussed in section 4.3.

Table 5.1: Signature verifications per second, on a single core ARMv8 @ 1.2GHz benchmarks

Implementation	RSA						Elliptic Curve Cryptography					
	RSA 1024-bit		RSA 2048-bit		RSA 4096-bit		ECDSA P-256		ECDSA P-384		Ed25519	
	mean	σ	mean	σ	mean	σ	mean	σ	mean	σ	mean	σ
OpenSSL 1.0.1f	4792.3	15.9	1317.4	3.1	343.2	0.6	243.2	2.0	98.2	0.7	-	-
OpenSSL 1.0.2k	7185.7	21.1	2060.2	2.3	554.7	1.4	312.5	3.4	101.2	0.8	-	-
OpenSSL 1.1.0e	10761.5	33.2	3525.5	10.1	1006.7	2.9	786.4	2.1	93.7	0.7	-	-
ed25519-donna	-	-	-	-	-	-	-	-	-	-	716.5	6.4

Table 5.2: Signature verifications per second, on a single core Intel i7 4500U @ 1.8GHz benchmarks

Implementation	RSA						Elliptic Curve Cryptography					
	RSA 1024-bit		RSA 2048-bit		RSA 4096-bit		ECDSA P-256		ECDSA P-384		Ed25519	
	mean	σ	mean	σ	mean	σ	mean	σ	mean	σ	mean	σ
OpenSSL 1.0.1f	87585.9	382.6	26349.4	64.3	7097.3	25.0	2413.2	25.3	1157.6	17.1	-	-
OpenSSL 1.0.2k	103624.0	593.2	32049.6	193.9	8764.8	36.7	8900.9	35.5	1162.6	13.3	-	-
OpenSSL 1.1.0e	100446.0	443.9	31845.9	122.6	8746.6	42.2	8606.3	53.9	1100.8	11.4	-	-
ed25519-donna	-	-	-	-	-	-	-	-	-	-	14960.1	125.1

Table 5.3: Multipliers of signature verifications, per second, differences between the architectures.

Algorithm	OpenSSL version			
	1.0.1f	1.0.2k	1.1.0e	X
RSA-1024	18.3	14.4	9.3	-
RSA-2048	20.0	15.6	9.0	-
RSA-4096	20.7	15.8	8.7	-
ECDSA P-256	9.9	28.5	10.9	-
ECDSA P-384	11.8	11.5	11.7	-
Ed25519	-	-	-	20.9

The results of the ECC validation benchmarking on both hardware architectures are shown in table 5.1 and 5.2. In table 5.1, the benchmark results of the constrained hardware architecture are shown. In table 5.2, the benchmark results of the server-class hardware architecture are presented.

In table 5.3, the comparison between the differences among the constrained and server-class hardware architectures is illustrated. Each value represents the multiplier of how many more signature verifications per second the server-class hardware architecture achieved, on average, for every algorithm and implementation. The reason for comparison of architecture speeds is because optimization conclusions from this table can be drawn. This will be discussed in the subsection 6.1, where we go more in-depth on the results.

Examining the results, the first thing that stands out is the huge difference between the amounts of signature validations between the different hardware architectures. As expected, the constrained hardware architecture is much slower, despite the quite limited difference in clock speeds. Another noticeable fact is the occurrence of large differences between one implementation and another, on the same hardware architecture, with the same algorithm. The difference between ECDSA P-256 in

OpenSSL 1.0.1f and in OpenSSL 1.1.0e, for example. The determined means are respectively 243.2 and 786.4 verifications. The latter value would be sufficient, even in a worst-case scenario, while the first would certainly not. Promising results are the mean of ECDSA P-256 and Ed25519 for the constrained hardware architecture. The results are respectively 786.4 for ECDSA P-256 with OpenSSL 1.10e and 716.5 for Ed25519. We go more in depth on these two ECC algorithms in section 6.

Currently, it is clear that ECDSA P-384 performs insufficient on the constrained hardware architecture, based on the worst-case scenario as presented in subsection 5.2; however, this implementation is not yet optimised for both x86 and ARM architectures.
Looking at the enormous performance improvements shown in previous optimizations, this could change in the future in combination with possible faster home routers.

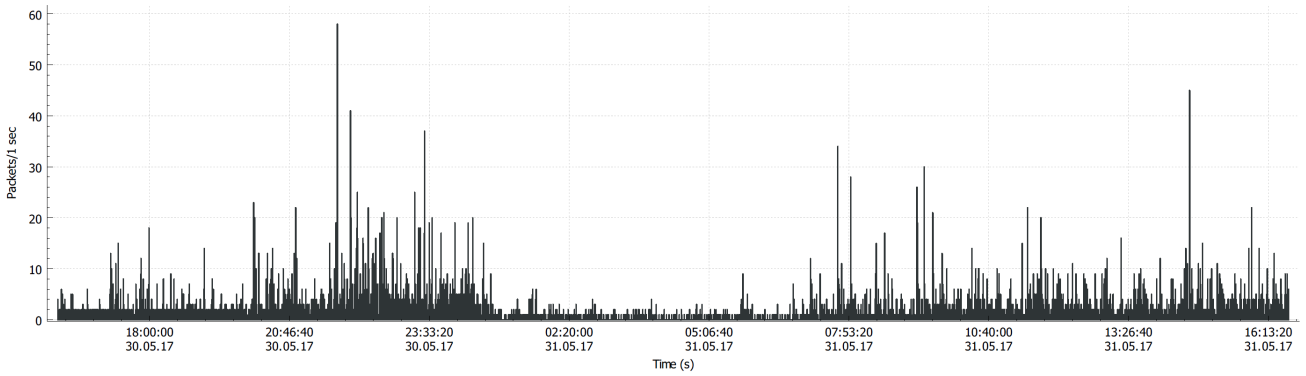

Figure 4.1: Graph of DNS queries per second over 24 hours

6. DISCUSSION

In this section, first the results obtained through our research are discussed and looked in more deeply.
Subsequently, the research methods used to obtain these results are discussed and seemingly potential weaknesses in our research methods are shown.

6.1 Results

Mentioned in the previous section, there are significant inequalities among the differences between the benchmarked architectures, as illustrated in table 5.3.
Since OpenSSL 1.0.1f, there have been various optimizations, both for x86 and ARM architectures. As illustrated in table 5.3, the differences between the architectures approach towards an order of a magnitude; however, the difference of Ed25519 between the ARM architecture and the x86 architecture is 20.9. The Ed25519 implementation is optimised for the x86 architecture, but not for the ARM architecture. We know this from the results of the server-class hardware architecture and the underlying mathematical primitives of Ed25519. The focus of this elliptic curve is on performance, which is clearly visible in table 5.2, illustrating the difference between ECDSA P-256 and Ed25519, on an x86 architecture .

Provided the state-of-the-art version of OpenSSL – 1.1.0e, is used, the amount of signature validations per second for ECDSA P-256 is roughly $8,600$ while the amount for Ed25519 is roughly $15,000$. Hence, an optimised implementation of the Ed25519 algorithm can be up to 1.5 times faster than the ECDSA P-256 algorithm. The Ed25519 version for ARM is not optimal, because otherwise the difference between the architectures would be around an order of a magnitude, just like the other algorithms.

Assuming that an optimised implementation of Ed25519 for ARM also outperforms ECDSA P-256 by a factor of 1.5x, this would yield a maximum of 1074.4 signature verifications per second. This is significantly higher than the already optimised ECDSA P-256 algorithm for ARM architectures, which achieves 768.4 signature verifications per second.

In the benchmarks, the full capacity of the CPU was available for signature verification, while in reality this will not be the case; however, even if only half of the CPU capacity is available, the results of this research are still the same.

6.2 Research methods

In this work, we have used the maximum burst rate from a single worst-case DNS scenario that was achievable in our case; however, in reality this amount could differ and potentially be higher. Nonetheless, this will not result in a different answer to the main question of this research.

We found that in the worst-case scenario, the architecture should be able to handle about 200 signature validations per second. Even if the burst rate would be doubled, it is still achievable with the right implementation and algorithm to use an ECC-based cryptosystem for DNSSEC.

Another important aspect to realize is the assumption that the cache of the resolver does nothing; however, the cache is the most efficient during bursts, because clients ask in many situations for the same IP addresses, which are already stored in the cache. In the conclusion section, we go more in depth into what the answer on our research question would be.

7. CONCLUSION

In this work, we analysed the question "will switching the cryptosystem of DNSSEC to an ECC-based one lead to performance issues for DNS resolvers on a constrained hardware architecture?". The main question is split up in two different parts, with two corresponding research questions. During the research, we found that the first research question: "How many signatures would a home router presumably have to verify in a worst-case scenario per second?", consists of two segments. The first part covers how many queries are initiated by a DNS client in a realistic scenario, at the maximum burst rate. The second part determines how many signatures validations this would require for the DNS resolver to handle in one second.
Based on a real-world measurement in a student dorm, and based on the worst-case scenario we established, we find that a resolver on a constrained platform should be able to perform at least 200 signature validations per second.

Through benchmarking on multiple architectures with various implementations and algorithms, we can determine that the established amount of signature validations per second, is achievable on a constrained hardware architecture.

However, certain implementation choices are necessary.
First of all, DNSSEC signers should use a suitable signing algorithm. The recommended algorithm for DNSSEC validation is either ECDSA P-256 or Ed 25519. Secondly, the algorithm implementation choice is crucial. ECDSA P-256 requires the state-of-the-art version of OpenSSL — 1.1.0e, to function for DNSSEC validation. For Ed 25519, the current implementation is sufficient, but further improvements can be made for the ARM architectures implementation. We find that the constrained architecture we benchmarked can perform around 786 signature validations per second for the commonly used ECDSA P-256 signature algorithm. This amount could be even higher for an optimised implementation of Ed25519 for ARM architectures.

The final conclusion is that switching to an ECC-based cryptosystem will not lead to performance issues for DNS resolvers on a constrained hardware architecture, provided that certain implementation choices are made. Our hypothetical worst-case scenario was quite extreme. We assumed no caching, a response for every initiated query, 100% DNSSEC employment, more than average signatures per response and only half of the CPU power available for signature verification. Despite these extreme worst-case assumptions, there were no performance issues, provided that certain implementation choices were met.

8. FUTURE WORK

For future work it would be of great value if a larger set of DNS resolvers on home routers could be tested, in order to find an average maximum burst rate, which can be used to predict more precisely how many signature validations such an architecture should be able to perform. This set of data, in combination with an optimised version of ECDSA P-384 for ARM architectures, could result in an even better DNSSEC suite. If, for example, it turns out that in 99% of the cases an optimised version of ECDA P-384 performs well enough for DNSSEC.
Currently, we do not have the right amount of empirical data on home resolvers to form such a conclusion.

9. REFERENCES

[1] M. Anoop. Elliptic curve cryptography, An Implementation Guide. 2007.

[2] S. M. Bellovin. Using the domain name system for system break-ins. *AT&T Bell Laboratories*, 1995.

[3] D. Eastlake. Domain Name System Security Extensions. *IETF RFC*, 2535, March 1999.

[4] K. Hageman. The Performance of ECC Algorithms in DNSSEC: A Model-based Approach. October 2015.

[5] K. Hageman, R. van Rijswijk-Deij, A. Pras, and A. Sperotto. The Performance Impact of Elliptic Curve Cryptography on DNSSEC Validation. *IEEE/ACM Transactions on Networking*, 2016.

[6] N. Jansma and B. Arrendondo. Performance comparison of elliptic curve and rsa digital signatures. April 2004.

[7] D. Kaminsky. Black Ops 2008: It's the End of Cache As We Know It. *Black Hat USA*, July 2008.

[8] National Institute of Standards and Technology. Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms. *NIST Special Publication 800-175B*, August 2016.

[9] G. van den Broek, R. van Rijswijk-Deij, A. Pras, and A. Sperotto. DNSSEC meets real world: dealing with unreachability caused by fragmentation. *IEEE Communications Magazine*, 52(4):154–160, April 2014.

[10] R. van Rijswijk-Deij. Improving DNS Security, a Measurement-based Aproach. *Ph.D. thesis*, June 2017.

[11] R. van Rijswijk-Deij, A. Pras, and A. Sperotto. DNSSEC and its potential for DDoS attacks - a comprehensive measurement study. *ACM Internet Measurement Conference*, November 2014.

[12] R. van Rijswijk-Deij, A. Pras, and A. Sperotto. Making the Case for Elliptic Curves in DNSSEC. *ACM SIGCOMM Computer Communication Review (CCR)*, 45(5), October 2015.