DNSSEC – elv és konfiguráció

Pásztor Miklós, ISZT

[2015. március]

Tartalomjegyzék

Ennek az írásnak a html változata elérhető a `http://deneb.iszt.hu/~pasztor/dnssec-elv-konfig.html` címen.

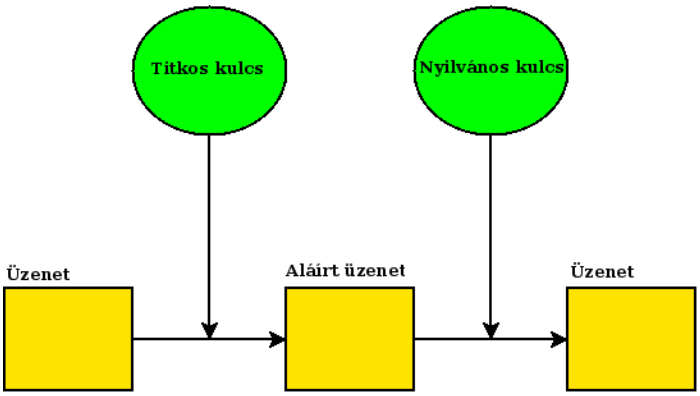
1. Alapismeretek

A DNSSEC két eleme maga az internet DNS, és a nyilvános kulcsú titkosítás elve. Megértéséhez ennek a két dolognak az ismerete szükséges. A következőkben rövid emlékeztető olvasható. Akiknek ez új, azok jobban teszik, ha először mást olvasnak [1], [2] .

1.1. Digitális aláírás

Nyilvános kulcsú titkosításnál és digitális aláírásnál **kulcspárokat** használunk. Egy ilyen kulcspár egy titkos és egy nyilvános részből áll. A nyilvános részt minél szélesebb körben hozzáférhetővé kell tenni. Például a weben a https (TLS) protokollnál használt X.509 tanúsítványok ilyen nyilvános kulcsokat tartalmaznak. A nyilvános kulcsokhoz tartozó titkos kulcspárt nagy gondossággal, bizalmasan kell kezelni, hiszen aki ezt birtokolja, az olvashatja a kulcs tulajdonosának küldött titkosított tartalmakat, illetve digitálisan aláírhat ezzel a kulccsal.

A digitális aláírást és az aláírás ellenőrzését szemlélteti az 1. ábra:



1. ábra. A digitális aláírás elve

1.2. Névfeloldás

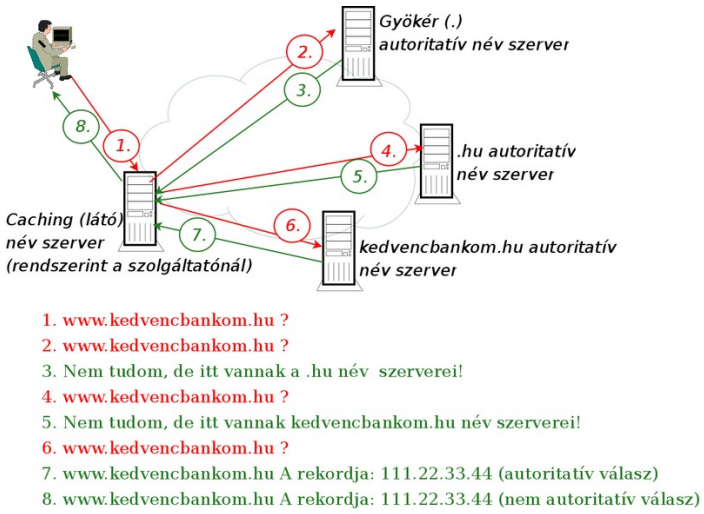
Az internet DNS legfőbb, eredeti célja, hogy nevekhez IP címeket rendelhessünk. A név → IP cím hozzárendelés mellett azonban sok más információhoz is hozzájuthatunk a DNS rendszer használata által.

A névfeloldás folyamata három típusú komponens együttműködésével valósul meg. Ezek:

1. Az interneten használt valamennyi eszközben megtalálható *stub rezolver*;
2. Rendszerint a felhasználóhoz hálózati értelemben közel levő *rekurzív névszerver*, (más néven caching vagy látó névszerver);
3. A DNS osztott hierarchiájának megfelelő *autoritatív névszerver*, (más néven hiteles, vagy mutató névszerver).

A névfeloldás folyamatát szemlélteti a 2. ábra:

[1] DNS-ről: `http://deneb.iszt.hu/~pasztor/dnslap.html`.
[2] A nyilvános kulcsú titkosítás elvéről: `http://deneb.iszt.hu/~pasztor/alairbev.html`.



2. ábra. A névfeloldás folyamata

1.3. DNS üzenetek szerkezete

A DNS protokoll többnyire UDP felett az 53-as porton működik, az esetek egy részében TCP felett szintén az 53-as porton. Érdemes felidézni, hogy milyen is az üzenetek szerkezete.

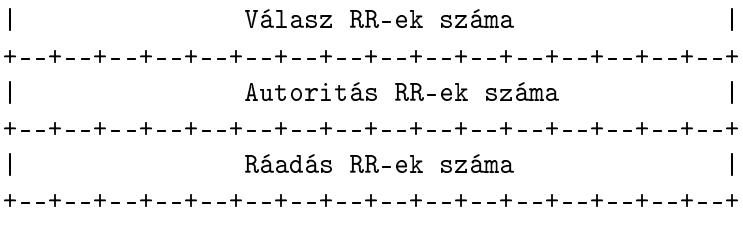
A DNS üzenetek szerkezetét az 1987-ből származó RFC1035 írja le. A kérdés és válasz üzenetek szerkezete egyforma:

```
+---------------------+
|       Fejrész       |
+---------------------+
|       Kérdés        | the question for the name server
+---------------------+
|   Válasz rekordok   | RRs answering the question
+---------------------+
|  Autoritás rekordok | RRs pointing toward an authority
+---------------------+
|  Ráadás rekordok    | RRs holding additional information
+---------------------+
```

Fejrész (header) mindig van, és a válasz mindig tartalmazza a kérdést, amire vonatkozik. A válasz, autoritás (authority) és ráadás (additional) szekciók DNS rekordokat tartalmazhatnak, de lehetnek üresek is.

A fejrész szerkezete:

```
  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      ID                       |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|QR|   Opcode  |AA|TC|RD|RA|   Z    |   RCODE   |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|               Kérdések száma                   |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

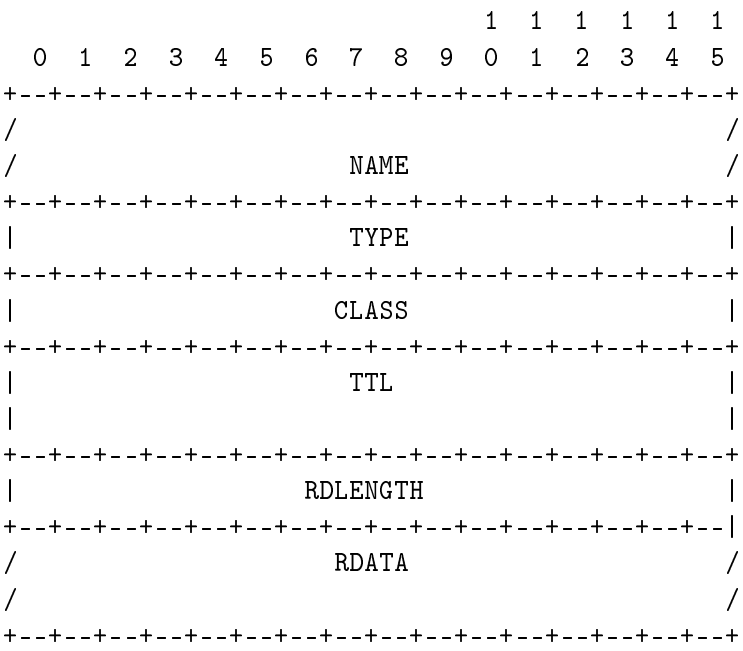


Az egyes mezők jelentése:

- ID: ennek segítségével lehet a kérdést és a választ párosítani
- QR: 0 ha kérdés, 1 ha válasz
- AA: A válasz autoritatív (Authoritative Answer)
- TC: A válasz csonkolt (Truncated)
- RD: Rekurziót kérek (Recursion Desired)
- RA: Rekurziót adok (Recursion Available)
- Rcode: a visszatérési érték: ha 0, siker

Amint látható, a fejrészből kiolvasható, hogy az üzenetben hány válasz, autoritás és ráadás rekord található.

Az egyes szekciókban vannak az úgynevezett RR-ek, resource rekordok. Ezek szerkezete:



Az egyes mezők jelentése:

- NAME, TYPE, CLASS: a rekord „bal oldala”, típusa, osztálya
- TTL: Time To Live. Ennyi másodpercig kell a cache-ben tartani a rekordot (4 byte)
- RDLENGTH: a rekordhoz tartozó adat hossza byte-ban
- RDATA: a rekordhoz tartozó adat. Formátuma függ a rekord típusától

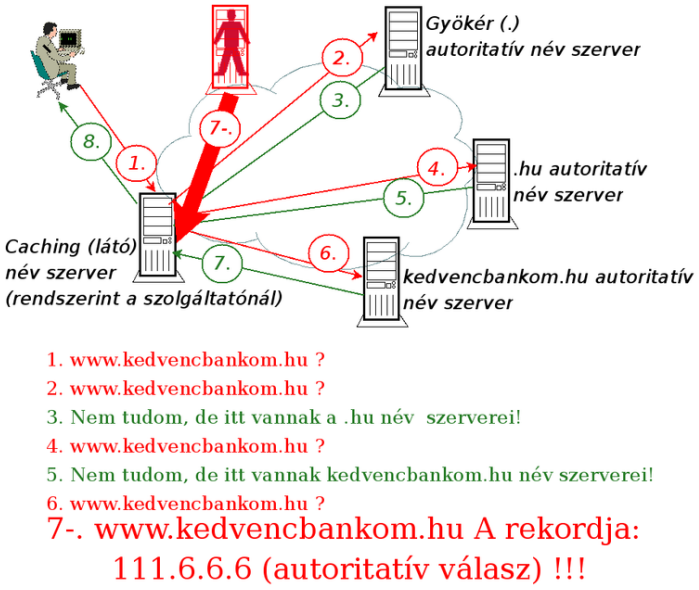
2. Mi indokolja a DNSSEC bevezetését?

A DNS **minden** internetes tevékenység — levelezés, webes böngészés, torrentezés vagy internetes telefonálás — működéséhez szükséges. A DNS működését veszélyek fenyegetik, a DNS ellen az évtizedek

során több jelentős sikeres támadást hajtottak végre. Ezért a DNS-t szükséges védeni minden internetes tevékenység/szolgáltatás védelme érdekében. A következőkben felsorolunk néhány DNS-t fenyegető veszélyt.

2.1. DNS cache poisoning / cache mérgezés

A 3. ábrán az látható, hogy egy imposztor autoritatív névszerver nevében válaszolhat, és hamis adatokat juttathat a rekurzív névszerver cache-ébe. A megfertőzött cache azután — attól függően, hogy milyen nagy felhasználói kört szolgál ki, és azok közül hányan kérik tőle a sikeresen megfertőzött nevet —, akár felhasználók tízezreit tévesztheti meg, és ezen felhasználók személyes adatai, kommunikációja, pénze mind a támadó kénye-kedvének vannak kitéve. Az ilyen támadáshoz szükséges, hogy a támadó a megszemélyesített autoritatív név szerver IP címét hamisítsa, és eltalálja a kérdésben szereplő paramétereket (UDP port, query ID). Ez nem lehetetlen külső (off-path) IP címről, de egészen triviális ha a támadó látja a támadni kívánt forgalmat (on-path), vagy abba bele is tud avatkozni (in-path).



3. ábra. DNS cache mérgezés

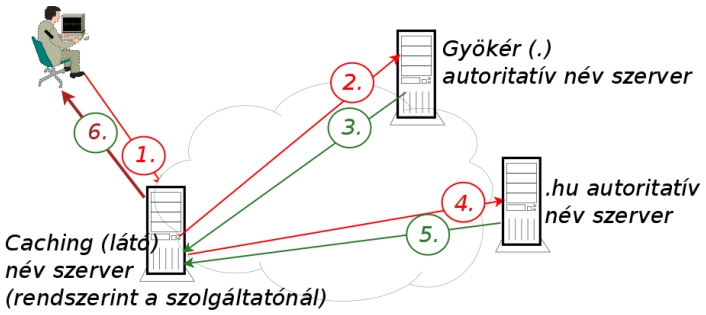
2.2. Hazug rekurzív névszerver

A 4. ábrán arra látunk példát, hogy a rekurzív névszerver hamis választ ad. Ez történhet rosszindulatú behatolás következtében, de a névszerver üzemeltető/tulajdonos szándékából is — például 2003-ban a Verisign követett el ilyet, amit nagy felháborodás követett [3], [4], [5] .

2.3. DNS választ hamisító router

Az 5. ábra azt a lehetőséget szemlélteti, amivel elsősorban internetszolgáltatók, illetve azok munkatársai, a munkatársak megbízói élhetnek: olyanok, akik valamiképpen hozzáférnek a DNS üzenetekhez miközben azok a hálózaton „utaznak”. Egy ilyen közbülső eszközben meg lehet tenni, hogy bizonyos kérdéseket

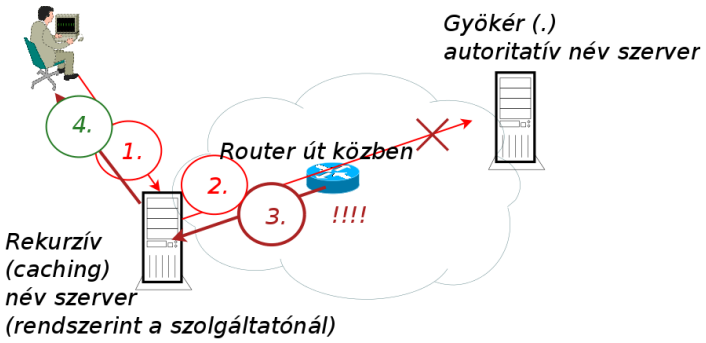
[3] `http://bcn.boulder.co.us/~neal/ietf/verisign-abuse.html` .
[4] `http://www.pfir.org/statements/vs-domain-abuse`.
[5] `https://archive.icann.org/en/topics/wildcard-history.html` .



1. gepelesihuba.hu ?
2. gepelesihuba.hu ?
3. Nem tudom, de itt vannak a .hu név szerverei!
4. gepelesihuba.hu ?
5. Nincs ilyen (NXDOMAIN)!
6. 11.22.33.44 = www.legjobb-akciok.hu !

4. ábra. A rekurzív névszerver csal

továbbítás helyett (vagy mellett) eleve megválaszoljunk, hamis válaszokat küldjünk vissza a kérdezőnek. Ez a támadás lehet rosszindulatú behatolók műve, de gyakran politikusok is rendelnek el ilyen manipulációt. Így működik többek közt a „kínai nagy DNS fal”: a kínai hatóságok bizonyos általuk veszélyesnek ítélt internet neveket az internetezők elől elrejtenek, meghamisítanak.



1. valakinek-nemtetszik.hu ?
2. valakinek-nemtetszik.hu ?
3. 11.22.33.55 = kamu-mas.hu !
4. 11.22.33.55 = kamu-mas.hu !

5. ábra. Közbülső router csal

3. A DNSSEC elve

DNSSEC segítségével védekezhetünk minden támadás, DNS üzenet hamisítás ellen, amikről az előzőekben szó volt. A DNSSEC fő ötlete, hogy a DNS **válaszokat** digitális aláírással látjuk el. Ilyen módon az üzenetek **hitelességét** (authenticity) és **sértetlenségét** (integrity) garantáljuk. Bizalmasság garantálása (titkosítás, confidentiality) nincs. Nem csak a tényleges DNS rekordokat, hanem a „nincs ilyen” válasz hitelességét is garantáljuk digitális aláírással. Ha a „nincs ilyen” üzenetek nem lennének digitális aláírással ellátva, akkor ezek hamisításával egyszerűen lehetne DoS támadást végrehajtani, ellehetetleníteni nevek feloldását. A DNSSEC véd a cache mérgezéstől, véd a DNS válaszok menet közbeni hamisítása ellen.

A DNSSEC egy másik fontos ötlete, hogy maguk az aláírások is DNS rekordok. Ilyen módon a DNSSEC

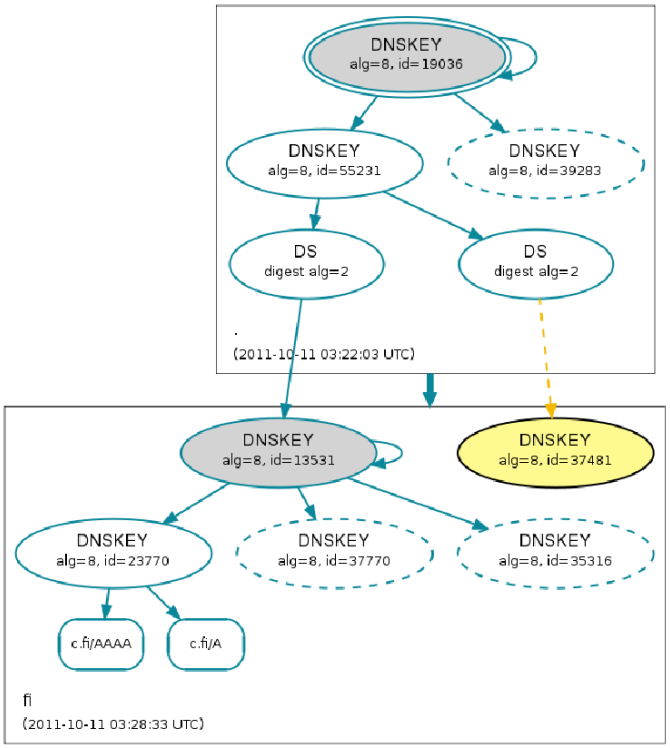
a DNS kiterjesztése.

A DNSSEC-nél az egyes zónákhoz tartozhat egy vagy több kulcspár. Ezen kulcsok titkos részével történik a rekordok aláírása, és a nyilvános részével az aláírások ellenőrzése. A kulcsok nyilvános része DNS rekordként jelenik meg.

3.1. Bizalmi lánc

A DNS delegáláshoz hasonlóan, a magasabb szinten aláírjuk a delegált zónában használt publikus kulcsot (pontosabban az abból képzett hash-t, a `DS` rekordot). Ilyen módon bizalmi lánc alakul ki: egy vagy több kiindulópontból, úgy nevezet **trust anchor**-ból digitális aláírások láncolatán át juthatunk el ahhoz az aláíráshoz ami egy-egy esetben a DNS információ hitelességét bizonyítja. Hasonlít ez ahhoz, ahogy például a webes böngészésnél az X.509 tanúsítványok láncolata hitelesít egy végső digitális aláírást, végső soron egy webhelyet. Nagy különbség, hogy a DNSSEC-nél nincs az X.509-hez hasonló PKI (Public Key Infrastucture), nincsenek CA-k (Certification Authority-k). DNSSEC esetén a rekurzív névszerverekbe be kell konfigurálni egy-vagy több nyilvános kulcsot, ezek lesznek a bizalmi lánc kiindulópontjai. Ha aláírt DNS rekorddal találkozunk, akkor az aláíró kulcs és egy trust anchor között folyamatos aláírt láncot kell találnunk.

A 6. ábra dnsviz.net-ről származik. Azt a bizalmi láncot mutatja, ami a gyökér zónához tartozó 19036 azonosítójú kulcstól kiindulva igazolja a `c.fi` rekordok hitelességét.



6. ábra. Bizalmi lánc a c.fi névig

3.2. DNSSEC a root zónában

A gyökér zóna is DNSSEC-cel védett 2010. óta, a gyökérhez is tartozik `DNSKEY` rekord. DNSSEC-et támogató rekurzív névszervernél legalább ezt a trust anchor-t meg kell adni a rekurzív névszerver kon-

figurációjában, nem elég csak az `NS` rekordot. A gyökér DNSKEY rekordot (vagy bármilyen más trust anchort) nem szabad DNS segítségével letölteni, arra alternatív módszert kell alkalmazni. Például a 2010. óta a mai napig (2015. március) érvényes gyökér zónához tartozó kulcs letölthető a hálózatról. [6] A gyökér kulcs publikálásának kérdéséről külön dokumentum készült. [7]

Aki a gyökér zóna titkos kulcsát birtokolja, az „mindent visz", ezért a gyökér zónához tartozó kulcso(ka)t különös gonddal kezelik. [8]

A DNSSEC autentikáció sikeres, ha igazolható aláírások és kulcsok láncolatával a kérdéses rekord (RRset) aláírása. A TLD-k évek óta sorban vezetik be a DNSSEC-et. A IANA folyamatosan figyelemmel kíséri és publikálja egyes országokhoz tartozó ccTLD-k DNSSEC státuszát. [9]

3.3. DNSSEC a rekurzív szerver oldalán

Ha egy DNSSEC-et használó rekurzív névszerver megtudott egy rekordot, akkor megnézi a rekordhoz (RRset-hez) tartozó aláírást is, autentikálja az RRset-et. Az autentikáció eredményeként egy rekord lehet:

- **Secure**: a bizalmi lánc szerint autentikált
- **Insecure:** a bizalmi lánc megszagad a trust anchor és a rekord között
  - Autentikált bizonyíték van arra, hogy hiányzik egy DS rekord
- **Bogus**: a bizalmi lánc azt mutatja, hogy:
  - Az aláírás hibát jelez
  - Hiányzik az aláírás
  - Az aláírás lejárt
  - Nem szupportált mezőt, algoritmust talált az ellenőrzés
- **Indeterminate:** nincs olyan trust anchor, ami a rekord *fölött* lenne

Amíg a DNSSEC nincs a nevek többségénél bevezetve, addig a nevek többsége insecure-nak mutatkozik. Ha a gyökér zónához tartozó trust anchort helyesen bekonfiguráltuk, akkor indeterminate nem lehet egy rekord sem.

4. DNSSEC segédeszközök, programok, webhelyek

4.1. dig

A `dig` klasszikus DNS nézegető program, a `dnsutils` csomag része, természetesen támogatja a DNSSEC-et. Erre szolgál a `+dnssec` kapcsoló. Ennek hatására bebillenti a kérdésben a `DO` bitet, vagyis a válaszban DNSSEC rekordokat is vár. A DNSSEC rekordokat barátságosan mutatja. Az időket `YYYYMMDDHHMM` formában, a kulcsokat base64-ben. Tudja mutatni az ellenőrzés teljes folyamatát a `+sigchase` kapcsoló segítségével. Miután a gyökérhez tartozó DNSKEY kulcsot a `/var/tmp/keys.key` fájlban elhelyeztük, kipróbálhatjuk a következő példát, ami a `.hu` SOA rekordjának ellenőrzését mutatja:

```
dig +sigchase -t soa hu. +trusted-key=/var/tmp/keys.key

;; RRset to chase:
hu. 3592 IN SOA a.hu. hostmaster.nic.hu. 2015061511 3600 900 259200 3600
```

[6] `https://www.iana.org/dnssec/file` .
[7] `http://data.iana.org/root-anchors/draft-icann-dnssec-trust-anchor.html` .
[8] `https://www.iana.org/dnssec/ceremonies` .
[9] `http://www.internetsociety.org/deploy360/dnssec/maps/` .

```
;; RRSIG of the RRset to chase:
hu. 3592 IN RRSIG SOA 8 1 3600 20150713115250 20150615083105 7023 hu. ioNGeBRxrZ1l+Mhr4lmV3fqHdqJo/



Launch a query to find a RRset of type DNSKEY for zone: hu.

;; DNSKEYset that signs the RRset to chase:
hu. 1100 IN DNSKEY 257 3 8 AwEAAcVWW5q+UNb2uxqiAWfSPhf1asbK36lfE/aCnwjhbrxnEi1R5Hc/ dIkYozJqf1Gw+Ac
hu. 1100 IN DNSKEY 257 3 8 AwEAAeCl6pKN64pTAiwa7yxZY+Vma+9Mu9ookNVMG9lNYr7WhiBp4tLC buE+JOdFGXnT3zu
hu. 1100 IN DNSKEY 256 3 8 AwEAAb5xzYP8qlEsvBGW/8PGR4Tm5j60ClWfevN2s3XzZvid1Ro9O0Ez sV27OmUYV5Sk4dv
hu. 1100 IN DNSKEY 256 3 8 AwEAAdI9HUvEVS1tz98bjd5m34Z0Th7TSH4auLbihZr+UcIaFLtbpa3j cqRtaTTaiazv0vC


;; RRSIG of the DNSKEYset that signs the RRset to chase:
hu. 1100 IN RRSIG DNSKEY 8 1 3600 20150713140955 20150615073104 18949 hu. 1YzPjuhUVdlazuxotwuf0knDI
hu. 1100 IN RRSIG DNSKEY 8 1 3600 20150713140955 20150615073104 20056 hu. WJq0yy4jxfNVqmgX51OPZf8DC



Launch a query to find a RRset of type DS for zone: hu.

;; DSset of the DNSKEYset
hu. 29747 IN DS 18949 8 2 DB876C0517541CA362A565DBF54FE58C230296B01B62611269AFFB5C 186A7D33


;; RRSIG of the DSset of the DNSKEYset
hu. 29747 IN RRSIG DS 8 1 86400 20150624170000 20150614160000 48613 . OKUn1aA6lSUIdOaxTVabNCNaz1bmN



;; WE HAVE MATERIAL, WE NOW DO VALIDATION
;; VERIFYING SOA RRset for hu. with DNSKEY:7023: success
;; OK We found DNSKEY (or more) to validate the RRset
;; Ok, find a Trusted Key in the DNSKEY RRset: 18949
;; VERIFYING DNSKEY RRset for hu. with DNSKEY:18949: success

;; Ok this DNSKEY is a Trusted Key, DNSSEC validation is ok: SUCCESS
```

4.2. drill

A `drill` az Nlnetlabs terméke. A program neve arra utal, hogy ha még mélyebbra akarunk hatolni, nem elég ásni, fúrni kell :-). Az `ldnsutils` debian csomag része. A `drill` alkotóinak kifejezetten DNSSEC nézegetés volt a célja. Látványos a `-S` (`sigchase`) kapcsoló:

```
drill -t ns -k /var/tmp/keys.key -S info.hu

;; Number of trusted keys: 2
;; Chasing: info.hu. NS
```

```
DNSSEC Trust tree:
info.hu. (NS)
|---info.hu. (DNSKEY keytag: 55609 alg: 8 flags: 256)
    |---info.hu. (DNSKEY keytag: 20527 alg: 8 flags: 257)
    |---info.hu. (DS keytag: 20527 digest type: 2)
        |---hu. (DNSKEY keytag: 56175 alg: 8 flags: 256)
            |---hu. (DNSKEY keytag: 18949 alg: 8 flags: 257)
            |---hu. (DNSKEY keytag: 20056 alg: 8 flags: 257)
            |---hu. (DS keytag: 18949 digest type: 2)
                |---. (DNSKEY keytag: 48613 alg: 8 flags: 256)
                    |---. (DNSKEY keytag: 19036 alg: 8 flags: 257)
;; Chase successful
```

4.3. DNS szerverek

A népszerű DNS szerverek mai változatai általában támogatják a DNSSEC-et: A Bind a legnépszerűbb DNS szerver, az ISC terméke rekurzív és autoritatív funkcióban is használható DNSSEC-cel. Az Nlnetlabs termékei különválasztják az autoritatív funkciót (NSD) és a rekurzív funkciót (Unbound). Mindkét program támogatja a DNSSEC-et. A Powerdns nagyrabecsült régi tagja a DNS szerverek családjának. Itt szintén két program valósítja meg az autoritatív és a rekurzív funkciót. Az autoritatív névszerver támogatja a DNSSEC-et, a rekurzív név szerverhez pedig egy kiegészítés, validator készül. [10] Az autoritatív névszerverek családjában új, igen figyelemreméltó program a Knot és a Yadifa. Mindkettő kezdetektől fogva támogatja a DNSSEC-et. D.J. Bernstein munkája a DJBDNS vélhetőleg **sose** fogja támogatni.

4.4. tcpdump, wireshark

Ha hibát keresünk, meg akarunk érteni valamilyen hálózati jelenséget, nagy segítséget jelentenek a `tcpdump` és a `wireshark` programok.

A `tcpdump` felismeri a DNSSEC-cel kapcsolatos rekordokat a `wireshark` a DNSSEC üzenetek szerkezetét is barátságosan, olvasmányosan tárja elénk. Például az idő adatokat úgy írja ki, ahogy azt ki szoktuk mondani. Ezt szemlélteti a 7. ábra.

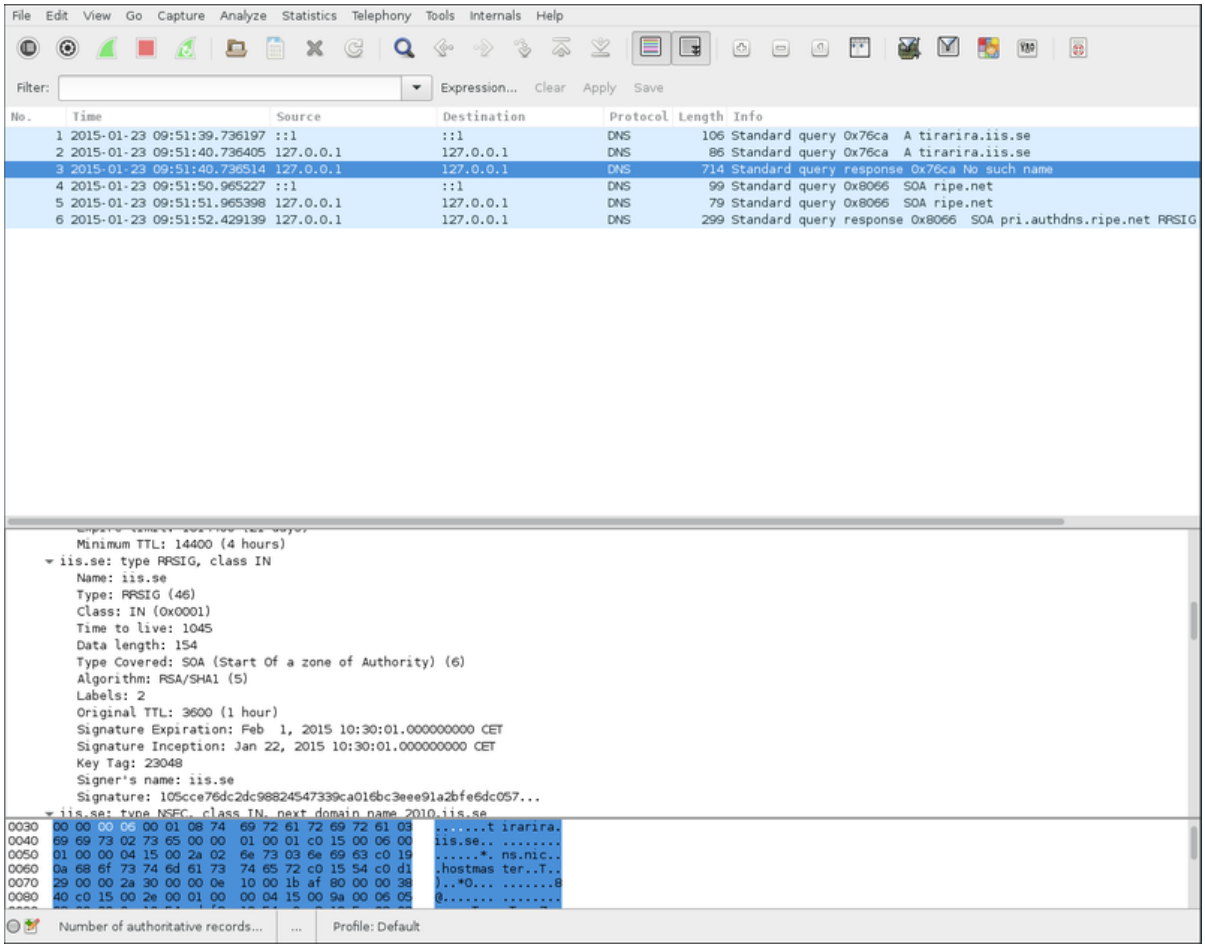
4.5. DNSSEC-et tudó, validáló, nyílt rekurzív névszerverek

A nyílt rekurzív névszerverek általában veszélyt jelentenek, de vannak gondosan kezelt, felügyelt nyílt rekurzív szerverek. Ezeknek például az lehet a célja, hogy alternatívát jelentsenek, ha nem tudunk rekurzív névszervert, vagy hogy teszteljük a DNS/DNSSEC beállításokat, és működést.

A DNS-OARC (DNS Operations, Analysis, and Research Center) mindenki által használható nyílt validáló rekurzív névszervert (Open DNSSEC Validating Resolver) üzemeltet. A konfiguráció és az IP címek:

Instance	IPv4	IPv6
BIND 9	149.20.64.20	2001:4f8:3:2bc:1::64:20
Unbound	149.20.64.21	2001:4f8:3:2bc:1::64:21

[10] Lásd: `http://blog.powerdns.com/2013/09/16/dnssec-validation-for-the-recursor/` .



7. ábra. A wireshark visszafejti a DNSSEC rekordok egyes mezőit

A sokak által használt Google publikus rekurzív DNS szerverek 2013. júliusa óta alapértelmezésben DNSSEC-et használnak:

IPv4	IPv6
8.8.8.8	2001:4860:4860::8888
8.8.4.4	2001:4860:4860::8844

4.6. Webhelyek

A DNSSEC működésével kapcsolatban igen sok webhely áll segítségünkre. Nagyon látványos a dnsviz.net szolgáltatása. Ez nem csak lekérdezi a DNS rekordokat, hanem egy rajzon megjeleníti az **aktuális** DNSSEC fát. Pirossal jelzi, ha hiba van. Tanulságos kipróbálni például a szándékosan hibásan konfigurált www.dnssec-failed.org és sigfail.verteiltesysteme.net neveket.

A dnssec.net DNSSEC-cel kapcsolatos információk gyűjtőhelye, a dnssec-deployment.org-ot pedig kifejezetten DNSSEC **terjedésével** kapcsolatos információknak szánták.

5. DNSSEC építőelemek

5.1. DNSSEC RFC-k

A DNSSEC-ről az első RFC 1997-ből származik (RFC2065). Az elképzelést azóta többször átdolgozták, bővítették. Jelenleg az irányadó legfőbb RFC-k: RFC4033 - RFC4035. Újabb RFC-k bővítik, kiegészítik az DNSSEC ajánlást:

- RFC4641: DNSSEC Operational Practices
- RFC5011: Automated Updates of DNS Security (DNSSEC) Trust Anchors
- RFC7344: Automating DNSSEC Delegation Trust Maintenance
- RFC6698: The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA
- RFC6781: DNSSEC Operational Practices, Version 2

5.2. EDNS0

EDNS0, azaz Extended DNS (RFC2671, melyet az RFC6891 felülbírált) kibővíti a DNS üzenet fejrészét. Ez úgy válik valóra, hogy bevezet egy pszeudorekordot: `OPT` (options). Az `OPT` rekord sose jelenik meg tényleges DNS rekordként, arra szolgál, hogy a DNS fejrészt „folytatni lehessen". A rekord opciókat tartalmaz, melyeknek szerkezete: `Code, Length, Data`. Az `OPT` pszeudorekord `Class` paramétere egy különösen fontos opció, az **UDP size**. A DNS üzenet hossza eredetileg legfeljebb 512 byte-ban volt korlátozva. A DNSSEC-cel jelentősen megnő az üzenetek hossza, ezért ebben a mezőben a csomag küldője közölheti, hogy mekkora UDP csomagokat képes fogadni. Jellemző értékek: 1024, 2048, 4096. Érdemes tudni, hogy nagy UDP csomag IP fragmentálást eredményez, ami nem feltétlenül kívánatos.

Az EDNS0 segítségével új flag-ek bevezetése válik lehetővé. Az OPT pszeudorekordban kapott helyet a DO bit.

5.3. DNSSEC flag-ek

A DNSSSEC működéshez a következő flag-ek bevezetése szükséges:

- `DO` (Dnssec Ok)

Ez a flag kérdésben használatos. Jelentése: kérem a DNSSEC rekordokat is.

- `CD` (Checking Disabled)

Ez a flag is kérdésben használatos. Azt jelenti: te ne ellenőrizz, majd én. Ilyenkor tehát a validáló rekurzor visszaadja az olyan rekordot is, amiknek a digitális aláírása hibás. Ez például akkor hasznos, ha a stub rezolver oldalán akarunk DNSSEC ellenőrzést végezni. Akkor is jó hasznát vehetjük, ha debugolunk: miért nem oldja fel a DNSSEC rekurzív névszerverünk a kérdezett nevet? A `CD` flag-et a `drill -o CD`, illetve a `dig +cdflag` opciókkal lehet kérni.

- `AD` (Authenticated Data)

Az AD bit válaszban használatos. Az a jelentése, hogy a rekurzív névszerver ellenőrizte és rendben találta DNSSEC szerint a választ. Az `AD` bit ellentétes az `AA` bittel, többet jelent mint az `AA`. Az `AD` bitet rekurzív névszerverek, az `AA` bitet autoritatív névszerverek adják vissza.

5.4. RR-set-ek

RR-set DNS rekordok egy olyan halmaza, ami egy zóna összes olyan rekordját tartalmazza, amiknek bal oldala (`name`), típusa (`type`) és osztálya (`class`) megegyezik.[11]

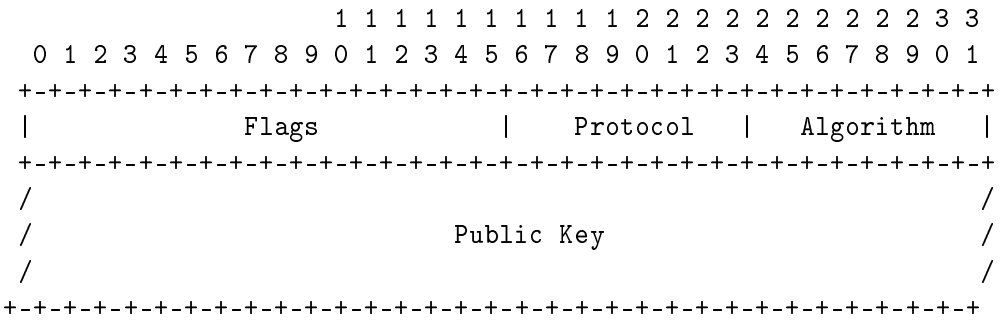
Például ez a **négy** rekord **két** RR set:

```
ppke.hu. SOA ns.ppke.hu. hostmaster.ppke.hu. 2010110901 7200 3600 604800 86400
ppke.hu. NS apa.btk.ppke.hu.
ppke.hu. NS ns2.iif.hu.
ppke.hu. NS ns.ppke.hu.
```

A két RR-set közül az egyikben egyedül a SOA rekord van, a másikban pedig a három NS rekord. DNSSEC-nél soha nem egyes rekordokat, hanem mindig RR-seteket írunk alá.

5.5. DNSKEY rekord

A DNSKEY rerkord az aláírásnál használt kulcspár **nyilvános** részének tárolására szolgál. A **titkos** párját lehetőleg még a DNS szerveren **sem** tároljuk: a célszerű gyakorlat az, hogy az aláírt zóna más gépen keletkezzen mint ahol szolgáltatjuk. A DNSKEY rekord szerkezete:



Az egyes DNSKEY mezők:

A *flags* mezőben a 7-es bitnek (decimális 256) állnia kell, ha DNS rekordok aláírására használt rekordról van szó, azaz gyakorlatilag mindig áll.

A *Protocol* mező értéke gyakorlatilag mindig 3 (egyelőre?).

Az *Algorithm* mező tartalmazza a kulccsal használt algoritmus kódját. Meghatározza, hogy hogyan kell értelmezni a ‚Public Key' mezőt. Az RFC4034 óta újabb algoritmusokat tettek megadhatóvá, ezekről szól az RFC5702. A mindenkori érvényes lista elérhető itt: `http://www.iana.org/assignments/dns-sec-alg-numbers`. Érdemes tudni, hogy a SHA1 algoritmus sérülékenynek bizonyul, elavult.

A *Public key* tartalmazza a nyilvános kulcsot. Ez bináris adat, a segédprogramok base 64-ben írják ki.

Vegyük észre, hogy nincs a kulcsnak lejárati ideje! Ez szokatlan azoknak, akik eddig például X.509 kulcsokat láttak. Persze a kulcs használója bármikor rendelkezheet úgy, hogy kulcsot cserél, de ez tisztán eljárásrend (policy) kérdése, magának a kulcsnak a lejárati idő nem része.

Íme egy példa DNSKEY rekordokra, pontosabban egy RR-set-re:

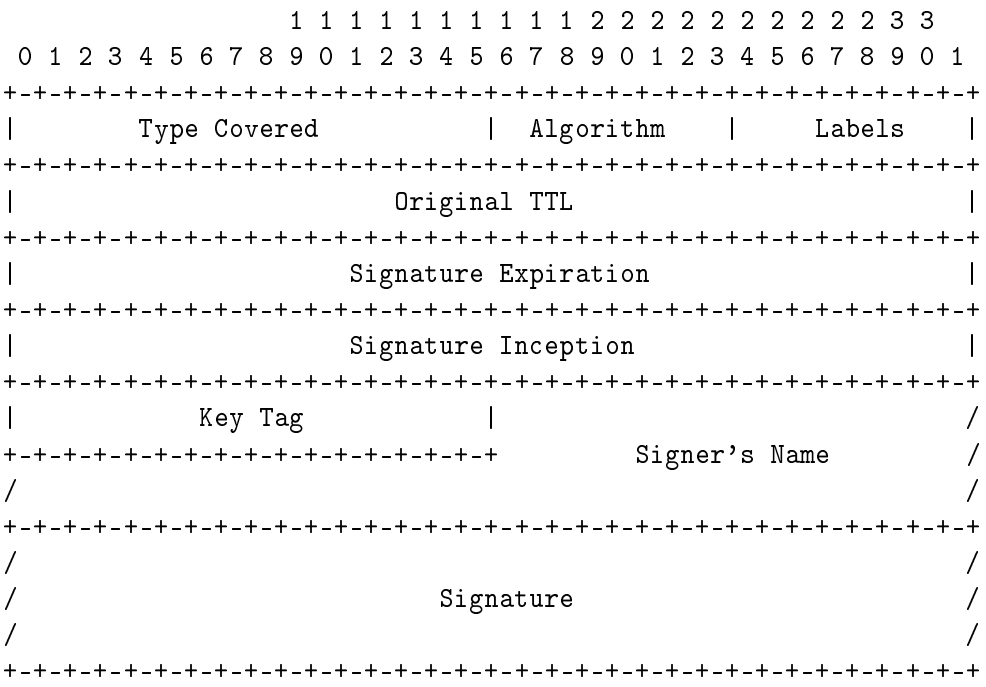
```
$drill -t dnskey fi.
;; ->>HEADER<<- opcode: QUERY, rcode: NOERROR, id: 25555
```

[11] Gyakorlatban az osztály szinte mindig `IN` (internet).

```
;; flags: qr rd ra ; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;; fi.  IN      DNSKEY

;; ANSWER SECTION:
fi.     3600    IN      DNSKEY  256 3 8 AwEAAZ71W+kyJO2KogLDTW2fXJWmirTuSsuX455BtQNKc+/zAI74whZq4s
fi.     3600    IN      DNSKEY  256 3 8 AwEAAb6XCtPWud5ROpcxj91zCJszffNqneOB3vitEDGf8NghhOBUYO3RUJ
fi.     3600    IN      DNSKEY  257 3 8 AwEAAYZ4Afrzb41QsPqOrYn3hfQWSF5FvXsTY3cvMOSGFnzF4CxAldYpiW
```

5.6. RRSIG rekord

Az RRSIG egy-egy RR-set (nem rekord!) aláírására szolgál. A DNSKEY RR-setet is aláírjuk. Az RRSIG rekord szerkezete:



Az egyes RRSIG rekord mezők:

A *Type covered* mutatja az aláírt RR-set típusát.

Az *Algorithm* az aláírásnál használt algoritmus. Erről — ahogy azt a `DNSKEY` rekordnál is említettük a `http://www.iana.org/assignments/dns-sec-alg-numbers/` címen lehet aktuális információt nyerni.

A *Labels* mező az aláírt rekordban szereplő 'label'-ek száma. [12] Kiderül belőle, hogy wildcard aláírásról van-e szó. Például ha van `*.example.com A` rekord, és a kérdés a `valami.example.com`, akkor a visszaadott A rekordban *Labels = 2* lesz, és az ellenőrzést ennek megfelelően a `*.example.com`-al kell folytatni.

Original TTL: az aláíráskor ennyi (volt) az aláírt rekord TTL értéke.

Signature Expiration és Inception 32 bites mezők, amik az aláírás érvényességének végét és kezdetét jelzik, az 1970. január elseje 0 óra után eltelt másodpercekkel kifejezve. A segédprogramok könnyebben érthető formában írják ki.

Key tag: 16 bites mező, ami a kulcs azonosítására szolgál. A nyilvános kulcsból keletkezik, úgy, hogy annak 16 bites darabjait összeadjuk. Értékeke tehát egy 65536-nál kisebb szám. Előfordulhat (bár nem

[12] DNS szóhasználatban *label* a dns névben két pont közötti rész.

valószínű), hogy két különböző kulcshoz ugyanaz a key tag tartozik.

Signer's name: az aláíró kulcs (DNSKEY) bal oldala.

Signature a tényleges aláírás. Bináris adat, a segédprogramok base64-ben írják ki.

Példa RRSIG rekordra:

```
ns1.stockholm.se.       3600    IN      RRSIG   A 5 3 3600
            20101219110002 20101119110002 54123 stockholm.se. prI7cdXdJLg3kLsYPi6vavi9UFlGq7pKNbjlF
            ;{id = 54123}
```

5.7. Autentikus „nincs ilyen" válasz

5.7.1. NSEC rekord

Az eredeti, 4033-as RFC-ben ez volt az egyetlen mód, hogy a „nincs ilyen rekord" (`NXDOMAIN`) választ hitelesítsük. Azóta — amint azt látni fogjuk — más lehetőség is van. Az NSEC rekord alkalmazásánál egy zóna létező rekordjait, pontosabban azok bal oldalait, a definiált **neveket**, lexikografikus sorrendbe rendezzük úgy, hogy egy kört alkossanak: az utolsó után újra az első következik. Az így keletkezett lánc minden egyes közéhez képezünk ezek után egy-egy NSEC rekordot. Ha nincs olyan rekord, amit kérdeznek, akkor olyan NSEC a válasz, ami annak az intervallumnak a két végét tartalmazza, ahova a nem létező rekord tartozna. Ilyen módon nem kell röptében generálni az NSEC rekordokat. (Ez általában nem is lenne lehetséges, hiszen a legtöbb esetben a DNS szerveren nincs is ott az aláíráshoz szükséges titkos kulcs! Ráadásul mivel az aláírás erőforrásigényes, DoS támadásra is módot adna, ha röptében kellene generálni az aláírásokat.) Az NSEC rekord hitelességét természetesen az az NSEC-hez tartozó RRSIG rekord igazolja.

NSEC rekord mezők:

```
                     1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                      Next Domain Name                         /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                       Type Bit Maps                           /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

A *Next domain name* mező tartalmazza a lexikografikosan következő nevet, az NSEC intervallum jobb oldalát.

Type bit maps: változó hosszúságú mező, ami mutatja, hogy a kérdezett névhez, az NSEC intervallum bal oldalához milyen típusú DNS rekordok **léteznek**. Ezzel lehet bizonyítani, hogy **a kérdezett** típus nem létezik. A mező úgy keletkezik, hogy a 16 bites `RR type` mezőt 256 darabra osztjuk, ez felel meg a 16 bit első byte-jának. Egy byte-on kódoljuk a window block sorszámát, és ez után egy byte-on azt, hogy hány byte-os bitmap következik. Ilyen módon egy ilyen bitmap hossza legfeljebb 256 bit, azaz 32 byte lehet. Formálisan, az RFC-t idézve:

```
Type Bit Maps Field = ( Window Block # | Bitmap Length | Bitmap )+
```

Íme egy példa NSEC válaszra:

```
; <<>> DiG 9.7.1-P2 <<>> +dnssec poipoi.se
;; global options: +cmd
```

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 23156
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 6, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;poipoi.se.                     IN      A

;; AUTHORITY SECTION:
se.                     6225    IN      SOA     catcher-in-the-rye.nic.se. registry-default.nic.se.
se.                     6225    IN      RRSIG   SOA 5 1 172800 20101128104814 20101120210441 26401
se.                     6225    IN      NSEC    0-0.se. NS SOA TXT RRSIG NSEC DNSKEY
se.                     6225    IN      RRSIG   NSEC 5 1 7200 20101127051349 20101119130441 26401 s
poiphone.se.            7196    IN      NSEC    poirot.se. NS RRSIG NSEC
poiphone.se.            7196    IN      RRSIG   NSEC 5 2 7200 20101126130810 20101118150540 26401 s
```

Ez a példa azt demonstrálja, hogy a `poipoi.se` név nem létezik. Az első NSEC rekord azt bizonyítja, hogy nincs `*.se` rekord. A második pedig azt mutatja, hogy ez a név a `poiphone.se` és a `poirot.se` nevek közé esik, amik léteznek és egymást követik az ABC szerint rendezett `.se` zónában. A `poiphopne.se` név létező típusai a `.se` zónában NS, RRSIG és NSEC. Mindkét NSEC rekord természetesen alá van írva a `.se` kulcsával.

Az előző példában szereplő NSEC rekord jobb oldalához szintén tartozik NSEC:

```
; <<>> DiG 9.7.3 <<>> +dnssec -t nsec poirot.se
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 41847
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 11, ADDITIONAL: 9

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 1480
;; QUESTION SECTION:
;poirot.se.                     IN      NSEC

;; ANSWER SECTION:
poirot.se.              7200    IN      NSEC    pois.se. NS RRSIG NSEC
poirot.se.              7200    IN      RRSIG   NSEC 5 2 7200 20111024051325 20111010110604 63576 s
```

Ilyen módon sorra tudunk venni `NSEC` rekordokat, a zónát **letapogathatjuk**. Hiába tiltott a zóna transzfer, az egész zónát meg lehet kapni. Több utility van, ami ezt csinálja. Például ezzel a paranccsal bárki letapogathatja az `arin.net` zónát:

```
ldns-walk arin.net @u.arin.net
```

5.7.2. Az NSEC rekord alternatívája: NSEC3

A zóna letapogatás lehetősége mellett az NSEC-cel más gondok is vannak: az NSEC rekord használatával aláírt zóna hatalmasra dagad. A méretnövekedés elsősorban az alkalmazott algoritmusoktól és az

aláíró kulcs méretétől függ. Átlagos paramétereket használva a `.hu` zóna például kb. hétszeresére nőtt. Ezen problémák kiküszöbölésére találták ki az NSEC3 technológiát, alkották meg 2008. márciusában az RFC5155-öt.

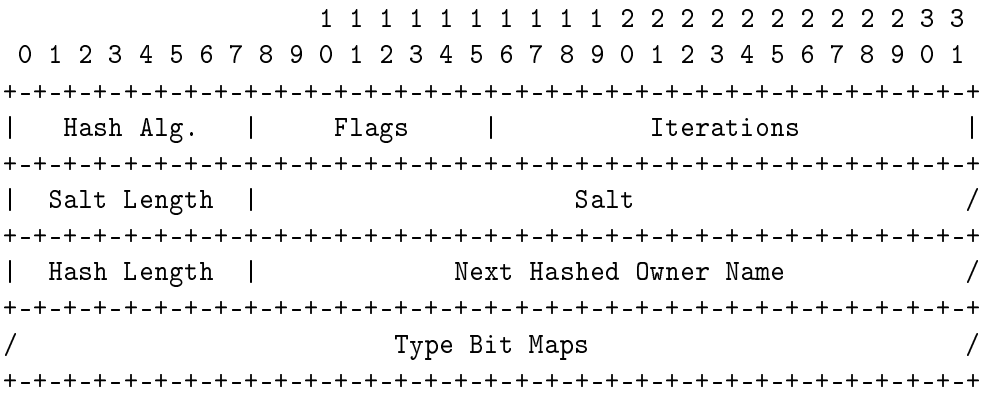
Az NSEC3 fő ötlete, hogy nem a neveket, hanem a nevekből képzett **hash**-eket rendezzük lexikografikusan. Ezek után a *nem létezést* az bizonyítja, hogy a kérdezett név hash-e két létező hash közé esik. A zóna letapogatás ilyenkor azért nehéz, mert hogy miből keletkezett a hash, azt legalább is nehéz kitalálni. Ha valaki tudni akarja, hogy egy zónában milyen nevek vannak, akkor könnyebben célt ér, ha egy szótár alapján próbálgatja a neveket. Persze az NSEC3 rekordok hitelességét éppen úgy RRSIG rekord igazolja, mint az NSEC rekordokét.

Az NSEC3 másik előnye, hogy delegálásokat tartalmazó zónáknál (delegation only zones, amilyenek például a TLD-k) nem szükséges minden névből hash-t és hash-közt képezni, csak a **DNSSEC delegálásokból**, vagyis azokból ahol DS rekord is áll. Ilyen módon a A *következő* a következő *DNSSEC-cel védett* delegálást jelenti. A zóna hossza nem nő csak a DNSSEC delegálások számával arányosan.

Opt-in-nek nevezzük azt a működési módot, amikor a zóna minden nevéből képezünk hash-t, és minden közből NSEC3 rekordot, amit aláírunk. Ez tehát lényegében olyan mint az NSEC, de a zone walking-ot megakadályozza.

Opt-out-nak nevezzeük azt a működési módot, amikor egy zónában csak az autoritatív nevekből és a **DNSSEC delegálások**-ból képezünk hash-t, és a közökből NSEC3 rekordot, amit aláírunk. Ez nagy segítség a sok delegálást tartalmazó zónáknál. Opt-out-ot csak *delegation-only* zónáknál érdemes használni.

Az NSEC3 rekord formátuma:



Az egyes mezők jelentése:

A *Hash algoritmus* mutatja, hogy milyen algoritmussal képezzük az NSEC3 rekordokhoz a hash-t. A lehetséges értékekről és jelentésükről szóló táblázat: `http://www.iana.org/assignments/dns-sec-alg-numbers/`.

A *flag*-ek közül az 1-es helyiértékű használt: **opt-out** Ha ez áll, akkor több aláíratlan **delegálás** lehet a két hash között.

Itarations: ennyiszer használjuk egymás után a hash algoritmust. Ha többször használjuk, akkor nehezebb egyező nevet előállítani, de nagyobb erőforrást igényel ellenőrzéskor a rekurzív névszervernél az ellenőrzés, és `NXDOMAIN` válasznál az autoritatív névszervernél a válasz generálása.

Salt: az eredeti nevet a végén kiegészítjük ezzel a karaktersorozattal, és ebből képezzük a hash-t. Ilyen módon dictionary támadás ellen védekezünk.

Itt egy NSEC3 példa:

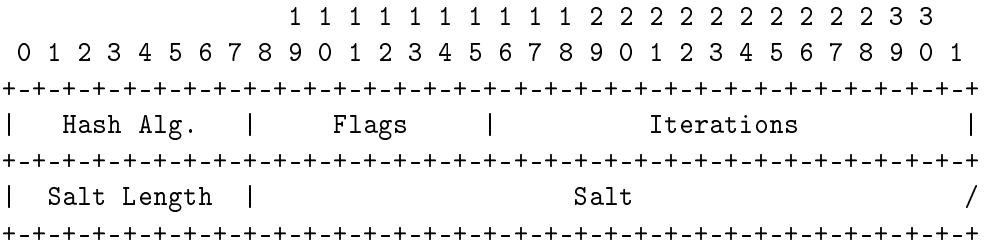
```
%dig +dnssec -t txt 2000.hu @193.239.149.6
```

```
; <<>> DiG 9.9.5-9-Debian <<>> +dnssec -t txt 2000.hu @193.239.149.6
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52054
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;2000.hu.                       IN      TXT

;; AUTHORITY SECTION:
2000.hu.                3600    IN      SOA     a.hu. hostmaster.nic.hu. 2015030900 3600 900 259200
2000.hu.                3600    IN      RRSIG   SOA 8 2 3600 20150406045739 20150309043936 7023 200
p22n6lal2cctol7cqg3nd83gv0gm15r3.2000.hu. 3600 IN RRSIG NSEC3 8 3 3600 20150327120505 201502270300
p22n6lal2cctol7cqg3nd83gv0gm15r3.2000.hu. 3600 IN NSEC3 1 1 5 757074175CBDE350 P22N6LAL2CCTOL7CQG3

;; Query time: 4 msec
;; SERVER: 193.239.149.6#53(193.239.149.6)
;; WHEN: Wed Mar 11 17:05:52 CET 2015
;; MSG SIZE  rcvd: 511
```

Az aláírt NSEC3 rekord azt bizonyítja, hogy a 2000.hu névhez tartozik NS, SOA, RRSIG, DNSKEY és NSEC3PARAM rekord (pontosabban RRset), de nem tartozik TXT.

5.8. Az NSEC3PARAM rekord

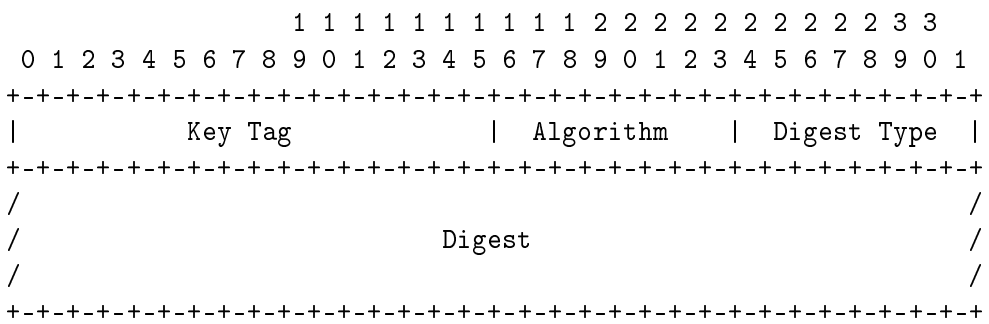
Ez a rekord az NSEC3 rekordoknál használt paramétereket tartalmazza. A zóna elején szereplő rekord, ennek alapján jönnek létre az egyes NSEC3 rakordok. Az NSEC3 rekordnál már megismert mezőket tartalmazza. A rekurzív névszervereknek nincs igazán dolguk ezzel a rekorddal. A rekord formátuma:



5.9. DS rekord

Ez a rekord nem más, mint az apuka zónában a gyerek DNSKEY-jéből képzett hash. Ennek az aláírásával valósul meg a bizalmi lánc. A DS rekordnak nem szabad a gyerek zónában előfordulni — ott a megfelelő DNSKEY rekord áll. A DNSKEY flag mezőjében a 15. bit (aminek helyi értéke 1) mutatja, hogy ebből a rekordból DS rekordot szándékozik képezni a gazdája. Ezt nevezik **secure entry point**-nak (`SEP`). A delegált zónára vonatkozó DS rekord autoritatív az apuka zónában, az NS rekord nem. Ezért a DS rekordot az apukában aláírjuk, az NS rekordot nem.

A rekord formátuma:



A *key tag* és *algorithm* mező a megfelelő DNSKEY rekordból származik. A *digest type* mező lehetséges értékei bővültek az RFC4034 óta. A SHA1 digest algoritmus elavult.

Példa DS rekordra:

```
praha.cz.               18000   IN      DS      28371 5 1 D0E13FEB1A9E1261E930338EC4C94934A19972A8
```

5.10. DNSSEC kulcs fajták: KSK és ZSK

Elvben egyetlen kulcspár, egyetlen DNSKEY rekord elég egyetlen zónában. A gyakorlatban és az ajánlások szerint érdemes két fajta rekordot használni.

KSK, Key Signing Key az a rekord, amiből DS-t képzünk, aminek áll a SEP (1-es helyi értékű) bitje, rendszerint hosszabb, nagyobb biztonságot nyújtó kulcs és ritkán változik — jellemzően néhány évenként. A KSK-val általában csak a DNSKEY RR-set-et írjuk alá, így igazoljuk a láncban következő ZSK hitelességét. Az apuka zónában a KSK-ból képzett DS rekordot írják alá. A KSK változtatás nem egészen egyszerű feladat: az apuka zóna gazdájának a közreműködését is igényli.

ZSK, azaz Zone Signing Key szolgál arra, hogy a DNSKEY rekord kivételével a zóna rekordjait aláírjuk. Ebből nem képezünk DS rekordot, a SEP (`1`-es helyi értékű bit) értéke `0`. Ezért ennek a kulcsnak a cseréje könnyen automatizálható, gyakran változhat — jellemzően néhány havonként. A kulcs mérete jellemzően kisebb, így gyorsabbak a kulccsal végzett kriptográfiai műveletek, és főleg kisebbek az aláírás rekordok. A kisebb rekordok kisebb DNS csomagokat eredményeznek, így jó esély van rá, hogy a DNSSEC válaszok egyetlen UDP csomagban elférnek.

5.11. Időzítések

Egy autentikált DNS rekord érvényességét sokféleképpen elvesztheti:

- Lejár a TTL
- Lejár az aláírás TTL-je
- Lejár az aláírás érvényességi ideje

A legkisebb számít: akármelyik jár le, a rekurzív névszerver már nem válaszol a cache-ből. Ha újra kérdezik, újra lekéri azt, ami hiányzik és ellenőriz.

5.12. Kulcs csere (key rollover)

Egy DNSSEC-cel védett zónában új kulcsra áttérni nem triviális feladat. Gondoskodni kell arról, hogy folyamatos legyen a működés:

- Új kulccsal ne kezdjünk el aláírni, mielőtt az el nem terjed aláírt DNSKEY RR-set-ben
- Ne vegyünk ki régi kulcsot, amíg még cache-elve érvényes aláírás lehet azzal

- KSK kulcs cserénél gondoskodjunk róla, hogy a megfelelő DS rekord megjelenjen az apuka zónában

Az RFC6781 (DNSSEC Operational Practices) hosszan tárgyalja a kulcs csere kérdését. Két módszert mutat be. [13]

- **Pre-publish** módszer: az új kulcsot jó előre publikáljuk, akkor kezdjük el használni, ha már mindenütt jelen van (propagation delay + Max Zone TTL). [14] Ekkor a régi kulccsal történő aláírásokat felválthatjuk — egyszerre, vagy fokozatosan — az új kulcs aláírásaival. Amikor minden aláírás már az új kulccsal van jelen mindenhol, vagyis a cache-ekből régi RRSIG-ek kiürültek, akkor lehet a régi kulcsot kivenni a zónából. Ezt a módszert érdemes használni ZSK rollover-nél.
- **Double signature** módszer: az új kulccsal **is** aláírunk. Ilyenkor az új DNSKEY megjelenésekor azonnal elkezdhetünk aláírni vele. Ha KSK-ról van szó, akkor az új kulcshoz tartozó DS rekordot be kell tenni az apuka zónába. Megfelelő idő eltelte után a régi kulccsal történő aláírás(oka)t elhagyhatjuk. Amikor már semmilyen cache-ben nincs a régi kulccsal készült aláírás, akkor törölhetjük a kulcsot a zónából, KSK esetén ezek után az apuka zónából elhagyható a megfelelő DS rekord is.

ZSK-val kapcsolatban általában pre-publish, KSK-val kapcsolatban a double signature módszer használatos. Az OpenDNSSEC nagy segítségünkre van kulcs cseréknél: ZSK esetén teljesen automatikusan végrehajtja, KSK esetén pedig félig automatikusan: a DS rekord apukába történő elhelyezéséről külön kell gondoskodnunk, de nem engedi, hogy elrontsuk a dolgot: addig nem veszi teljes használatba az új rekordot, amíg egy erre szolgáló külön paranccsal nem hoztuk a tudomására, hogy a DS ott van az apuka zónában.

5.13. Automatikus trust anchor update - RFC5011

A rekurzív validáló (DNSSEC-et értő) névszerverekben az ellenőrzés archimédeszi pontja (pontjai) a trust anchor(ok), ahogy arról már fentebb szó volt. Ha egy ilyen DNSKEY változik, akkor az összes ezt használó rekurzív névszerverben kézzel kell beírni az újat — hacsak nem használunk RFC5011 szerinti automatikus trust anchor update-et.

Az RFC5011 alapgondolata, hogy a régi KSK-val alá lehet írni egy frissen generáltat. Ezzel az aláírással jelezheti egy KSK gazdája, hogy rollover-t hajt végre. Egy ideig mindkét KSK a zónában van. A rezolverek az új KSK-t is beteszik/betehetik trust anchor-nak. Egy idő után a régi KSK-t ki lehet venni a DNSKEY RRset-ből és a rezolvereknél a trust anchor-ok közül.

Felmerül azonban egy új gond. Ha két KSK közül akár csak az egyik egy imposztor birtokába jut, akkor azzal mindkét kulcsot érvénytelenné teheti. Ez ellen való védekezésül az RFC bevezet egy új DNSKEY flag-et: ez a revoke bit, a 8., aminek helyi értéke 512. Egy másik elv, hogy ha megjelenik egy új KSK, akkor azt egy ideig (AddPend, vagy Holddown time) nem veszi be trust anchornak a rekurzív név szerver.

Az RFC5011 szerinti kulcs állapotok:

```
                          NEXT STATE
        ------------------------------------------------
 FROM   |Start  |AddPend |Valid  |Missing|Revoked|Removed|
-------------------------------------------------------------
Start   |       |NewKey  |       |       |       |       |
-------------------------------------------------------------
AddPend |KeyRem |        |AddTime|       |       |       |
```

[13] Tovább finomítják, pontosítják a módszereket a következő internet draft-ok:
`https://tools.ietf.org/html/draft-ietf-dnsop-dnssec-key-timing-06` és
`https://tools.ietf.org/html/draft-mekking-dnsop-dnssec-key-timing-bis-00` .

[14] A *propagation delay* az az idő, ameddig valamennyi autoritatív szerver átveszi a teljes zónát, *Max Zone TTL* a zónában levő legnagyobb TTL idő.

```
-----------------------------------------------------------
Valid   |       |        |       |KeyRem |Revbit |       |
-----------------------------------------------------------
Missing |       |        |KeyPres|       |Revbit |       |
-----------------------------------------------------------
Revoked |       |        |       |       |       |RemTime|
-----------------------------------------------------------
Removed |       |        |       |       |       |       |
-----------------------------------------------------------
```

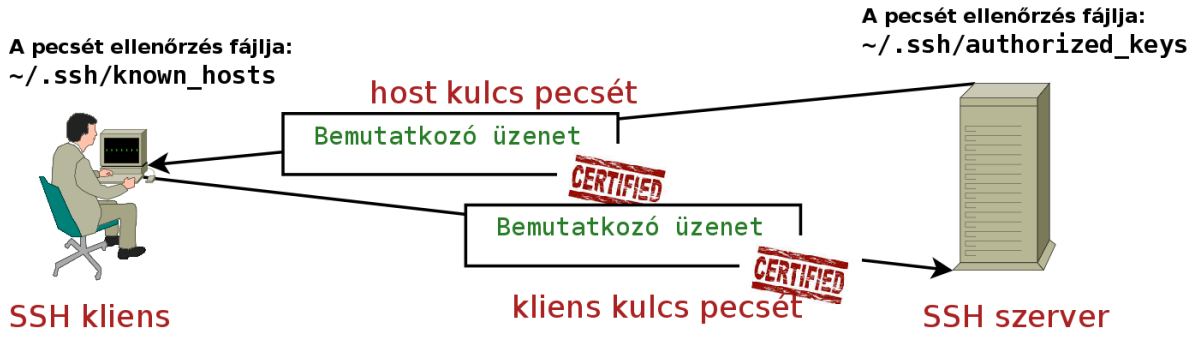
State Table

6. DNSSEC alkalmazások

A DNS rekordok digitális aláírásának több közvetlenül használható előnye van biztonságot kívánó protokolloknál, például az SSH és a TLS protokolloknál.

6.1. Ssh és host kulcsok

A 8. ábra mutatja, hogy ssh kommunikáció esetén a kliens és a szerver kölcsönösen digitális aláírással igazolják magukat.



8. ábra. SSH kapcsolatfelvétel: autentikálják egymást a kommunikáló felek

Bizonyára sokan találkoztak már azzal a problémával, hogy ha szkriptekben akarunk ssh parancsot kiadni, akkor először kézzel kell a kliens oldalon kapcsolatot teremteni a szerverrel, hogy a szerver nyilvános kulcsa az `.ssh/known-hosts` fájlba bekerülhessen. Ezen a problémán is segít a DNSSEC.

Az RFC4255 bevezeti az SSHFP rekordot, ami egy szerver nyilvános SSH kulcsát tartalmazza. Az SSHFP rekordban és így a szerver ssh kulcsában megbízhatunk, **ha** DNSSEC-cel védett. Nem kell „kézzel" beavatkoznunk, ha új ssh kulccsal találkozunk például upgrade miatt. Szkriptekben is biztonsággal kiadhatunk távoli gépekre `ssh` parancsot. Az Openssh régen támogatja az SSHFP rekordot, **feltéve**, hogy az DNSSEC-cel védett.

6.2. Az X.509 PKI

Egy webhely (levelezőszerver, VPN kliens stb.) hitelességét X.509 szabvány szerinti tanúsítvány „igazolja".[15] Az X.509 tanúsítványok egy aláírt nyilvános kulcsot tartalmaznak. A bizalmi lánc kiindulópontjai a

[15] Érdemes idézőjelbe tenni az „igazol" szót, mert a dolog gyenge lábakon áll.

CA-k: Certification Authority-k, a hivatásos tanúsítványkibocsátók. Minden böngésző (levelezőkliens stb.) konfigurációja tartalmazza a root CA tanúsítványok egy listáját, és a root tanúsítványok által közvetlenül vagy közvetve aláírt tanúsítványokat hitelesnek fogadja el a program. Tanúsítványkibocsátók hibázhatnak és lehetnek hanyagok. De ennél több is igaz: sokszor beigazolódott, hogy a tanúsítványkibocsátók kormányok és bűnözők hatására hamisítanak. Ilyen felfedezett eset például a Comodo [16], vagy a Diginotar [17] esete.

Ezen a problémán is tud segíteni a DNSSEC.

A **DANE** (DNS-based Authentication of Named Entities) arra szolgál, hogy DNSSEC aláírással igazoljuk az X.509 tanúsítvány hitelességét. Ilyen módon kiegészíthetjük, ellenőrizhetjük, helyettesíthetjük a CA-k információját. Ennek érdekében új DNS rekordot vezettek be, a **TLSA** rekordot [18]. A TLSA rekord rugalmasan használható. Tartalma lehet:

- „Ez az én CA-m!"
- „Ez az én tanúsítványom, de ellenőrizd a szokásos módon is!"
- „Saját CA-t használok, ez az!"
- „Ez az én tanúsítványom!"

Egyre több programban van DANE implementáció: nem csak böngészők, hanem SMTP szerverek és SIP szerverek is támogatják.

7. DNSSEC hátrányok

A DNSSEC bevezetése előtt a DNS egyik fő szépsége az egyszerűség volt. Egyszerű volt a protokoll, egyszerű a konfigurálás, nem sok dolga volt se az eszközöknek, se az üzemeltetőknek.

A DNSSEC esetében mindenhol több erőforrásra van szükség:

- Nagyobbak lettek a zónák
- Nehezen olvashatók a rekordok
- Több, hosszabb rekord, nagyobb hálózati forgalom keletkezik
- Több processzoridőre van szükség (kripto műveletek)
- Nagyobb hozzáértést igényel
- Bonyolultabb a konfiguráció
- Könnyebb elrontani
- A hibák erőteljesebben hatnak

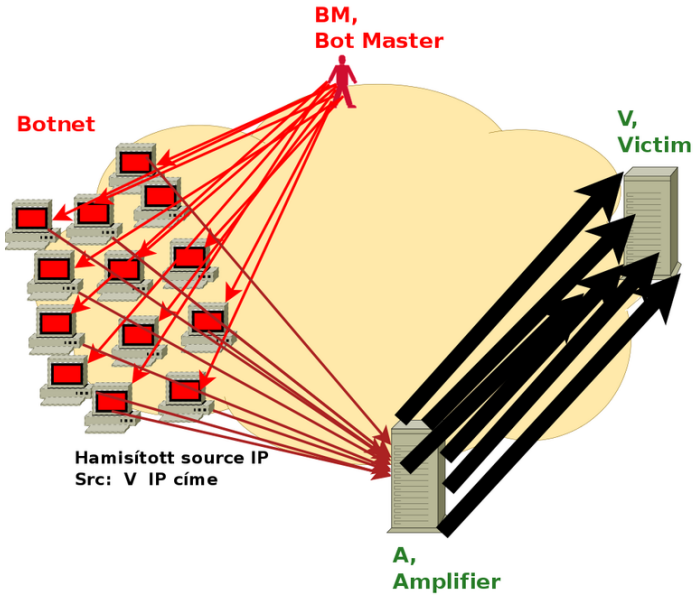
Egy másik hátrány, hogy Amplification (erősítést felhasználó) támadásra adnak módot. Ilyen támadást szemléltet a 9. ábra.

Egy ilyen támadással teljesen meg lehet bénítani a támadott gépet, sőt annak környezetében a hálózatot. A támadás egyik tulajdonsága, hogy hamisított IP címet, a támadott gép IP címét használja. Jellemzően UDP protokollt használó szolgáltatáson alapul (TCP-n nem működőképes). Az ilyen támadásoknál leggyakrabban használt protokollok a NTP, SNMP és a **DNS**. Persze olyan kéréseket kell a támadáshoz használni, ahol a válasz nagyobb mint a kérdés. Ez a DNSSEC-nél különösen igaz.

7.1. Védekezés amplification támadások ellen

Legfontosabb védekezés a hamisított IP címek kiszűrése. Ha ez megvalósulna, minden IP cím hamisításon alapuló támadás lehetetlenné válna. Erre vonatkozik a BCP38 (alias RFC2827), ennek a problémának

[16] `https://www.schneier.com/blog/archives/2011/03/comodo_group_is.html`.
[17] `https://blog.torproject.org/blog/diginotar-debacle-and-what-you-should-do-about-it`.
[18] RFC6698.



9. ábra. Erősítéses támadás (amplification attack)

tárgyalására jött létre a `http://www.bcp38.info` webhely. Hamisított IP címeket a csomagok forrásánál lehet megszűrni. Ezért fontos, hogy minden szolgáltató csak a saját hálózatába tartozó címekről jövő csomagokat engedjen ki a hálózatából. [19] Évek óta minden fórumon szorgalmazzák a BCP38 bevezetését, és nem is hiábavalóan, de nem teljes sikerrel: a BCP38 sok hálózatban nincs bevezetve.

Az autoritatív névszervereken tudunk védekezni a DNS-en alapuló erősítéses támadások ellen *Response Rate Limiting* (RRL) bevezetésével. RRL alkalmazásakor az autoritatív névszerver nem válaszol, ha ugyanazt a **választ** ugyanabba a hálózatba (CIDR) blokkba túl gyakran kellene adni. Hallgatás helyett TCP-re is tereli (alapértelmezésben minden második) ilyen kérdést oly módon, hogy egy rövid választ ad, melyben bebillenti a TC bitet. Az RRL-t az ártatlan felhasználók alig érzékelik, de a DNS erősítést meghiusítja. Az RRL-t több autoritatív névszerver program, például a Bind, az NSD, a Knot és a Yadifa is támogatja.

8. Hogyan vezessük be a DNSSEC-et a saját eszközeinken?

A DNSSEC akkor hatékony, ha minden DNS szereplő alkalmazza: a stub rezolver, a rekurzív névszerver és az autoritatív névszerver is.

8.1. Stub rezolver

Első pillanatra a stub rezolver oldalán látszik a legegyszerűbbnek a DNSSEC bevezetése: adjunk meg DNSSEC-et tudó rekurzív névszervert a rezolver konfigurációban (pl. `/etc/resolv.conf`), és készen is vagyunk. A gyanakvó felhasználó azonban nem lehet teljesen nyugodt: a rekurzív névszerver és a stub rezolver között maradt egy támadási felület. Ezt szokták a „last mile” [20] problémájának nevezni.

Erre megoldás lehet, ha a **saját gépünkön** futtatunk rekurzív névszervert, és `127.0.0.1`-et adunk meg névszerverként. Ennek is vannak azonban hátrányai:

- A cache kevéssé lesz hatékony

[19] Ez a Reverse Path Forwarding (RPF) szűrés.
[20] Utolsó mérföld.

- Az autoritatív névszerverekre nagyobb teher hárul, ha ez széles körben elterjed

Mindkét bajon sokat javít, ha `forwarder`-ként használjuk a régi névszervert. A gyakorlat azt mutatja, hogy nem érezhető lassulás a nevek feloldásában. Lokális gépen rekurzív névszervert - pl. Unbound-ot - telepíteni, üzemeltetni nem is nehéz akár unix, akár windows operációs rendszer alatt.

Egy másik megoldás lehet az utolsó szakaszon `TSIG`-et használni.

Nem egy weblap áll rendelkezésünkre, ha ellenőrizni akarjuk, hogy használunk-e DNSSEC-et. Például:

- `http://dnssec-or-not.org`
- `http://dnssectest.sidn.nl/test.php`

Ezeken a web helyeken más és más tartalom jelenik meg attól függően, hogy éppen használunk-e DNSSEC-et.

8.2. Rekurzív névszerver

Rekurzív névszerverben néhány paraméter beállítása és a root-hoz tartozó trust anchor megadása jelenti a DNSSEC bevezetését. Két példát mutatunk be:

8.2.1. Bind

```
options {
    dnssec-enable yes;
    dnssec-validation yes;
};
```

Ennek a beállításnak a hatására a DNS kérésekben állni fog a `DO` (Dnssec OK) bit, és a rekurzív névszerver DNSSEC szerint ellenőrizni fogja a kapott válaszokat.

```
trusted-keys {
 "." 257 3 8
   "AwEAAagAIK ...
};
```

Ezzel álllítjuk be a gyökér névszerverhez tartozó trust anchort statikus módon.

Ha RFC5011 szerinti rollovert akarunk megengedni, akkor pedig így:

```
managed-keys {
 "." 257 3 8
   "AwEAAagAIK ...
};
```

8.2.2. Unbound

Az Unbound alapértelmezésben DNSSEC támogatással bír, ezért csak a trust anchort kell megadnunk a DNSSEC működéshez:

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

Vagy, ha RFC5011 szerinti rollovert akarunk megengedni:

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

Ellenőrizhetjük, hogy egy rekurzív névszerver használ-e DNSSEC-et.

1. módszer. Ha igen, `SERVFAIL` a válasz a következő kérdésre:
   `%dig www.dnssec-failed.org @149.20.64.20`
2. módszer. Ha igen, áll a következő kérdésre adott válaszban az `ad` bit:
   `%dig www.ripe.net @149.20.64.20`

8.3. Autoritatív névszerver

Autoritatív névszervernél két feladat áll előttünk: egyrészt létre kell hoznunk, és folyamatosan karban kell tartanunk a DNSSEC-cel védett zónát, másrészt szolgáltatni kell azt. A mai névszerver programok többsége lehetőséget ad az első feladat megoldására is, de azok helyett érdemes megfontolni az Opendnssec használatát. A második feladat megoldása nem nehéz. Az NSD és a Bind is alapértelmezésben támogatja a DNSSEC-et: értik a `DO` bitet, a kérdésekben és e bit hatására a DNSSEC rekordokat is szolgáltatni fogják.

8.3.1. NSEC vagy NSEC3 ?

Amikor egy zónát DNSSEC-cel konfigurálunk, el kell döntenünk, hogy NSEC, vagy NSEC3 rekordokkal akarjuk-e a nem-létezést kezelni. Az NSEC3-nak hátránya, hogy nagyobb terhet ró a rekurzív oldalon a névszerverre, és nagyobb terhet ró az autoritatív névszerverekre is: mindig hash-t kell számolni, összehasonlítani. Az NSEC3-at nehezebb áttekinteni.

Mindezekért NSEC3-at csak akkor érdemes használni, ha:

- Jellemzően delegálások vannak a zónában (pl. TLD-k esetén), vagy
- Sok domain név, nehezen kitalálható, kritikus funkciójú eszköznevek vannak a zónában

Az egy-két nevet tartalmazó zónákban NSEC-et érdemes használni. A zónák túlnyomót többsége ilyen: a feltétlenül szükséges SOA és NS rekordokon kívül egy-két rekordot tartalmaz. Például két A rekordot (`www.valami.hu, mail.valami.hu`) és esetleg egy MX rekordot. Ilyen esetekben semmi előny nem származik az NSEC3-ból, csak hátrány.

8.4. DNSSEC a .hu alatt

A `.hu` alatt 2011. óta kísérleti rendszer működött. 2014. októberében élesítettük a DNSSEC konfigurációt, a `.hu` zóna alá van írva. 2015-ben bevezetjük a DNSSEC-cel védett delegálás lehetőségét. Ennek módja, hogy a regisztrátor a regisztrátori felületen a DNSSEC checkbox kiválasztásával kéri a domain DNSSEC delegálását. Ezek után a domain ellenőrzés során a `regcheck` procedúra:

- Az autoritatív DNS szerverből kiolvassa a `DNSKEY` rekordo(ka)t
- Ellenőriz
- Ha mindent rendben talál, képezi a `DS` rekordot/rekordokat

A `.hu`-ba nem csak a felolvasott `NS` rekordokat, hanem a `DS` rekordo(ka)t is beteszi. Ettől kezdve a delegált zóna DNSSEC-cel védett.

9. OpenDNSSEC, OpenDNSSEC konfigurálás

9.1. Mi az OpenDNSSEC?

Az OpenDNSSEC egy könnyen kezelhető, konfigurálható DNSSEC megoldás. Szabad szoftver, folyamatosan fejlesztik. A fejlesztésében részt vesz több TLD: `.se, .uk, .nl`, és még több TLD használja, például a `.hu` is. Az OpenDNSSEC **nem** tartalmaz névszervert, csak a DNSSEC zóna előállítása a cél. Az OpenDNSSEC bármilyen névszerverrel együtt tud működni, akár váltogathatjuk a név szerver programokat anélkül, hogy az OpenDNSSEC konfiguráción bármit változtatnánk.

Az OpenDNSSEC a kulcskezelést `PKCS#11` interfész-en végzi. A PKCS11 az RSA Labs-tól származó szabvány, melyet elsősorban különböző HSM-ek (Hardware Security Module)-ok támogatnak. Az OpenDNSSEC csapat szoftveres HSM-et fejlesztett, a `softhsm`-et.

9.2. Policy driven: KASP = Key And Signing Policy

Az OpenDNSSEC legkellemesebb tulajdonsága, hogy a felhasználó csak az **eljárásrendet**, a *policy*-t határozza meg, ezek után az OpenDNSSEC-re bízza a következő feladatokat:

- Kulcsok generálása
- Kulcsok használatbavétele, törlése
  - A kulcsok ugyan nem járnak le, de rendelkezünk lejáratról!
- Aláírások generálása, újragenerálása
- Zóna publikálás
- Naplózás

Nem megy automatikusan a `KSK` → `DS` → `DS` publikálás. Erre — pillanatnyilag — nincs automatizmus az OpenDNSSEC-ben. (Kétséges, hogy kívánatos lenne-e). Minden esetre az OpenDNSSEC ebben is segít: generálja a kulcsot amikor a policy megkívánja, és figyelmeztet, hogy juttassuk el az apuka zónába az új `DS` rekordot. Ha a `DS` rekord bekerült az apuka zónába, akkor egy paranccsal tudomására hozhatjuk, hogy az apuka átvette a `DS`-t:

```
ods-ksmutil key ds-seen -x  32701 --zone bioetika.hu
```

9.3. OpenDNSSEC építőelemek

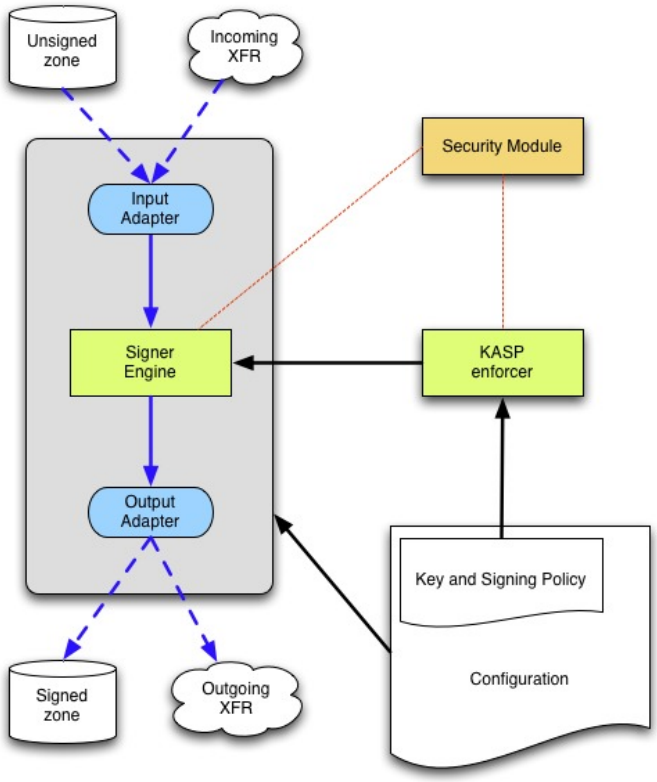
A 9. ábrán láthatjuk az OpenDNSSEC építőelemeit.

Az ábrán látható, hogy a KASP meghatározásán kívül lényegében csak arra van szükség, hogy megmondjuk honnan vegye az aláirandó nyers zónát és hova tegye az aláírt zónát. A zóna input és output is lehet egyszerű fájl, vagy pedig egy DNS zóna transzfer.

9.4. KASP

Az OpenDNSSEC működés szívében áll a *Key and Signing Policy*, a KASP. Több eljárásrendet (policy-t) tudunk definiálni, és egy policy meghatározásakor nem kell megmondani, hogy az melyik zónához, vagy zónákhoz tartozik. A kezelt zónák listáját külön kell megadnunk, és a zónákhoz rendelünk aztán policy-t. Több policy-t is definiálhatunk, de használhatjuk ugyanazt a policyt sok ezer zónával. A policy-ban eldöndentő kérdések:

- Milyen kulcsokat használjunk?
- Mennyi időnként cseréljük?
- NSEC vs. NSEC3?



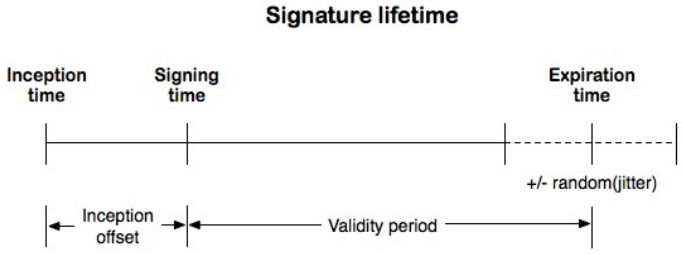
10. ábra. OpenDNSSEC építőelemek (Forrás: OpenDNSSEC wiki)

Többször kell időt specifikálnunk ami az ISO 8601 szabvány szerint történik. Ez a szabvány időpont, ismétlődés, intervallum szabványos megadását írja le. Az OpenDNSSEC-ben csak az **időtartam** definiálása érdekes. Ez ilyen formában történik:

- `P[n]Y[n]M[n]DT[n]H[n]M[n]S`
- `P` jelenti, hogy időtartam-ot adunk meg (**P** eriódus)
- `T` választja el az órát a dátumtól

Vegyük észre, hogy `M` mást jelent `T` előtt mint `T` után. Jó tudni (de erre mindig figyelmeztet is), hogy az OpenDNSSEC-nél 1 év **mindig** 365 nap, 1 hónap **mindig** 31 nap.

A 10. ábrán láthatjuk az OpenDNSSEC aláírási időparamétereket.



11. ábra. Aláírási paraméterek (Forrás: OpenDNSSEC wiki)

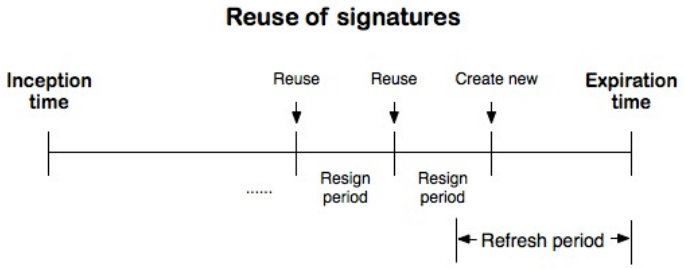
Az *inception offset* mutatja, hogy mennyire „datálja vissza” a számítógép órájához képest az aláírások kezdetének idejét, a *signature inception* időt. Erre azért van szükség, mert ha az aláírás után nem sokkal valaki ellenőrzi azt, és az órája jelentősen hátrább jár mint a mienk, akkor azt találhatja, hogy a jövőben

keletkezett az aláírás, és ezért nem fogadja el. Ez ellen véd ez a paraméter.

A *jitter* azt mondja meg, hogy egy bizonyos időszakra szóló aláírások időtartamai mennyire szóródjanak véletlenszerűen. Ha ugyanis egy időpontban sok azonos időtartamra szóló aláírás keletkezik, akkor azok megújítása, a rekordok ismételt aláírása egyszerre válik esedékessé. A *jitter* paraméter segítségével elnyújthatjuk ezt az újra aláírást, ilyen módon takarékoskodhatunk az erőforrásokkal.

9.5. Aláírások generálása

A 11. ábra mutatja az aláírások újrafelhasználásával kapcsolatos paraméterek jelentését.



12. ábra. Újra felhasznált aláírások (Forrás: OpenDNSSEC wiki)

Az OpenDNSSEC *Resign period* időnként újra, és újra megnézi, hogy szükséges-e ismét aláírni egy-egy rekordot. Ha az aláírás lejárati ideje, az *Expiration time* még messze van, akkor változatlanul hagyja az aláírást. Nem várja meg azonban a lejárati időt, hanem akkor írja újra a rekordot, ha a pillantanyi idő már *Refresh period*-nál jobban megközelítette a lejárati időt.

9.6. OpenDNSSEC kulcs állapotok

Az OpenDNSSEC az egyes zónákhoz a policy-ban meghatározott módon kulcsokat generál. A kulcsok a következő állapotokban lehetnek:

- GENERATE
  - A kulcsot éppen legenerálta — nem szoktuk látni
- PUBLISH
  - A zónában ott a kulcs, de még nem terjedt el, még nem biztonságos vele aláírni
- READY
  - Már elkezdhetnénk használni a kulcsot
  - KSK esetén ilyenkor még a `ds-seen` paranccsal kell bíztatni, hogy ténylegesen használatba jusson
- ACTIVE
  - Az ilyen kulcsot használja aláírásra az OpenDNSSEC
- RETIRE
  - A kulcsot már nem használjuk aláírásra, de a világban még lehetnek ezzel a kulccsal aláírt rekordok
- DEAD
  - A kulcsot ki lehet venni a zónából

9.7. OpenDnssec telepítés, használat

Az OpenDnssec rendelkezésre áll például a Debian disztribúciókban. Az `opendnssec` és a `softhsm` csomagok (és függőségeik) telepítése után néhány konfigurációs fájlt kell csak editálnunk és készen is vagyunk. A legfontosabb, hogy a konfigurálás **előtt** határozzuk meg, hogy milyen eljárásrend (policy) szerint akarjuk működtedni a DNSSEC zónáinkat. Ajánlatos a generált zónákat ellenőrizni mielőtt névszerverek szolgáltatnák. Erre igen hasznos eszköz a *validns*, amiből szintén van Debian csomag.

Az OpenDnssec nagy előnye, hogy ha egyszer végiggondoltuk, bekonfiguráltuk, akkor attól kezdve — éppen úgy, mint a DNSSEC nélküli DNS esetén — akár el is feledkezhetünk az egészről: a zónánk ugyan időről időre változni fog, új kulcsok, aláírások keletkeznek, de mindez megbízhatóan és automatikusan.

9.7.1. DNSSEC bevezetésének lépései

Ha egy zónában be akarunk vezetni DNSSEC-et, akkor a következő lépéseket kell megtenni:

1. Telepíteni, konfigurálni a megfelelő szoftvereket (pl. OpenDnssec)
2. Ellenőrizni, tesztelni.
   (a) Érdemes kipróbálni rollover-eket.
   (b) Érdemes a monitorozó rendszerünket kiegészíteni DNSSEC monitorozással.
3. Módosítani a zóna delegálást, hogy a DS rekord(ok) bekerüljön/bekerüljenek az apuka zónába.
   (a) A `.hu` esetén ez a regisztrátori felületen a DNSSEC checkbox bepipálását jelenti.

A 3. lépés előtt még „büntetlenül” el lehet rontani a DNSSEC konfigurációt: addig egy DNSSEC hiba a világ számára még nem akadályozza meg a nevek feloldását, hiszen a neveink még a világ számára *insecure* állapotban vannak: nincs bizalmi kapcsolat a mi DNSSEC konfigurációnk és a gyökér DNSSEC konfiguráció között.

A 3. lépés előtt validáló rekurzív névszerverünkbe bevezethetünk egy új trust anchort, ami a mi zónánk DNSKEY rekordját tartalmazza. Így a mi rekurzív név szerverünk már DNSSEC szerint fogja kezelni a neveinket. Ilyen módon egy átmeneti intenzív teszt időszakra is lehetőségünk nyílik.

10. Olvasnivaló

- `http://www.dnssec.net/`
- `http://www.root-dnssec.org/`
- `https://www.dnssec-deployment.org/`
- `http://dnssec-debugger.verisignlabs.com/`
- `http://dnsviz.net`
- `http://www.nlnetlabs.nl/publications/dnssec_howto/`
- `http://www.afnic.fr/medias/documents/DNSSEC/afnic-dnssec-howto-en-v2.pdf`
- `http://labs.apnic.net/blabs/?p=316`
- `http://www.bortzmeyer.org/files/article-satin2011.pdf`
- `http://dns.comcast.net/index.php/help`