# Research Project 1 - DNSCurve Analysis

Michiel Timmers
michiel.timmers@os3.nl,
System and Network Engineering,
Universiteit van Amsterdam
Netherlands

In cooperation with ON2IT B.V.

February 2, 2009

# 1 Abstract

Our Internet traffic depends highly on the Domain Name System (DNS) protocol. In the recent past it's been shown that there are some serious problems with this protocol that are now known as the "Kaminsky DNS bug" [1]. The adoption of software, like DNSSec, that must put a halt to these kind of problems is very modest. A possible explanation for this slow deployment is the administrative complexity of its implementation. DNSCurve is a new kind of implementation that focuses on DNS security. It uses a very strong elliptic-curve encryption [2]. One of the strong points for its adoption, claimed [3] by its developer D. J. Bernstein [4], is the easy adoption of DNSCurve for software authors and the easy deployment for system administrators.

## 2 Acknowledgments

I would like to thank Jeroen Scheerder of ON2IT B.V. [5] for his assistance, feedback and tips during this research project. His knowledge was very useful.

# Contents

# 3 Introduction

DNSCurve uses link-level public-key protection for the protection of DNS packets. This means that all DNS traffic is encrypted using a strong elliptic-curve encryption. By encrypting all DNSCurve traffic it would protect our Internet at a much higher level. DNSCurve uses a bottom-up approach as a deployment model, this means that if you protect two DNS servers using DNSCurve you are completely secure between these two sites. Other DNS protection implementation demands that al DNS servers in the tree above you are protected as well, See section ”Comparison with DNSSec” for more information

## 3.1 Research question

This project is based on the following research question:

> Is it possible to describe a acceptation model that will give clear view of the adoption of DNSCurve?

To make this acceptation model I have looked at the DNSCurve architecture, models and possible improvements in the source code. There is also a small section that looks at DNSSec and how its being deployed at the moment.

# 4 What is the problem with DNS

In July 2008 CERT announced that Dan Kaminsky, a security researcher, had found problems in the DNS protocol. These problems are fundamental mistakes that where made in the DNS protocol, this meant that all software implementation that used some kind of DNS service is effected. Dan Kaminsky worked in secret with all major platforms (like Windows, Cisco, Nominum and BIND) to fix and patch these flaws and made all patches available at the same day. After about 30 days of the publication of the patches Dan Kaminsky published the some details of the flaw but without point out the exact problem because this would still be to dangerous. DNSCurve is designed to protect our DNS servers for these kinds of possible flaws.

# 5 What does DNSCurve

DNSCurve encrypts the DNS traffic at a link layer level so that every DNS packet will be protected. The main developer of DNSCurve, Daniel J. Bernstein, also developed the eliptic-curve packet that DNSCurve uses for its encryption and decryption. This encryption library is called NaCl (pronounced "salt") and has a public website where the code can be downloaded and information can be found about the usage of this encryption library. The website has the following quote on its main page that describes the current implementation status:

> Are you sure that you're one of the people who's supposed to be reading these pages? The NaCl release is close but hasn't happened yet; some essential verification tasks still have to be done. Caveat lector.

As you can see the NaCl library that DNSCurve uses isn't finished yet but you can use it if you wish and there is some basic information available on how to use it. Unfortunately, this can't be said about DNSCurve.

## 5.1 Current DNSCurve status

At this moment the source code of DNSCurve isn't available at a publicly known location. Because my project coach Jeroen Scheerder is one of the four developers at the DNSCurve project he could supply me with the location for the current code. This code dates from 18 September 2008 (Almost 6 month before this project) and doesn't have any documentation what-so-ever. Another problem that was discovered during this project is that the NaCl library was being further developed while the DNSCurve wasn't. This caused problems compiling the DNSCurve code. See more about this in the chapter "DNSCurve installation and modifications". As you can see the DNSCurve is currently far from being released and this had a big impact on this project because a good implementation using DNSCurve wasn't doable.

# 6 DNSCurve architecture

The DNSCurve architecture is based on the usage of the NaCl library. The code of DNSCurve consists of the following files:

```
michiel@michielLT:~/agl-dnscurve-d069c503cabfbdb7657e73a9798b9c21593ce715/forward$ ls -all
total 84
drwxr-xr-x 2 michiel michiel  4096 2008-09-19 01:26 .
drwxr-xr-x 5 michiel michiel  4096 2008-09-19 01:26 ..
-rw-r--r-- 1 michiel michiel  1826 2008-09-19 01:26 base32.c
-rw-r--r-- 1 michiel michiel   306 2008-09-19 01:26 base32.h
-rw-r--r-- 1 michiel michiel   476 2008-09-19 01:26 base32-test.c
-rw-r--r-- 1 michiel michiel  1479 2008-09-19 01:26 dnscurve-keygen.c
-rw-r--r-- 1 michiel michiel  2657 2008-09-19 01:26 dnscurve-test-client.c
-rw-r--r-- 1 michiel michiel  2969 2008-09-19 01:26 dns.h
-rw-r--r-- 1 michiel michiel  7719 2008-09-19 01:26 dns_packet.c
-rw-r--r-- 1 michiel michiel  1055 2008-09-19 01:26 dns_random.c
-rw-r--r-- 1 michiel michiel   530 2008-09-19 01:26 dns_random.h
-rw-r--r-- 1 michiel michiel 16272 2008-09-19 01:26 forward.c
-rw-r--r-- 1 michiel michiel   263 2008-09-19 01:26 ip_parse.c
-rw-r--r-- 1 michiel michiel    92 2008-09-19 01:26 ip_parse.h
-rw-r--r-- 1 michiel michiel  1353 2008-09-19 01:26 Makefile
-rw-r--r-- 1 michiel michiel   390 2008-09-19 01:26 randombytes.c
-rw-r--r-- 1 michiel michiel  2047 2008-09-19 01:26 udpserver.c
```

These files are used to make the forwarder functional. How to compile and uses these files is described in the chapter ”DNSCurve installation and modifications”. But there is one executable that I want to point out at this moment, and that is ”dnscurve-keygen”. This executable generates a public a private key using the NaCl library. Its claimed [7] that the NaCl library is able of ”computing Curve25519 shared secrets for ten million servers takes under ten minutes of computation on a Core 2 Quad”. I have tested this while using a simple laptop configuration (Intel 1,83 GHz, 2 MB L2-cache, 667 MHz FSB CPU) using a simple Perl script. Before making the queries I modified the dnscurve-keygen.c file so that it wouldn't print all keys to the console because normally you would write this to some cache. As shown in the following example it takes about 30 minutes to generate 1 million keys.

```
Enter number of keys to generate: 1000000
Starting to generate 1000000 keys

Finished in:1970 wallclock secs (15.47 usr 220.42 sys + 660.37 cusr 851.44 csys = 1747.70 CP
```

See appendix 11.1 for the Perl code.

Figure 1 shows the DNSCurve files and there connection with each other and the NaCl library.

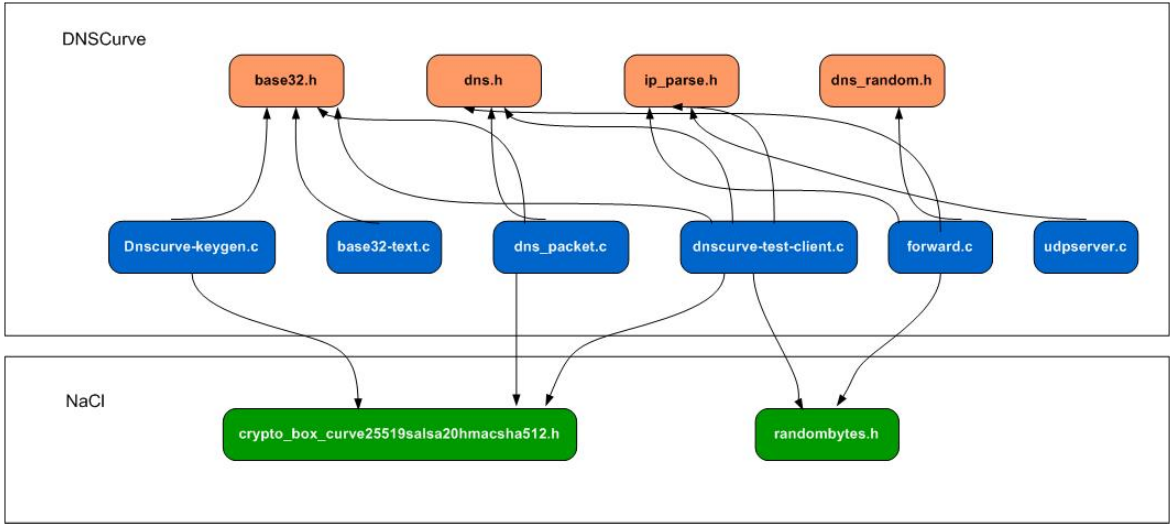

Figure 1: Relations in the DNSCurve forwarder source code

# 7 Implementation models

System administrator that want to rollout DNSCurve in there environment need to upgrade there DNS (recursive) cache as well as there DNS authoritative server.

**DNS cache (recursive server):** If you need to upgrade your DNS cache to a DNSCurve cache you need to replace the entire cache server. It's not possible to put some DNSCurve server in front of your DNS cache like its possible with a DNS authoritative server. A implementation of such a server is not available at this moment.

**DNS server (authoritative server):** A DNSCurve authoritative server can be implemented in two basic configuration. You can use a DNS server that can handle DNSCurve queries (see the "os3.nl" server in figure 2) or you can install a DNSCurve forwarder in front of your normal DNS server (see the "nl" server in figure 2). This first implementation of DNSCurve using a DNS server which can handle DNSCurve queries is still far from being realized. The later is currently under development and on this implementation the rest of this document is based. Figure 2 show the possible implementation models that would become available as DNSCurve becomes more mature.

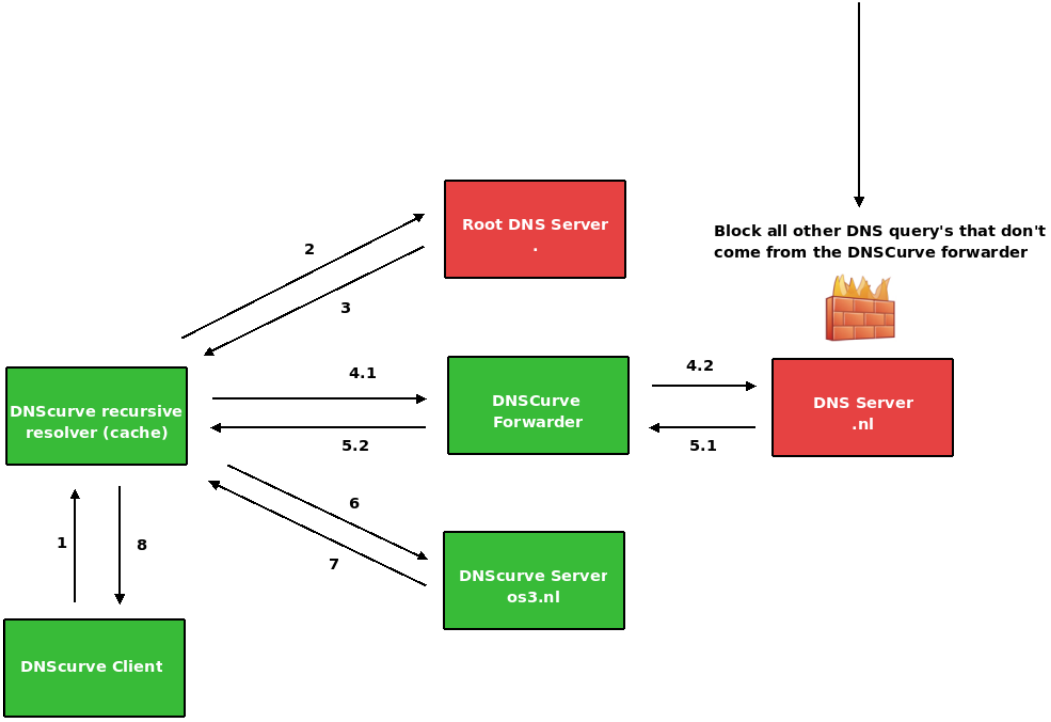
Root DNS Server
.
Block all other DNS query's that don't come from the DNSCurve forwarder
2
3
DNScurve recursive resolver (cache)
4.1
5.2
DNSCurve Forwarder
4.2
5.1
DNS Server .nl
6
7
DNScurve Server os3.nl
1
8
DNScurve Client

Figure 2: A partial DNSCurve environment

```
1   DNSCurve query - What is the IP address of www.os3.nl
2   Normal DNS query - What is the IP address of www.os3.nl
3   Normal DNS reply - Don't know but here is the name server of .nl
4.1 DNSCurve query - What is the IP address of www.os3.nl
4.2 Normal DNS query - What is the IP address of www.os3.nl
5.1 Normal DNS reply - Don't know but here is the name server of os3.nl
5.2 DNSCurve reply - Don't know but here is the name server of os3.nl
6.  DNSCurve query- What is the IP address of www.os3.nl
7.  DNCurve reply - Here is the IP address of www.os3.nl
8.  DNSCurve reply - Here is the IP address of www.os3.nl
```

Because the forwarder will forward DNSCurve queries and normal DNS queries there is no need to accept queries that don't come from the DNSCurve forwarder and you can block these kind of queries as shown in figure 2. Because DNSCurve encrypts the DNS data at a link level it would only require two DNSCurve services to make use of DNSCurve protection.

## 7.1 Step-by-step rollout

To deploy a DNSCurve forwarder in your environment you must follow the following steps:

1. Install the DNSCurve forwarder on a new IP address. If you install the forwarder on the same computer as your existing DNS server then you need to put it on a different IP address from the existing DNS server.

2. Configure the DNSCurve forwarder to forward to your existing DNS server's IP address.

3. Add, in your DNS data, a special DNSCurve server name for the DNSCurve forwarder. The name is specific to this DNSCurve forwarder and is automatically generated during installation of the forwarder.

4. Add the same DNSCurve server name in your parent DNS data.

5. After a week, remove the old non-DNSCurve server names.

The above steps can be found on the DNSCurve website. The following shows a example regarding step 4 using a BIND format.

```
;$ORIGIN mtimmers.practicum.os3.nl
$TTL 900
@     IN     SOA    uz5ipeefklaritof8fsusghdr183csnt8i779mpsih9p85cn79r2pg.mtimmers.
                    practicum.os3.nl.     hostmaster (
                    2008091702 ; serial
                    21600      ; refresh after 6 hours
                    3600       ; retry after 1 hour
                    604800     ; expire after 1 week
                    86400 )    ; minimum TTL of 1 day

      IN     NS     uz5ipeefklaritof8fsusghdr183csnt8i779mpsih9p85cn79r2pg.mtimmers.
                    practicum.os3.nl.
      IN     NS     uz5sh0k6d0t87le4jbhituig39hgfm3u3blrt6rcds3pmsdhtq1qri.mtimmers.
                    practicum.os3.nl.

      IN     MX     10     mail1.mtimmers.practicum.os3.nl.
      IN     MX     10     mail2.mtimmers.practicum.os3.nl.

             IN     A       145.100.104.21

uz5ipeefklaritof8fsusghdr183csnt8i779mpsih9p85cn79r2pg
             IN     A       145.100.104.21
uz5sh0k6d0t87le4jbhituig39hgfm3u3blrt6rcds3pmsdhtq1qri
             IN     A       145.100.104.21
mail1      IN     A     145.100.104.21
mail2      IN     A     145.100.104.21
server      IN     A       145.100.104.21

www          IN     CNAME   server
ftp          IN     CNAME   server
```

So one of the things that DNS administrators must keep in mind is the long name-server names that are used. These long names must be tested in your environment and with your registrar to see if any problems could arise. Daniel J. Bernstein tested this by registering the domain ”testingalongservernamewithdigits0123456789andlettersyz.ns.dnscurve.org” with a registar successfully. Although a label inside a domain name can officially be 256 characters in length according to RFC1035 [8], users must test this beforehand using there own registar to avoid any problems.

# 8 DNSCurve installation and modifications

The following sections describes on how to install a DNSCurve forwarder and the modifications that I had to make during this project to make it work (See appendix 11.2). It's very important to understand that the DNSCurve, at the time of writing this document, isn't finished and no documentation is available about how to use it. Because of this all the following about how to install and use the DNSCurve forwarder was done by studying the source code.

For this project I have tried to install a DNSCurve forwarder to see how the current implementation is. The first thing that you have to do is to install the NaCl library.

```
    wget http://hyperelliptic.org/nacl/nacl-20081203.tar.bz2
    bunzip2 < nacl-20081203.tar.bz2 | tar -xf -
    cd nacl-20081203
    ./do
```

After this you have to copy the header files that where compiled to your include directory. These header files are used by the DNSCurve software:

```
cd build/*/include
mkdir /usr/include/nacl
cp * /usr/include/nacl/
```

And now comes the problem that will show that DNSCurve is still in development and that this caused that this project couldn't install a fully workable DNSCurve forwarder. The problem is that there are some variables in DNSCurve that are not recognized by NaCl, this is because NaCl was further developed while DNSCurve was not. It will generate the following error after trying to compile the code with "make":

```
gcc -Wall -ggdb -std=c99 -c forward.c
forward.c: In function dns_reply:
forward.c:307: error: crypto_box_curve25519salsa20hmacsha512_AUTHBYTES undeclared (first
use in this function)
forward.c:307: error: (Each undeclared identifier is reported only once
forward.c:307: error: for each function it appears in.)
forward.c:308: error: crypto_box_curve25519salsa20hmacsha512_EXTRABYTES undeclared (first
use in this function)
make: *** [forward.o] Error 1
```

After studying the code I have made a guess in which variables the crypto_box_curve25519salsa20hmacsha512 and crypto_box_curve25519salsa20hmacsha512_EXTRABYTES where changed. This was just a guess and it could be wrong but my project coach and I thought this was the most plausible choice. I changed them to and using the following commands in the "forward" directory.

```
sed -i 's/crypto_box_curve25519salsa20hmacsha512_AUTHBYTES/
crypto_box_curve25519salsa20hmacsha512_PUBLICKEYBYTES/g' *

sed -i 's/crypto_box_curve25519salsa20hmacsha512_EXTRABYTES/
crypto_box_curve25519salsa20hmacsha512_NONCEBYTES/g' *

sed -i 's/crypto_box_curve25519salsa20hmacsha512_ref_AUTHBYTES/
crypto_box_curve25519salsa20hmacsha512_ref_PUBLICKEYBYTES/g' *
```

So after making the compile statement "make" a second time to code compiles without any problem. This has created the following executables:

**dnscurve-keygen** For generating a public an private key that DNSCurve can use. This executable doesn't require any further input from the user.

**dnscurve-test-client** The purpose of this "test-client" is not fully known. One might think that this is a client to send DNSCurve queries to test. However, the output that it generates isn't normal clear text and it doesn't matter if the DNSCurve forwarder can't be reached. "Usage: ./dnscurve-test-client <target ip> <target port> <target public key>". Studying the code shows that it will try to query www.google.org.

**forward** This is the forwarder that will forward the DNS queries (Both normal and DNSCurve packets) to the back-end DNS server. "Usage: ./forward <DNS server IP>"

**udpserver** A UDP server that will listing on port 53 for queries "Usage: ./udpserver <IP address> <UDP port number> <child process and args>"

To start the DNSCurve forwarder the first thing that needs to be done is to generate a public and a private key that the forwarder can use. This can be done by using the "dnscurve-keygen" executable. Once the public and private key are generated the private key must be set as a environment variable using the following statement:

```
export DNSCURVE_PRIVATE_KEY="<PRIVATE KEY>"
```

After this has been done the forwarder can be started using the following command:

```
./udpserver 0.0.0.0 53 ./forward <IP address>
```

Ok, now we have the forwarder up and running and its time to do some testing. A test that can be done is to send a normal DNS lookup to the forwarder and see if it will respond.

```
dig www.os3.nl @localhost

; <<>> DiG 9.5.0-P2 <<>> www.os3.nl @localhost
```

```
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 38130
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.os3.nl. IN A

;; ANSWER SECTION:
www.os3.nl. 0 IN CNAME info4u.os3.nl.
info4u.os3.nl. 53656 IN A 145.100.96.70

;; Query time: 23 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Mon Feb  2 00:18:14 2009
;; MSG SIZE  rcvd: 65
```

Now we know that the DNSCurve forwarder works for normal DNS queries it would be nice to test it using DNSCurve queries. Unfortunately, as stated before, the test client (which probably is ”dnscurve-test-client”) doesn’t give any useful information if we execute it, the only thing we know is that it tries to query www.google.org as its coded in the source file. The thing to do here is to see if we can make the DNSCurve forwarder use some kind of logging as its running and receiving queries. No logging is currently available in any of the DNSCurve programs. I have modified the forward.c code so it will log information to the syslog daemon (see appendix 11.2). The messages that are sent to the syslog daemon are:

**DNSCurve: Starting forward: <IP address>** Simple messages that the forwarder is started and that its forwarding to a specific IP address;

**DNSCurve: Plain DNS packet forward** Messages that a normal DNS packet has been forwarded;

**DNSCurve: DNSCurve packet forward** Messages that a DNSCurve packet has been forwarded;

**DNSCurve: Invalid DNSCurve packet** Messages that a there was a invalid DNSCurve packet and that it has been dropped.

To test if the syslog would work, and of course more importantly if it would receive any DNSCurve queries I looked at the messages that the syslog daemon had created.

```
Jan  29 00:17:53 michielLT dnscurve[3820]: DNSCurve: Starting forward: 212.142.28.66
Jan  29 00:18:13 michielLT dnscurve[3820]: DNSCurve: Plain DNS packet forward
```

Here you can see that the server was started and a few seconds later received a normal DNS packet. The dnscurve-test-client was also tested but this didn't generate any log. This indicates that there was never a proper DNSCurve packet received by the forwarder. My assumption is that the NaCl variables that where changes in the DNSCurve code where not right. I have changed them to other plausible choices but that didn't had any affect on the outcome. So unfortunately I couldn't test if the forwarder worked with DNSCurve queries, but by making it all the way so that it would answer to plain DNS queries was already a achievement because of the lack of any cohesive documentation.

# 9 DNSCurve and DNSSec

The DNSCurve project is not the only project that handles DNS security. DNSSec has been around for a while but till now it has not been deployed widely. DNSSec adds public-key signatures to DNS records while DNSCurve add protection at link level communication, some basic difference between DNSCruve and DNSSec is available on the DNSCurve website [6]. Both offer different security goals that can be complementary to each other. But as mentioned DNSSec isn't widely deployed till this day. Here are some thoughts about the DNSSec deployment and how its different from DNSCurve.

**Top-Down Versus Bottom-Up** DNSSec is a top-down approach while it comes to security. This means that a DNSSec server can't be fully secure if the ones above it aren't talking DNSSec. And at this time the root zones are still not signed using DNSSec. These root servers are mostly under a more bureaucratic organization than the ones that are lower in the tree, this is for good reasons but adaption of new improvements is very slow. DNSCurve uses a so called bottom-up approach. This means that is you have to two DNSCurve services that the communication between will always be secure. This is a much more efficient way of deploying because you don't need to care about DNS servers that are higher in the DNS tree.

**DNSSEC suicide** With DNSSec you need to generates new signatures every month, if you don't do this you zone will be considered as not being secure and thus "drops" of the Internet. Dr. Bernstein names this behavior DNSSEC Suicide. Compared to DNSSec the keys that DNSCurve uses keys that are generated only once. Only if the private key is compromised the administrator needs to generate a new key.

# 10 Recommendations

Bellow are recommendations that will improve the adoption of DNSCurve.

**Finish the code and make it publicly available:** This is one of the most imported elements regarding DNSCurve at this time. With the discovery of the Kaminsky DNS bug the public demands a more secure way in using DNS. While DNSSec is currently getting more and more publicity, by the time that DNSCurve is finished DNSSec could be widely deployed. If that would be the case then there wouldn't be that much pressure anymore for implementation another security feature in DNS.

**Documentation regarding installation** This document was writing while there wasn't any documentation available regarding installation of DNSCurve which made it very time consuming. If the DNSCurve will become available to the public some basic instructions about how to install DNSCurve would be useful

**Code improvement:** The NaCl library is being developed while the DNSCurve code isn't. The first thing is to make the DNSCurve software in sync with its library. After this the dnscurve-test-client could be improved so that it would support variable queries (and not only www.google.org) and report some sane output. For the forwarder it would be advisable to start handling logging and reporting to a syslog server.

Ok, lets get back to the research question:

> Is it possible to describe a acceptation model that will give clear view of the adoption of DNSCurve?

Someone could argue that this report has been done premature and that DNSCurve is not ready for any kind of study. I think that the most important thing is that it't time to act now! This document is not to early, DNSCurve is going to be to late if DNSSec if being widely deployed and that public awareness of DNS security is going to fade away. However, if DNSCurve is ready for deployment it could go very fast because of its bottom-up approach.

## 10.1 Further research

**DNSCurve and DNSSec cache comparison** Both DNSCurve and DNSSec change the DNS cache. Research on how these two caches work and if it would be possible to make a cache that support both protocols would be advisable.

**How to make a DNS server support DNSCurve** This document mainly focused on how DNSCurve work with the use of a forwarder. If DNSCurve would become more popular there will be a demand of normal DNS software that would support DNSCurve. Some documentation about this is already available on the DNSCurve website.

# 11 Appendix

## 11.1 generate-keys

```
   #!/usr/bin/perl

# Michiel Timmers (michiel.timmers AT os3.nl)
# Project RP1: DNSCurve Analysis
# Website: https://www.os3.nl/2008-2009/students/michiel_timmers/rp1


use Benchmark;


print "Enter number of keys to generate: ";
chomp( my $n = <STDIN> );

print "Starting to generate $n keys\n";

# start timer
$start = new Benchmark;
$start2 = time();

$i=0;
while ($i < $n){
 system("./dnscurve-keygen");
 $i++;
}
print "\n";

# end timer
$end = new Benchmark;
$end2 = time();

# calculate difference
$diff = timediff($end, $start);


print "Finished in:",timestr($diff, ’all’)," seconds\n";
print "Finished in: ",($end2 - $start2)," seconds\n";
```

## 11.2 Code modifications

```
2d1
< #define _GNU_SOURCE
24,25d22
< #include <syslog.h>
<
310,311c307,308
<   if (8 + length + crypto_box_curve25519salsa20hmacsha512_PUBLICKEYBYTES +
<       crypto_box_curve25519salsa20hmacsha512_NONCEBYTES >
---
>   if (8 + length + crypto_box_curve25519salsa20hmacsha512_AUTHBYTES +
>       crypto_box_curve25519salsa20hmacsha512_EXTRABYTES >
321c318
<     length + 8 + 8 + crypto_box_curve25519salsa20hmacsha512_PUBLICKEYBYTES;
---
>     length + 8 + 8 + crypto_box_curve25519salsa20hmacsha512_AUTHBYTES;
452,457c449
<           dns_forward(buffer, n, efd, &sin, txid, 0, NULL, NULL, NULL, 0);
<    // write a message to syslog about none DNSCurve packet
<         openlog ("dnscurve", LOG_CONS | LOG_PID | LOG_NDELAY, LOG_LOCAL1);
<         syslog (LOG_NOTICE, "DNSCurve: Plain DNS packet forward");
<           closelog ();
<
---
>           dns_forward(buffer, n, efd, &sin, txid, 0, NULL, NULL, NULL, 0);
460,464d451
<    // write a message to syslog about invalid DNS curve packet
<         openlog ("dnscurve", LOG_CONS | LOG_PID | LOG_NDELAY, LOG_LOCAL1);
<         syslog (LOG_ERR, "DNSCurve: Invalid DNSCurve packet");
<           closelog ();
<
469,472d455
<    // write a message to syslog about valid DNS curve packet
<         openlog ("dnscurve", LOG_CONS | LOG_PID | LOG_NDELAY, LOG_LOCAL1);
<         syslog (LOG_NOTICE, "DNSCurve: DNSCurve packet forward");
<           closelog ();
545,550d527
<   openlog ("dnscurve", LOG_CONS | LOG_PID | LOG_NDELAY, LOG_LOCAL1);
<   syslog(LOG_WARNING, "DNSCurve: Starting forward: %s",argv[1]);
<   syslog(LOG_AUTHPRIV|LOG_ERR,argv[1]);
<   syslog(LOG_ERR, "Success/Failure?: ");
<   closelog();
<
```

# Bibliography

[1] Dan Kaminsky: An Astonishing Collaboration
http://www.doxpara.com/?p=1162,

[2] NaCl: Networking and Cryptography library
http://nacl.cr.yp.to

[3] quote: ”Despite its extremely high level of security, DNSCurve is very easy for software authors to implement, and very easy for administrators to deploy.”
http://dnscurve.org

[4] D. J. Bernstein’s home page
http://cr.yp.to/djb.html

[5] ON2IT Security
http://on2it.net

[6] DNSCurve: Comparison of DNSSEC and DNSCurve
http://dnscurve.org/dnssec.html

[7] DNSCurve: ”Computing Curve25519 shared secrets for ten million servers takes under ten minutes of computation on a Core 2 Quad”
http://dnscurve.org/crypto.html

[8] Domain names - implementation and specification
http://www.ietf.org/rfc/rfc1035.txt