Classical internet applications
Lab session: DNSSEC

Anatoly Tykushin Security and Network Engineering, a.tykushin@innopolis.ru

September 26, 2016

Contents

1 Setting up a validating resolver

1.1 What does a validating resolver do?

A DNSSEC validating resolver uses RRSIG(resourse record signature), DNSKEY(public key), DS(delegation signer), NSEC(next secure record) **records and public key (asymmetric) cryptography to prove the integrity of the DNS data**. A private key (specific to a zone) is used to encrypt a hash of a set of resource records — this is the digital signature stored in a RRSIG record. The corresponding public key is stored in the DNSKEY resource record. The validating resolver uses that DNSKEY to decrypt the RRSIG and then compares the result with the hash of the corresponding resource record set to verify it is not changed. A hash of the public DNSKEY is stored in a DS record. This is stored in the parent zone. **The validating resolver retrieves from the parent the DS record and its corresponding signature (RRSIG) and public key (DNSKEY); a hash of that public key is available from its parent**. This becomes a chain of trust — also called an authentication chain. The validating resolver is configured with a trust anchor — this is the starting point which refers to a signed zone. The trust anchor is a DNSKEY or DS record and should be securely retrieved from a trusted source (not using DNS).

1.2 Adding support for DNSSEC to Unbound

1.2.1 Changes to unbound configure file

Before enabling DNSSEC it is needed to update root.key file containing key of root server - begin of trust anchor.

In BIND server we should enter the ***dnssec-enable yes; dnssec-validation: yes***; statements in the options directive of your named.conf.

As an alternative to bind you could use UNBOUND as a DNSSEC aware recursive nameserver. UNBOUND does not need any special configuration options except for the configuration of a trust-anchor to perform DNSSEC validation.

To enable DNSSEC for Unbound we have to set autotrust option as shown below:

```
auto-trust-anchor-file: "/usr/local/etc/unbound/root.key"
```

After that used reconfigure command (unbound-control reload). How the check proceeded you can see in the next subsection.

1.2.2 Root key verifying

Using command

```
    dig com. SOA +dnssec 188.130.155.38:5355
```

we should see AD flag in the answer. Contents of the answer are shown below

```
    ; <<>> DiG 9.10.3-P4-Ubuntu <<>> com. SOA +dnssec 188.130.155.38:5355
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 24420
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 512
;; QUESTION SECTION:
;com. IN SOA

;; ANSWER SECTION:
com. 38 IN SOA a.gtld-servers.net. nstld.verisign-grs.com. 1473089869 1800 900 604800 86400
com. 38 IN RRSIG SOA 8 1 900 20160912153749 20160905142749 27452 com. dKR3chwoniI4eY4Rk5cOEai8rOV9ignNeKr

;; Query time: 43 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Mon Sep 05 18:52:28 MSK 2016
;; MSG SIZE  rcvd: 268

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 14480
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
```

```
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;188.130.155.38:5355. IN A

;; AUTHORITY SECTION:
. 1794 IN SOA a.root-servers.net. nstld.verisign-grs.com. 2016090500 1800 900 604800 86400

;; Query time: 38 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Mon Sep 05 18:52:28 MSK 2016
;; MSG SIZE  rcvd: 123
```

So as you can see in the log of the answer section flags contains:

- qr
- rd
- ra
- ad - DNS Authenticated Data (RFC 3655).

That means that validation was successfull

Results of the validity check os3.nl is shown in the log:

```
    tyvision@st5:~$ dig os3.nl. SOA +dnssec @188.130.155.38 -p 5355

; <<>> DiG 9.10.3-P4-Ubuntu <<>> os3.nl. SOA +dnssec @188.130.155.38 -p 5355
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 7756
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;os3.nl.                IN    SOA

;; ANSWER SECTION:
os3.nl.            21599    IN    SOA    ns1.os3.nl. hostmaster.os3.nl. 1474376338 3600 1800 21600 3600
os3.nl.            21599    IN    RRSIG    SOA 5 2 21600 20161020115858 20160920115858 42048 os3.nl. GmF4

;; Query time: 385 msec
;; SERVER: 188.130.155.38#5355(188.130.155.38)
;; WHEN: Sat Sep 24 15:2
```

Again you can see the **AD** flag in the flags section of answer.

2 Where does BIND/Unbound store the DNSSEC root key?

You can specify it by yourself in file unbound.conf (as it was shown in section 1.2.1 - for my configuration /usr/local/etc/unbound/root.key

3 How do "managed keys" differ from "trusted keys"? Which RFC describes the mechanisms for managedkeys?

Trusted-keys are used for configuring trust anchors for DNSSEC validation. You can configure them so that all the data served by the authoritative servers for "foo.bar" is validated before it is handed to the protected infrastructure that have the recursive servers configured as their forwarder.

Managed-key allows automatic tracking of the key using a protocol known as RFC-5011 (for BIND 9.7 and later versions).

4 (6)How did you modify the DNSSEC root key? (7)What problems did your server encounter, and how did it react?

In case you modify root key (/usr/local/etc/unbound/root.key) then any queries that are made will meet SERVFAIL error.

5 (9)Lookup which cryptographic algorithms are available for use in DNSSEC. Which one do you prefer, and why? (10) In practice, different algorithms, key sizes and key lifetimes are chosen for KSKs and ZSKs. Discuss what are these differences in

1. algorithm
2. key length
3. key rollover period

List of supported algorythms for key generation is shown below:

```
root@st5:/etc/nsd# ldns-keygen -a list
Possible algorithms:
RSAMD5
RSASHA1
RSASHA1-NSEC3-SHA1
RSASHA256
RSASHA512
ECC-GOST
ECDSAP256SHA256
ECDSAP384SHA384
DSA
DSA-NSEC3-SHA1
hmac-md5.sig-alg.reg.int
hmac-sha1
hmac-sha256
```

To answer properly to this question let's look at RFC 4641 (section 3.4-3.5):

Key Algorithm

There are currently three different types of algorithms that can be used in DNSSEC: RSA, DSA, and elliptic curve cryptography. The latter is fairly new and has yet to be standardized for usage in DNSSEC.

RSA has been developed in an open and transparent manner. As the patent on RSA expired in 2000, its use is now also free.

DSA has been developed by the National Institute of Standards and Technology (NIST). The creation of signatures takes roughly the same time as with RSA, but is 10 to 40 times as slow for verification.

RFC's developers suggest the use of RSA/SHA-1 as the preferred algorithm for the key. The current known attacks on RSA can be defeated by making your key longer. As the MD5 hashing algorithm is showing cracks, we recommend the usage of SHA-1.

At the time of publication, it is known that the SHA-1 hash has cryptanalysis issues. There is work in progress on addressing these issues. The usage of public key algorithms based on hashes stronger than SHA-1 (e.g., SHA-256) is recommended, as soon as these algorithms are available in protocol specifications and implementations.

As far as I know today, ECC is one of the secured algorithms (384 bits key length ECC is equal to 4096 bit RSA). So I'd prefer to use it.

Key Sizes and Key Lifetimes

When choosing key sizes, zone administrators will need to take into account how long a key will be used, how much data will be signed during the key publication period and, optionally, how large the key size of the parent is. As the chain of trust really is "a chain", there is not much sense in making one of the keys in the chain several times larger then the others. As always, it's the weakest link that defines the strength of the entire chain.

Generating a key of the correct size is a difficult problem; RFC 3766 tries to deal with that problem. The first part of the selection procedure in Section 1 of the RFC states:

1. Determine the attack resistance necessary to satisfy the security requirements of the application. Do this by estimating the minimum number of computer operations that the attacker will be forced to do in order to compromise the security of the system and then take the logarithm base two of that number.

This culminated in the table given below (slightly modified for our purpose):

System requirement for attack resistance (bits)	Symmetric key size (bits)	RSA or DSA modulus size (bits)
70	70	947
80	80	1228
90	90	1553
100	100	1926
150	150	4575
200	200	8719
250	250	14596

The key sizes given are rather large. This is because these keys are resilient against a trillionaire attacker. Assuming this rich attacker will not attack your key and that the **key is rolled over once a year**, we come to the following recommendations about KSK sizes: 1024 bits for low-value domains, 1300 bits for medium-value domains, and 2048 bits for high-value domains.

Whether a domain is of low, medium, or high value depends solely on the views of the zone owner. One could, for instance, view leaf nodes in the DNS as of low value, and top-level domains (TLDs) or the root zone of high value. **The suggested key sizes should be safe for the next 5 years.**

As ZSKs can be rolled over more easily (and thus more often), the key sizes can be made smaller. But as said in the introduction of this paragraph, making the ZSKs' key sizes too small (in relation to the KSKs' sizes) doesn't make much sense. Try to limit the difference in size to about 100 bits.

Conclusion: I've choosed ECC (elliptic curve cryptography) 256 and 257 bits size (they've been choosed by default). Once a year I'll change ZSK and once in 5 years I'll change KSK.

6 Show the signed version of your zone file. How does it differ from the unsigned version? Any unexpected differences?

```
      st5.os3.su.    1800    IN    SOA    st5.os3.su. mail.st5.os3.su. 2014070312 3600 900 1209600 1800
st5.os3.su.    1800    IN    RRSIG    SOA 12 3 1800 20161022195553
20160924195553 51752 st5.os3.su.
8O97BIw2UZnWPIK7gmCM4u/736O67sJtUqgYqOwFdXJXo2x7F2wtrvZ0deT76Ty+lxCzZ
u0kb9sF2O7gZV6BOQ==
st5.os3.su.    1800    IN    A    188.130.155.38
st5.os3.su.    1800    IN    RRSIG    A 12 3 1800 20161022195553
20160924195553 51752 st5.os3.su.
q8CAkXnx+3l2ZGWFX+FW0iPJlk7Ecrbz7yHZ0VXiOXWGD6oMBjiimc/hO90fFhJnmZLWr
nLZlZlT8ydeeKH7kg==
st5.os3.su.    1800    IN    NS    st5.os3.su.
st5.os3.su.    1800    IN    NS    st6.os3.su.
st5.os3.su.    1800    IN    RRSIG    NS 12 3 1800 20161022195553
20160924195553 51752 st5.os3.su.
Z4ToHbI4LEX9Om9WQ3cdEwKHZhEiWdHQMY3rLFmG6AwGIlqht0fylRFT0zsr/CBhr6nw
9SqFNGNgdv0LwjpYqw==
st5.os3.su.    1800    IN    MX    10 mail.st5.os3.su.
st5.os3.su.    1800    IN    MX    20 mail.st2.os3.su.
st5.os3.su.    1800    IN    MX    30 mail.st3.os3.su.
st5.os3.su.    1800    IN    RRSIG    MX 12 3 1800 20161022195553
20160924195553 51752 st5.os3.su.
VyLdW6LNSST2APKT3d675B+aZLqmYuPN9/2cKjVNGAIPEVUGNJnVWKoec9AJkGPCRTV
gxCUjdVYuTZ54roEr7A==
st5.os3.su.    1800    IN    DNSKEY    256 3 12
0S7zg6fl8auWpYJj1iAYhfSxZgzt3Og4LvnEcmq6vYCuf52It8VI85Tz2FojqoEkYnd
2OKckccT6u5vvQ8boxA== ;{id = 51752 (zsk), size = 512b}
st5.os3.su.    1800    IN    DNSKEY    257 3 12
vdUOxM3GSs5CClyUtWc8qAuT8TFQvo8Vzn4AQOQCsyC6NFHSrDcVXZPzCyiaaIATjurA
ZlYnR9OSw4hWHIS9bA== ;{id = 9712 (ksk), size = 512b}
```

```
st5.os3.su.    1800    IN    RRSIG    DNSKEY 12 3 1800 20161022195553
20160924195553 9712 st5.os3.su.
qo1C+Bv1wYb02PDMPQ+HOhfZOu82wbwF9AehVX/AFsr74d7PnjOSLE9oP9QCkejTJ8YK
pr3g/E+fvHvw7sx+cQ==
st5.os3.su.    3600    IN    NSEC3PARAM    1 0 1 46dd71e0b4ae4aba
st5.os3.su.    3600    IN    RRSIG    NSEC3PARAM 12 3 3600 20161022195553
20160924195553 51752 st5.os3.su.
hXqJt7IwTzAJJtRlNb2/und2VT9SA6Ofy5f3Ey1uUdJ6iJsZ4KbRUMcuiiqwAuWULvgx
wztYckeRcIkK90lYbQ==
kv55ifjkshf5uc1ar4qmt1cjjm5qfm81.st5.os3.su.    1800    IN    NSEC3    1 1 1
46dd71e0b4ae4aba  lsv4vcdtnh6ke5hpqnig9l50mifl9gv9 A NS SOA MX RRSIG DNSKEY
NSEC3PARAM
kv55ifjkshf5uc1ar4qmt1cjjm5qfm81.st5.os3.su.    1800    IN    RRSIG    NSEC3
12 4 1800 20161022195553 20160924195553 51752 st5.os3.su.
zPJP3s5gUF+XbctrKZbB2PWtU7gvP5WJLzS66txI1i6Is3dY971fGSCzWIkXd1M8RMwU
sDVYNMWme6QuGraGMw==
1dqfu31pae5arlqebp64917r0hef2ru7.st5.os3.su.    1800    IN    NSEC3    1 1 1
46dd71e0b4ae4aba  49432a02tsq982umfk6e7fjup5d31hlm
1dqfu31pae5arlqebp64917r0hef2ru7.st5.os3.su.    1800    IN    RRSIG    NSEC3
12 4 1800 20161022195553 20160924195553 51752 st5.os3.su.
0w6CSh7L1cCV9XeNUo6DvXmMSGZ5U+bZ8LwLLYWWtn3aNfACIyqPD6qC/JSFGQ2HZ1hMn
6eCcMa4iAi5ghcjsw==

st6.st5.os3.su. IN DS 1528 12 1 60B91FA482D5776F6939A5513D21FBC7854AF7EC
st6.st5.os3.su. IN DS 1528 12 2 19FA386ADE337DD9CE56A6F76AF487C8627C9415A90A2C8D504F74CA 5B483745


mail._domainkey.st5.os3.su.    1800    IN    TXT    "v=DKIM1; k=rsa; "
"p=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDEwMICCqjZ6ZIHOUDDKl6ZntzAY24
rV0NhitPMcKJSF453XHwzJOHQCC8ywk+uN3zIROF4p/Rvk1D5q0eA5HIgRPjY8d0CbwqDl
U8E8q4SfYoawNwL2jZYEZ2aVLOUwcFleTlj1lUnmv8SCIh6a856AVrMEh7tCFkZVjRk284
DewIDAQAB"
mail._domainkey.st5.os3.su.    1800    IN    RRSIG    TXT 12 5 1800
20161022195553 20160924195553 51752 st5.os3.su.
Q7DNJUhfBCeCw2DvMkiueFvHYRc0uIt7HOMfFXM/+W/zA7LUtDptJG8TKOCe9t41roHakb
H9uz9rzq82D0j48A==
e5bu41vpba2459t0fs85mcg560bdd0mq.st5.os3.su.    1800    IN    NSEC3
1 1 1 46dd71e0b4ae4aba  kbue47jr6gqg03n3itjd5sqj11cfn0r9 TXT RRSIG
e5bu41vpba2459t0fs85mcg560bdd0mq.st5.os3.su.    1800    IN    RRSIG
NSEC3 12 4 1800 20161022195553 20160924195553 51752 st5.os3.su.
CmdFpjNMPYhL51UnbD7ltlA3Ya9XlVBoTho2OBQwfGzJ9HQ0ZvyBacX62J7o+0UWrX1d
bWAW9scsC9H/V1hsbQ==
godmail.st5.os3.su.    1800    IN    CNAME    thegod.st5.os3.su.
godmail.st5.os3.su.    1800    IN    RRSIG    CNAME 12 4 1800 201610
22195553 20160924195553 51752 st5.os3.su.
FjVXZp/Ev3PTE06DIBkPVfCT57lr2qpHCu30Edhrp2K4g2qGf/bp0vkb/2KeE1PaI6EUI
HpSAsV9z/BCxhNweA==
49432a02tsq982umfk6e7fjup5d31hlm.st5.os3.su.    1800    IN    NSEC3
1 1 1 46dd71e0b4ae4aba  5r39o8da0269pq1p8nd435p6segn8nsd CNAME RRSIG
49432a02tsq982umfk6e7fjup5d31hlm.st5.os3.su.    1800    IN    RRSIG
NSEC3 12 4 1800 20161022195553 20160924195553 51752 st5.os3.su.
zG5qAWT5QNyvxZCSCQZfGjpA7f+RbxMrgdYWYk0hP1RMXsCh5mO0eveIX2n2ZhGsZ6AG
s7z6ufKD1s6RN7lDUA==
mail.st5.os3.su.    1800    IN    A    188.130.155.38
mail.st5.os3.su.    1800    IN    RRSIG    A 12 4 1800 20161022195553
20160924195553 51752 st5.os3.su.
QAhgsZCUqyXxNWm8iRCPVkK0p1X0NW8gaLGh2MKoF5sSdzS2TIP1wriRXaDvzeSffcf0W
HKqScWl5k+emTihPg==
m6mdh3e73su9hjb2gnju8vf9qs94ogtu.st5.os3.su.    1800    IN    NSEC3
1 1 1 46dd71e0b4ae4aba  1dqfu31pae5arlqebp64917r0hef2ru7 A RRSIG
m6mdh3e73su9hjb2gnju8vf9qs94ogtu.st5.os3.su.    1800    IN    RRSIG
NSEC3 12 4 1800 20161022195553 20160924195553 51752 st5.os3.su.
```

```
WGT54VzNFbXP8+t0eUct13MwQoX1ZoXz7CdcH41C5MtY3Y+79iExwASQ5ZLPUa8zx9x
+7p37HwiLhB9rGJVffg==
numbaone.st5.os3.su.    1800    IN    CNAME    sne1.st5.os3.su.
numbaone.st5.os3.su.    1800    IN    RRSIG    CNAME 12 4 1800 201610
22195553 20160924195553 51752 st5.os3.su.
tcVn22aMSrjAMffEr8ScwL1ByHXplrH7LEwH75uYIZH1knF9oQSf0BX8I8oqPt0A/o+
9SH99PFowFy782hJ21A==
lsv4vcdtnh6ke5hpqnig9l50mifl9gv9.st5.os3.su.    1800    IN
NSEC3    1 1 1 46dd71e0b4ae4aba  m6mdh3e73su9hjb2gnju8vf9qs94ogtu CNAME RRSIG
lsv4vcdtnh6ke5hpqnig9l50mifl9gv9.st5.os3.su.    1800    IN    RRSIG
NSEC3 12 4 1800 20161022195553 20160924195553 51752 st5.os3.su.
lIlSEAjR8bjFJx2Mz/hAtbLWOhKiZOYZmST6zwgVp/oRF4vgAHO5HPzfeSOHWfWv8lJOlM
Uy8FT3f/4tVg2/fw==
resmail.st5.os3.su.    1800    IN    MX    15 mail.st5.os3.su.
resmail.st5.os3.su.    1800    IN    RRSIG    MX 12 4 1800 20161022195553
20160924195553 51752 st5.os3.su.
lXHLQLF1JE2gg2a1/5HaGhZobYbI7/mPmVDVnaA4ayGEU1EkrNtvXd1/KidVK6yE/
RMvxhcZSS9jz7m+3EwUSQ==
5r39o8da0269pq1p8nd435p6segn8nsd.st5.os3.su.    1800    IN    NSEC3
1 1 1 46dd71e0b4ae4aba  95s6a6cmoaqc90jac6il0l6vcnsn0du9 MX RRSIG
5r39o8da0269pq1p8nd435p6segn8nsd.st5.os3.su.    1800    IN    RRSIG    NSEC3
12 4 1800 20161022195553 20160924195553 51752 st5.os3.su.
f8JQqE3R0FTgzizjHZo68EmD0iDyg2oHxJDvfrWtE2Dreny2Zy/N7lRZUhu6c7Xu7Q3z
daB5V57ZxAxSQHw7kw==
st6.st5.os3.su.    1800    IN    A    188.130.155.39
st6.st5.os3.su.    1800    IN    NS    st6.os3.su.
st6.st5.os3.su.        IN DS 1528 12 1
60B91FA482D5776F6939A5513D21FBC7854AF7EC
st6.st5.os3.su.        IN DS 1528 12 2
19FA386ADE337DD9CE56A6F76AF487C8627C9415A90A2C8D504F74CA 5B483745
kbue47jr6gqg03n3itjd5sqj11cfn0r9.st5.os3.su.    1800    IN    NSEC3    1 1 1
46dd71e0b4ae4aba  kv55ifjkshf5uc1ar4qmt1cjjm5qfm81 NS
kbue47jr6gqg03n3itjd5sqj11cfn0r9.st5.os3.su.    1800    IN    RRSIG    NSEC3
12 4 1800 20161022195553 20160924195553 51752 st5.os3.su.
YOVNAdobVyEnMdHClojNjJ+JZHKBcvRDvzCR8O8EZjL7unRymAE11RiJfoexxvhDpXQyNCwXV3DHwf
qbSWR7iw==
www.st5.os3.su.    1800    IN    A    188.130.155.38
www.st5.os3.su.    1800    IN    RRSIG    A 12 4 1800 20161022195553
20160924195553 51752 st5.os3.su.
D4jfVL2zPNFmKouuYKUczuouMdktbGgk55llY7e5ZnKawJFN1l/Hjao4HLfJ2U+KM8GGJJV7cigJ2D
w8S/oyHg==
95s6a6cmoaqc90jac6il0l6vcnsn0du9.st5.os3.su.    1800    IN    NSEC3    1 1 1
46dd71e0b4ae4aba  e5bu41vpba2459t0fs85mcg560bdd0mq A RRSIG
95s6a6cmoaqc90jac6il0l6vcnsn0du9.st5.os3.su.    1800    IN    RRSIG    NSEC3
12 4 1800 20161022195553 20160924195553 51752 st5.os3.su.
sGMaPdW8VcaxIwX6xsU/Rob6JZg3cGK2fJ3/5+IqCB3+21MKFaaLjd4tGg1nT2iAnb0onLbvjc263h
bExjAyyw==
```

Firstly, we have, for example, some A records in an openformat and then we have them in a crypted format in a RRSET. This was made for backwards compatibility with DNS. So for DNSSEC we have RRSIGs for RRSETs. Also we have NSEC3 records according to their purpose (point to the neighbour). We have DNSKEYs for the zone. Also I've added 2 DS records from st6.os3.su because I've delegated zone st5.st6.os3.su to his master nameserver.

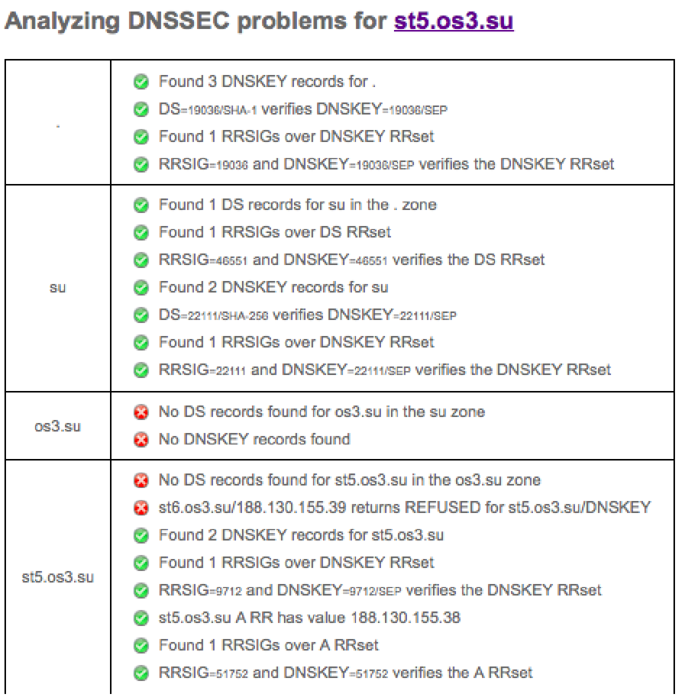
7 (13)Which DS record do you need to send to your neighbors, and why that one?

I have to send him DS record of DNSKEY for zone st5.st6.os3.su, which he has delegated to me. It will be used to verify all my answers for his requests. In next section we will test our configuration.

8 (14) Show the results of the examination of your secured domain

You can see result of verification on the figures 1 and 2. There are a few errors occured:

- **no DS records found for the os3.su in .su zone, no DNSKEY** errors in zone os3.su are okay, because it isn't signed using DNSSEC
- **no DS record for st5.os3.su in the os3.su** that's okay too, but it can be fixed by providing generated DS records
- **last 2 errors** commited because of some misconfiguration on st6.os3.su



Analyzing DNSSEC problems for st5.os3.su

.	✓ Found 3 DNSKEY records for . ✓ DS=19036/SHA-1 verifies DNSKEY=19036/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=19036 and DNSKEY=19036/SEP verifies the DNSKEY RRset
su	✓ Found 1 DS records for su in the . zone ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG=46551 and DNSKEY=46551 verifies the DS RRset ✓ Found 2 DNSKEY records for su ✓ DS=22111/SHA-256 verifies DNSKEY=22111/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=22111 and DNSKEY=22111/SEP verifies the DNSKEY RRset
os3.su	✗ No DS records found for os3.su in the su zone ✗ No DNSKEY records found
st5.os3.su	✗ No DS records found for st5.os3.su in the os3.su zone ✗ st6.os3.su/188.130.155.39 returns REFUSED for st5.os3.su/DNSKEY ✓ Found 2 DNSKEY records for st5.os3.su ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=9712 and DNSKEY=9712/SEP verifies the DNSKEY RRset ✓ st5.os3.su A RR has value 188.130.155.38 ✓ Found 1 RRSIGs over A RRset ✓ RRSIG=51752 and DNSKEY=51752 verifies the A RRset

Figure 1: Vefirying st5.os3.su zone

9 (15) Describe which DS and DNSKEY records are important for your domain. Which keys are used where?

DNSKEY and DS for st5.os3.su zone. 2 DNSKEYs are stored in st5.os3.su.zone file. DS key must be relied at os3.su zone. DS is a hash and it is used to verify DNSKEY of child zone.

10 (16) Start planning for a Zone Signing Keyrollover

a) Describe the options for doing a ZSK rollover, make a motivated choice for one procedure. According to RFC 4641 we there are 2 major ways to perform a ZSK rollover:

- Pre-Publish Key Rollover
- Double Signature Zone Signing Key Rollover

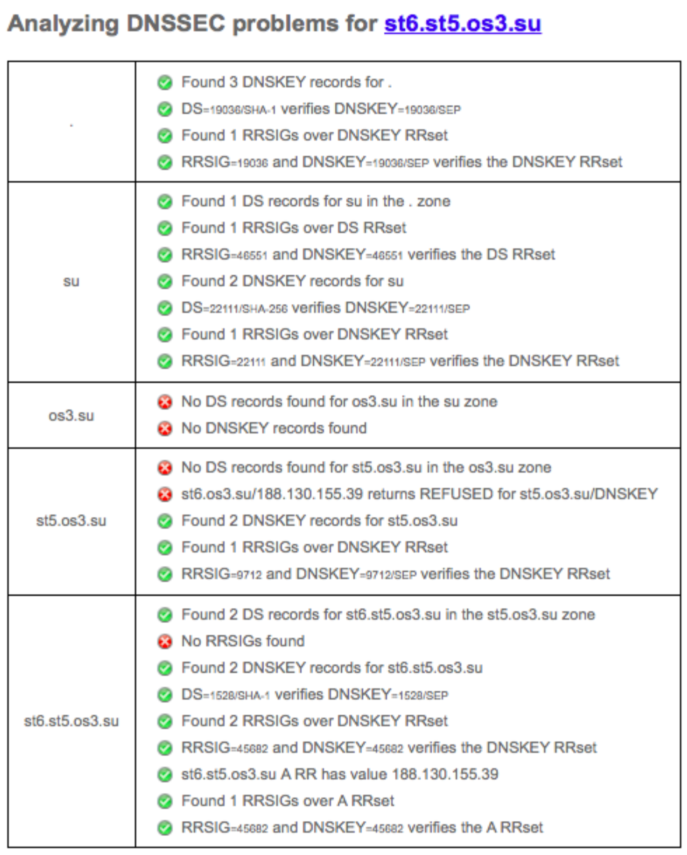
To make a motivated choise let's look up for pros and cons of every algorithm.

Pre-publish key rollover: This rollover does not involve signing the zone data twice. Instead, before the actual rollover, the new key is published in the key set and thus is available for cryptanalysis attacks. A small disadvantage is that this process requires four steps.

Double signature ZSK rollover: The drawback of this signing scheme is that during the rollover the number of signatures in your zone doubles; this may be prohibitive if you have very big zones. An advantage is that it only requires three steps.

As we have a small zones for our lab session - the first path was chosen.

(b) How do you implement this procedure with the tools for signing your zone?


Analyzing DNSSEC problems for st6.st5.os3.su

.	✓	Found 3 DNSKEY records for .
	✓	DS$_{=19036/SHA-1}$ verifies DNSKEY$_{=19036/SEP}$
	✓	Found 1 RRSIGs over DNSKEY RRset
	✓	RRSIG$_{=19036}$ and DNSKEY$_{=19036/SEP}$ verifies the DNSKEY RRset
su	✓	Found 1 DS records for su in the . zone
	✓	Found 1 RRSIGs over DS RRset
	✓	RRSIG$_{=46551}$ and DNSKEY$_{=46551}$ verifies the DS RRset
	✓	Found 2 DNSKEY records for su
	✓	DS$_{=22111/SHA-256}$ verifies DNSKEY$_{=22111/SEP}$
	✓	Found 1 RRSIGs over DNSKEY RRset
	✓	RRSIG$_{=22111}$ and DNSKEY$_{=22111/SEP}$ verifies the DNSKEY RRset
os3.su	✗	No DS records found for os3.su in the su zone
	✗	No DNSKEY records found
st5.os3.su	✗	No DS records found for st5.os3.su in the os3.su zone
	✗	st6.os3.su/188.130.155.39 returns REFUSED for st5.os3.su/DNSKEY
	✓	Found 2 DNSKEY records for st5.os3.su
	✓	Found 1 RRSIGs over DNSKEY RRset
	✓	RRSIG$_{=9712}$ and DNSKEY$_{=9712/SEP}$ verifies the DNSKEY RRset
st6.st5.os3.su	✓	Found 2 DS records for st6.st5.os3.su in the st5.os3.su zone
	✗	No RRSIGs found
	✓	Found 2 DNSKEY records for st6.st5.os3.su
	✓	DS$_{=1528/SHA-1}$ verifies DNSKEY$_{=1528/SEP}$
	✓	Found 2 RRSIGs over DNSKEY RRset
	✓	RRSIG$_{=45682}$ and DNSKEY$_{=45682}$ verifies the DNSKEY RRset
	✓	st6.st5.os3.su A RR has value 188.130.155.39
	✓	Found 1 RRSIGs over A RRset
	✓	RRSIG$_{=45682}$ and DNSKEY$_{=45682}$ verifies the A RRset

Figure 2: Verifying st6.st5.os3.su zone

The tools are the same. We'll use keygen tool twice: for generating active and passive keys.

(c) Which timers are important for this procedure? The most important timer is TTL field. All timings are shown in next question

(d) Implement the procedure and use a DNSSEC debugger to verify each step. Don't forget to show the results of each verification.

Order of steps:

1. Generate new ZSK
2. Write new key in the zone file (unsigned zone file)
3. Resign zone
4. Publish the new zone in NSD and wait for all caching servers to update. This is the time of the highest TTL in your zone
5. Roll the keys. Edit zone and raise serial. This time we'll use the new ZSK for signing, and include the old one as a passive key. To do this, we need to add the content of the old key to the zonefile as well.
6. When you have waited long enough (highest is TTL), you can remove the old key from the zone. Edit the zonefile, raise the serial and remove both .key contents from the file. From this point on, the old key files are no longer needed. You can remove them, but better move them to a separate directory for backup purposes.
7. Re-sign the zone with the new key, and publish the zone.

To perform a ZSK rollover I wrote a bash script (shown in appendix A.1)

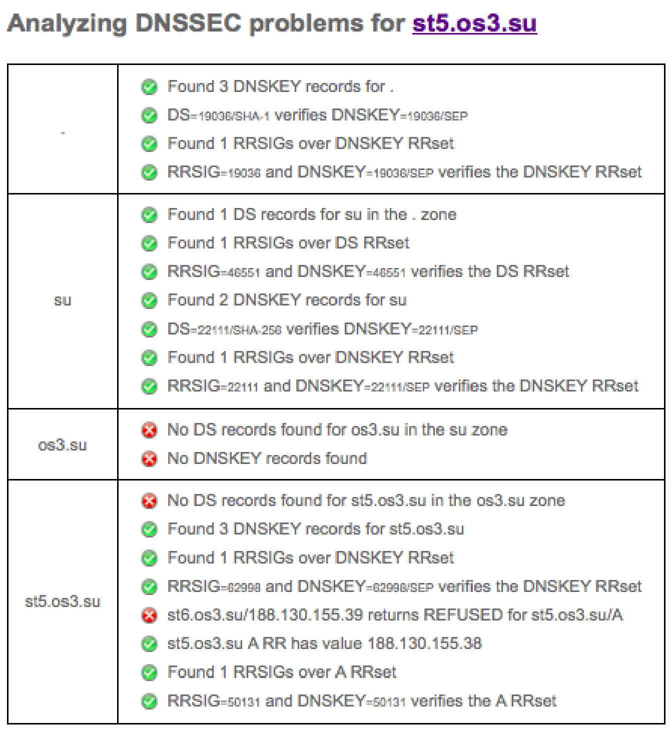
One thing which is now under testing stage is old keys backup procedure. It may be finished in a couple of weeks.

During procedure I've checked what DNSSEC analyze tool was shown: (figures 3 and 4

As you can see that after the zsk rollover RRSIGs are changed

11 (17)Can you use the same procedure for a KSK rollover? What does it depend on?

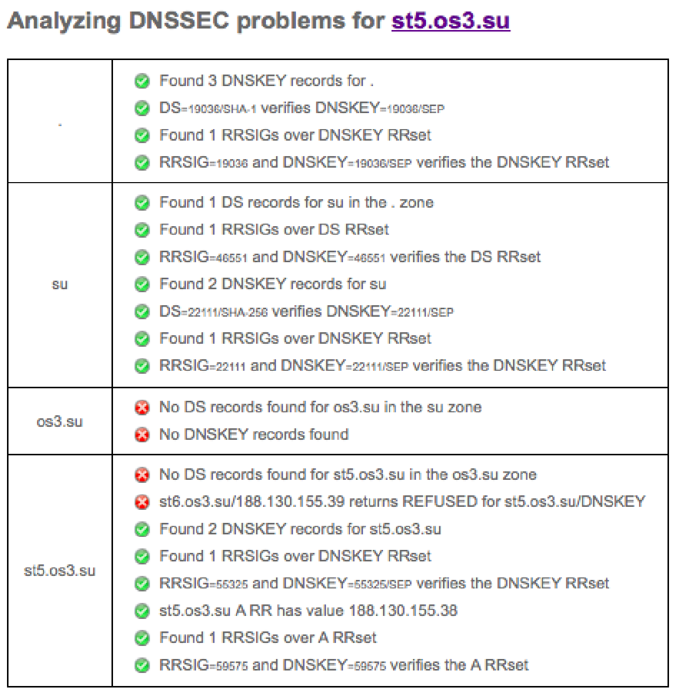
It's much more difficult to automate KSK rollover, because change of DS records in the parent zone required (some kind of interaction).



Analyzing DNSSEC problems for st5.os3.su

.	✓ Found 3 DNSKEY records for . ✓ DS=19036/SHA-1 verifies DNSKEY=19036/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=19036 and DNSKEY=19036/SEP verifies the DNSKEY RRset
su	✓ Found 1 DS records for su in the . zone ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG=46551 and DNSKEY=46551 verifies the DS RRset ✓ Found 2 DNSKEY records for su ✓ DS=22111/SHA-256 verifies DNSKEY=22111/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=22111 and DNSKEY=22111/SEP verifies the DNSKEY RRset
os3.su	✗ No DS records found for os3.su in the su zone ✗ No DNSKEY records found
st5.os3.su	✗ No DS records found for st5.os3.su in the os3.su zone ✓ Found 3 DNSKEY records for st5.os3.su ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=62998 and DNSKEY=62998/SEP verifies the DNSKEY RRset ✗ st6.os3.su/188.130.155.39 returns REFUSED for st5.os3.su/A ✓ st5.os3.su A RR has value 188.130.155.38 ✓ Found 1 RRSIGs over A RRset ✓ RRSIG=50131 and DNSKEY=50131 verifies the A RRset

Figure 3: Vefirying st5.os3.su zone during key change (3 DNSKEY records)



Analyzing DNSSEC problems for st5.os3.su

.	✓ Found 3 DNSKEY records for . ✓ DS=19036/SHA-1 verifies DNSKEY=19036/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=19036 and DNSKEY=19036/SEP verifies the DNSKEY RRset
su	✓ Found 1 DS records for su in the . zone ✓ Found 1 RRSIGs over DS RRset ✓ RRSIG=46551 and DNSKEY=46551 verifies the DS RRset ✓ Found 2 DNSKEY records for su ✓ DS=22111/SHA-256 verifies DNSKEY=22111/SEP ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=22111 and DNSKEY=22111/SEP verifies the DNSKEY RRset
os3.su	✗ No DS records found for os3.su in the su zone ✗ No DNSKEY records found
st5.os3.su	✗ No DS records found for st5.os3.su in the os3.su zone ✗ st6.os3.su/188.130.155.39 returns REFUSED for st5.os3.su/DNSKEY ✓ Found 2 DNSKEY records for st5.os3.su ✓ Found 1 RRSIGs over DNSKEY RRset ✓ RRSIG=55325 and DNSKEY=55325/SEP verifies the DNSKEY RRset ✓ st5.os3.su A RR has value 188.130.155.38 ✓ Found 1 RRSIGs over A RRset ✓ RRSIG=59575 and DNSKEY=59575 verifies the A RRset

Figure 4: Verifying st5.os3.su zone after roll over (again 2 DNSKEY records)

For the rollover of a KSK, the same consideration as for the rollover of a ZSK. KSK rollover can be done both by double-signature or pre-publish(it has some drawbacks) key rollovers.

Appendices

A ZSK Rollover

A.1 Script

```
#!/bin/bash
cd /etc/nsd/zones

echo "Enter zone name you want to sign and rollover"
read zone

echo "Doing stuff for $zone"
echo "Generate key ZSK1"
export ZSK=‘ldns-keygen -a ECC-GOST "$zone"‘

if [ $? -eq 0 ]; then
    echo "OK. ZSK is $ZSK"
else
    echo FAIL
    exit 1
fi

echo "Generate key ZSK2"
export ZSK2=‘ldns-keygen -a ECC-GOST "$zone"‘
if [ $? -eq 0 ]; then
    echo "OK. ZSK2 is $ZSK2"
else
    echo FAIL
    exit 1
fi


echo "Generate KSK"
export KSK=‘ldns-keygen -k -a ECC-GOST "$zone"‘
if [ $? -eq 0 ]; then
    echo "OK. KSK is $KSK"
else
    echo FAIL
    exit 1
fi


echo ""
echo "Now add ZSK2 to the zonefile"
cat $ZSK2.key >> $zone.zone

echo "Signing zone with ZSK and KSK"
sudo ldns-signzone -n -p -s $(head -n 500 /dev/urandom | sha1sum | cut -b 1-16) $zone.zone $ZSK $KSK

echo "Now check nsd.conf file: zone filenames and other options"
echo "Type any letter when you’re ready"
read $a

sudo nano ../nsd.conf


#sudo ldns-signzone -n -p -s $(head -n 500 /dev/urandom | sha1sum | cut -b 1-16) st5.st6.os3.su.zone $ZSK

echo "Restarting NSD"
sudo service nsd restart
```

```
echo "After restart of the zone you have to edit zone file: change serial number, delete new dnskey, add
echo "New DNSKEY is $ZSK2. Check it in you zonefile"
echo "CHANGE YOUR SERIAL"
echo "Type any letter when you're ready"

read $a
cat $zone.zone | grep -i -v "DNSKEY" > $zone.zone.tmp
rm $zone.zone && mv $zone.zone.tmp $zone.zone

cat $ZSK.key >> $zone.zone
nano $zone.zone

echo ""
echo "Resigning zone"

sudo ldns-signzone -n -p -s $(head -n 500 /dev/urandom | sha1sum | cut -b 1-16) $zone.zone $ZSK2 $KSK
sudo service nsd restart

echo "Zone is resigned with a new key. But old key is still in there"
echo "Restart NSD"
echo "..."
echo ""
echo "Type any letter when you're ready for the next step"
read $a

echo "Now we've been waiting a lot, deleting old DNS key"
cat $zone.zone | grep -i -v "DNSKEY" > $zone.zone.tmp
rm $zone.zone && mv $zone.zone.tmp $zone.zone

echo "Resign zone"
sudo ldns-signzone -n -p -s $(head -n 500 /dev/urandom | sha1sum | cut -b 1-16) $zone.zone $ZSK2 $KSK
sudo service nsd restart

echo "Old key's been deleted. Zone is now signed with a new key. NSD server is restarted"
```

A.2 Output

```
/etc/nsd/zones
Enter zone name you want to sign and rollover
st5.os3.su
Doing stuff for st5.os3.su
Generate key ZSK1
OK. ZSK is Kst5.os3.su.+012+50125
Generate key ZSK2
OK. ZSK2 is Kst5.os3.su.+012+31318
Generate KSK
OK. KSK is Kst5.os3.su.+012+29238

Now add ZSK2 to the zonefile
Signing zone with ZSK and KSK
[sudo] password for tyvision:
Zone not read, error: Syntax error, could not parse the RR's class at st5.os3.su.zone line 36
Now check nsd.conf file: zone filenames and other options
Type any letter when you're ready

Restarting NSD
After restart of the zone you have to edit zone file: change serial number, delete new dnskey, add old dn
New DNSKEY is Kst5.os3.su.+012+31318. Check it in you zonefile
CHANGE YOUR SERIAL
```

```
Type any letter when you're ready

Resigning zone
Zone is resigned with a new key. But old key is still in there
Restart NSD
...

Type any letter when you're ready for the next step

Now we've been waiting a lot, deleting old DNS key
Resign zone
Old key's been deleted. Zone is now signed with a new key. NSD server is restarted
```