SSL-Absicherung mittels DANE

und

Sicheres DNSSEC mit Bind 9.x

- **Heinlein Support**
  - IT-Consulting und 24/7 Linux-Support mit ~25 Mitarbeitern
  - Eigener Betrieb eines ISPs seit 1992
  - Täglich tiefe Einblicke in die Herzen der IT aller Unternehmensgrößen

- 24/7-Notfall-Hotline: 030 / 40 50 5 - 110
  - 25 Spezialisten mit LPIC-2 und LPIC-3
  - Für alles rund um Linux & Server & DMZ
  - Akutes: Downtimes, Performanceprobleme, Hackereinbrüche, Datenverlust
  - Strategisches: Revision, Planung, Beratung, Konfigurationshilfe

heinlein

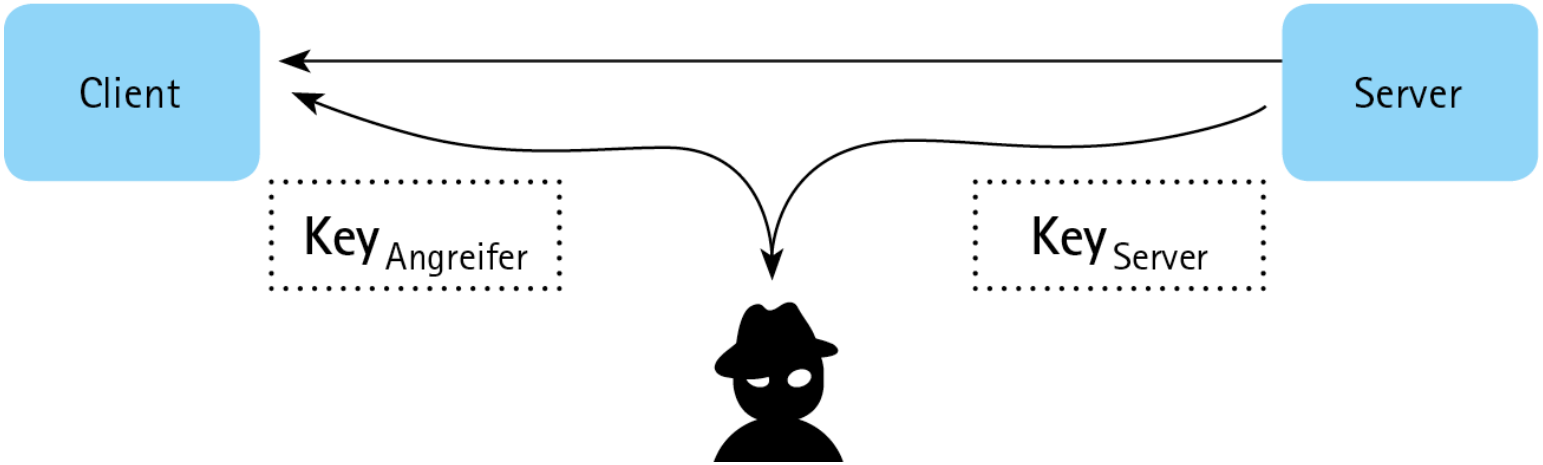
Johannes Rundfeldt <j.rundfeldt@heinlein-support.de>

Warum SSL/TLS alleine nicht sicher ist

Johannes Rundfeldt <j.rundfeldt@heinlein-support.de>

Warum SSL/TLS nicht ausreicht

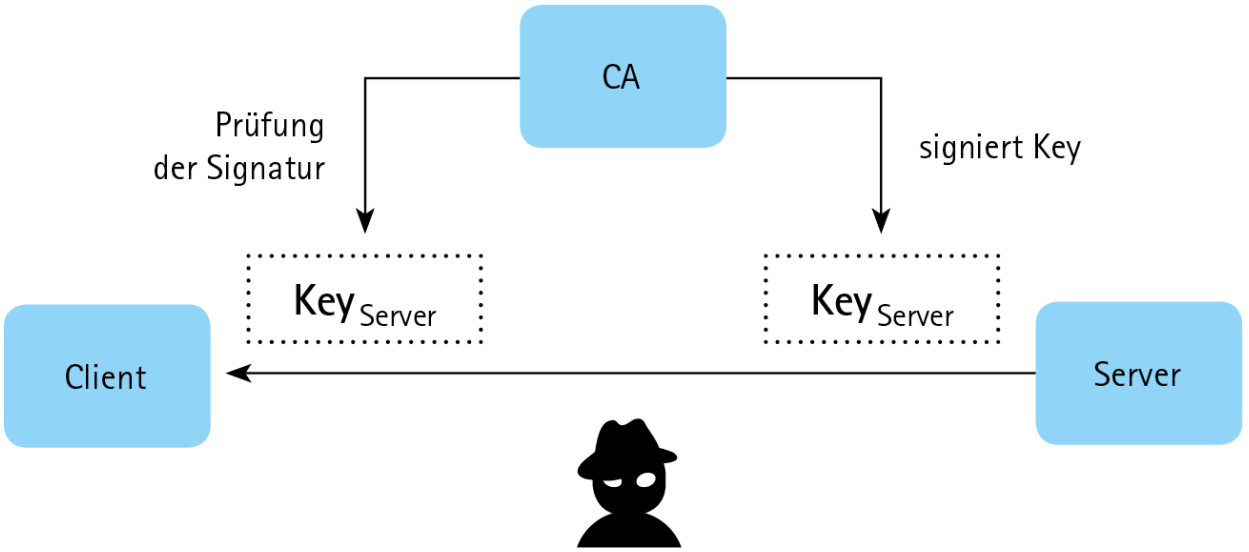
→ Ein man-in-the-middle könnte das SSL-Zertifikat austauschen.



Johannes Rundfeldt <j.rundfeldt@heinlein-support.de>

Warum SSL/TLS nicht ausreicht

- Darum braucht man eine CA.



- Welcher CA kann man trauen? Kann man allen 200 CAs trauen?

Johannes Rundfeldt <j.rundfeldt@heinlein-support.de>

Warum SSL/TLS nicht ausreicht

- Mit welchen Servern rede ich?
  - Über DNS-Poisening könnten andere MX-Records untergeschoben werden.
  - Kann ich mich ohne DNSsec auf das DNS verlassen?
- Viele Provider haben nur selbstsignierte Zertifikate.
  - Kann man ändern, muß man dann aber auch!
- Was ist, wenn kein SSL/TLS möglich ist? Was ist, wenn der MitM das SSL-Announcement unterdrückt?
  - Dann normalerweise Fallback auf Klartextübertragung
- SSL/TLS ist nur Nexthop-Verschlüsselung. Danach liegt die Mail zunächst wieder im Klartext vor
  - Kann man seinem Provider trauen?

Johannes Rundfeldt <j.rundfeldt@heinlein-support.de>

Warum SSL/TLS trotzdem sinnvoll und gut ist

- Es schützt vor ungewollten Mitlesern
- Es erhöht den Aufwand für jemanden, der mitlesen will
- Es ist eine selbstverständliche Grundabsicherung
- Das Problem: Es ist halt immer nur optional. Ein MitM kann SSL unterdrücken.
- Rund 80% unserer SMTP-Verbindungen laufen über SSL/TLS
  - Ein deutlicher Anstieg in letzten Jahr.

Johannes Rundfeldt <j.rundfeldt@heinlein-support.de>

So funktioniert DANE

- DANE TLS führt einen neuen TLSA-Record ein (TLS Association)
- Der definiert über Hashwerte, welche Zertifikate der Client akzeptieren darf.
  - Die Zertifikate einer bestimmten CA
  - Bestimmte genannte Zertifikate
  - Zertifikate, die von einem bestimmten Vertrauensanker abstammen
- Der TLSA-Record wird dann für _25._tcp.mx1.example.com definiert
  - Also individuell pro Dienst/Port und Hostname
  - Port kann auch Wildcard sein *._tcp_example.com
    - Ein Mailserver hat viele Ports: 25, 465, 587, 110, 143,993, 995, 2000, 4190

Johannes Rundfeldt <j.rundfeldt@heinlein-support.de>

Technische Spezifikation des TLSA-Records

```
_25._tcp.mx1.example.com. IN TLSA 3 1 1 5c1502a6549c423b
e0a0aa9d9a16904de5ef0f5c98c735fcca79f09230aa7141
```

- Feld 1: Certification Usage (hier: 3)
  - 0 = PKIX-TA: CA-Zertifikat, das in der Validierungskette auftauchen muss (?)
  - 1 = PKIX-EE: CA-Root-Zertifikat, gegen das die CA-Kette validiert (?)
  - 2= DANE-TA: CA-Zertifikat mit dem der Key unterschrieben sein muß
  - 3= DANE-EE: Konkreter exakter Public Key des Servers (self signed!)
- Feld 2: Selector Field
  - 0 = Certificate Binary Structure 1 = DER-encoded Binary Structure
- Feld 3: Machting Tye (hier: 1)
  - 0 = Ganzes Zertifikat
  - 1 = SHA-256 Hash des Zertifikats 2 = SHA-512-Hash des Zertifikats
- Feld 4: Zertifikatswert (hier: 5c1502a...)
- Default: „TLSA 3 1 1“

Johannes Rundfeldt <j.rundfeldt@heinlein-support.de>

Probleme von DANE TLS

- DNS selbst wäre durch einen MitM leicht angreifbar
  - Die TLSA-Records könnten unterdrückt und ausgetauscht werden
  - Anschließend wäre der Client wieder blind und naiv wie früher
- Also: Absicherung der DNS-Records über DNSsec
  - Theoretisch: Kein Problem. „Muß man nur machen“
  - Praktisch: DNSsec hat sich bis heute nicht fläckendeckend durchgesetzt. Zahlreiche Fallstricke, komplexeres Handling der Records, große Sorgfalt notwendig.
  - DANE TLS sorgt im Prinzip für den ersten flächendeckenden Einsatz von DNSsec!
- Problem: Hohe Latenz bei Überprüfung der zahlreichen DNSsec-RR
  - Schwierig bei HTTP, XMPP & Co!

heinlein

Johannes Rundfeldt <j.rundfeldt@heinlein-support.de>

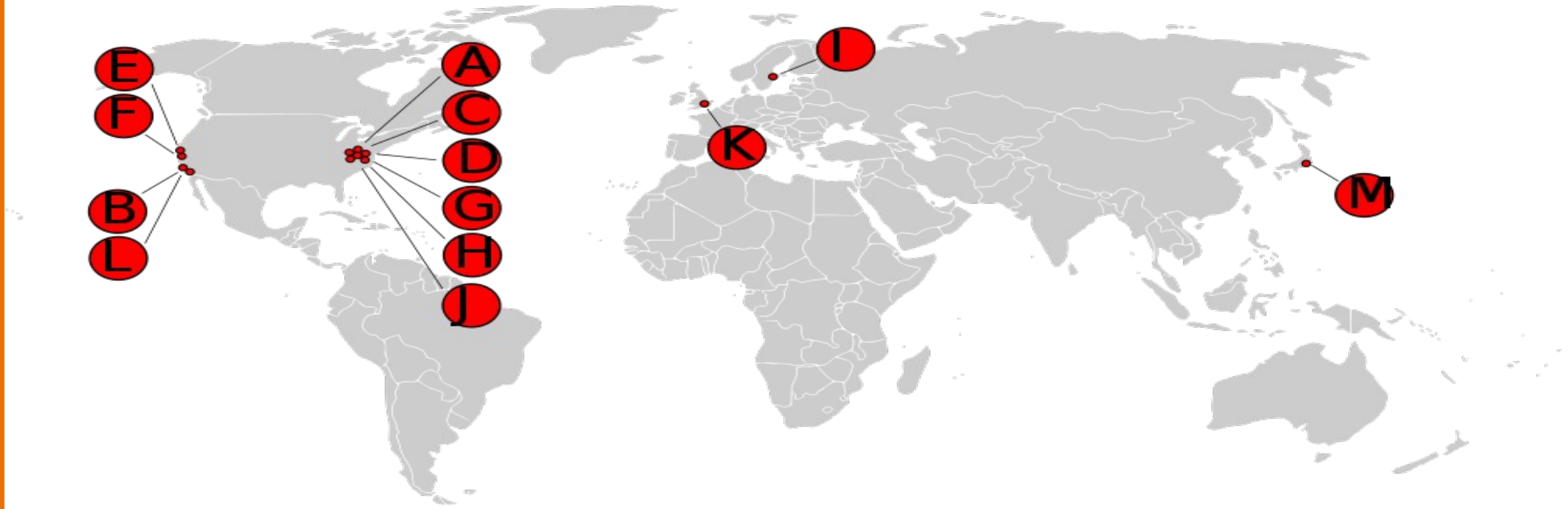
Die Mutter aller Dienste:

Das DNS

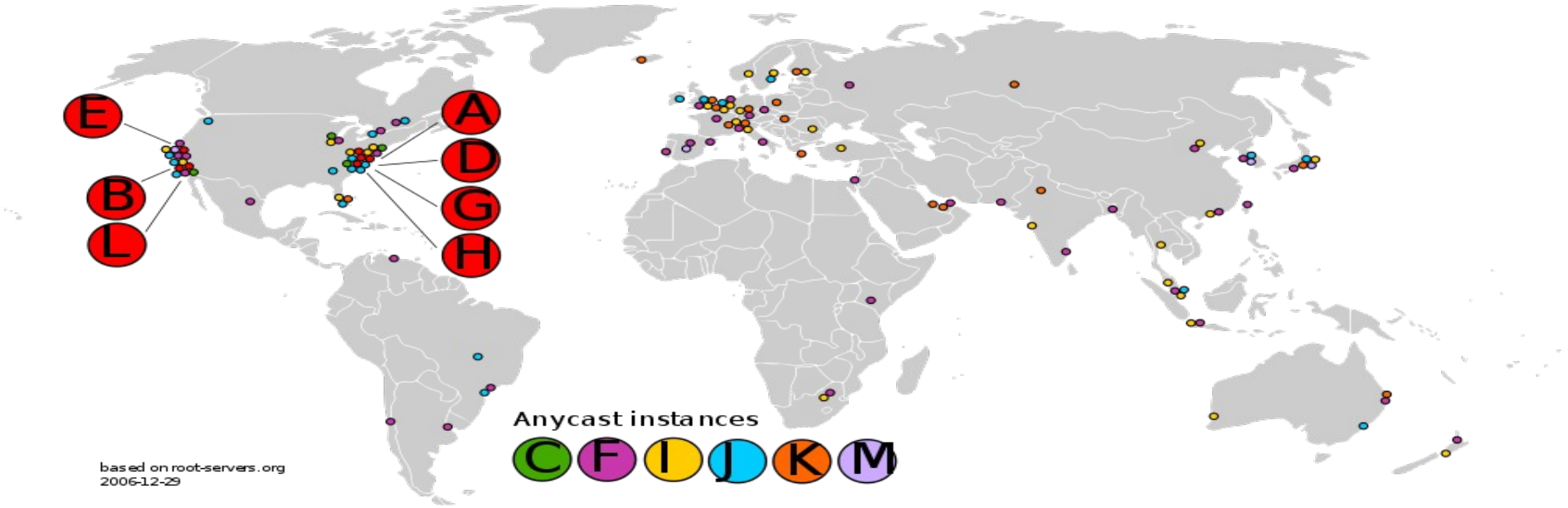
- DNS: Erstmalig 1983 in RFC 882/883 entwickelt
  - Reservierter DNS-Port: 53
  - Kommunikation Client – Server über Ports 1024 an 53
  - Kommunikation Server – Server i.d.R. Über Port 53 an 53!
- I.d.R. per UDP/IP
  - Schön schnell
  - Einfaches Frage/Antwort-Prinzip
  - Kurze Fragen, kurze Antworten
- Manchmal aber auch per TCP/IP
  - Bei DNS-Zonentransfers wegen langer Dateien und zeitunkritischem Transfer
  - Manchmal aber auch nötig bei langen DNS-Records > 512 Bit
  - DNS-Query über TCP manchmal hakelig

- Die DNS-Root-Server verwalten die ca. 3000 TLD-Zonen
  - Root-Server-Datei muß als Startpunkt den DNS-Servern bekannt sein
- Es gibt 13 Root-Server (A-M)
  - Verteilung per Anycast auf 123 reelle Systeme
- Verwaltet wird das durch die ICANN
  - ...untersteht dem US-Handelsministerium...
- Betrieben wird die Root-Zone durch Verisign
  - ...ebenfalls unter US-Recht...

- Anfang 2000: 10 von 13 Servern in den USA, davon 6 Server auf kleinstem Raum an der US-Ostküste



→ Seit 2006: 123 Server verteilt per Anycast über die Welt



DNS – latent unsicher

- Wie bei jedem Protokoll sind Man-in-the-middle-Angriffe denkbar
- Das UDP-Protokoll bietet zahlreiche Angriffsmöglichkeiten
  - Records könnten injiziert werden
  - Records könnten unterdrückt werden
  - Records könnten verändert werden
- DNS ist von A bis Z nicht vertrauenswürdig

- Auf DNS-Daten basiert aber fast das gesamte Internet
  - Irgendwie erstaunlich, daß es dann doch so gut funktioniert

So funktioniert DNSSEC (von unten nach oben gesehen)

Das ist ein A-Record :-)

```
mailbox.org. IN A  213.203.238.17
```

Damit den keiner fälschen kann, brauchen wir eine digitale Unterschrift:

```
mailbox.org. IN A  213.203.238.17

mailbox.org. IN RRSIG  A 7 3 900
20141209161951 20141109160341 5719
mailbox.org. JqMJ9tzkwU6Yn2unUzlsbr0
kvApVNvHVKzyAUOaRVoZCISWKZRPkeuz7swu
```

Damit wir die digitale Unterschrift prüfen können, brauchen wir einen Public Key.

```
mailbox.org. IN A  213.203.238.17

mailbox.org. IN RRSIG [...]

mailbox.org. IN DNSKEY  256 3 7
BQEAAAABwcvTaaZokGcz2HFSgv+ixKiuypnY
zA3z/pu9MlZ1XFD2qeN7KgVB/mmlvFKDUgKd
raUV2m2KglZLzc4d8GoXTnpLDhcWVJx9et9t
```

Damit der Public Key sicher ist („Trust hat“), publizieren wir ihn in der nächsthöheren Ebene.

```
mailbox.org. IN DS 38499 7 2
574F226E410BFBD86BDE1FD276EC9E88D778
449A65BBDCF1F218E4D1 9F851312
```

Auch die TLD signiert unseren Key per RRSIG:

```
mailbox.org. IN DS 38499 7 2
574F226E410BFBD86BDE1FD276EC9E88D778
449A65BBDCF1F218E4D1 9F851312

mailbox.org. IN RRSIG  DS 7 2 86400
20141208155603 20141117145603 60764
org. b9oWU3saTlNaii7Bp9+odhiwx
```

Damit wir die digitale Unterschrift der TLD prüfen können, hat auch sie einen Public Key:

```
mailbox.org.  IN DS 38499 7 2 [...]

mailbox.org.  IN RRSIG  DS [...]

org.          IN DNSKEY  256 3 7
AwEAAYpYfj3aaRzzkxWQqMdl7YExY81NdYSv
+qayuZDodnZ9IMh0bwMcisdua7ssaiuziuz
```

Die TLD muß ihren Key als DS-Record in „.“ veröffentlichen...

Den Key von „.“ müssen wir kennen / hart importieren.

=> Trust Chain steht.

- Resolver müssen die allerobersten DNSKEY-Records der Root-Nameserver kennen
  - Muß halt ähnlich wie root.hint verteilt werden
  - Hilfsweise:

```
dig +nocomments +nostats +nocmd +noquestion -t DNSKEY . > trusted-key.key
```

- Clients verifizieren nicht, sondern vertrauen ihrem DNS-Resolver, dass dieser „authenticated data“ liefert
  - dig kann jedoch auch selbst validieren WENN entsprechend compiliert

```
dig +topdown +sigchase +multiline +trusted-key=./trusted-key.key -t A heinlein-support.eu
```

Soweit so gut.

Aber nun fangen die Probleme erst an.

- An irgendeiner Stelle („Trust Anchor“) muß man beginnen, jemandem zu glauben.
- Im Idealfall „ganz oben“, also „.“ auf den Root-Servern
  - Hilfsweise irgendwo Quereinstieg und dann von dort aus abwärts („Security Island“)
- Jeder signiert den Einstieg bei seinem Nachfolger
- Man muß seine Keys bei der nächst höheren Ebene hinterlegen
  - Problem: Schlüsseltausch aufwändig/manuell durchführbar
  - Regelmäßiger Schlüsseltausch aber sinnvoll!

Das Problem: Das Publizieren der Keys

- Automatische Verfahren zum Publizieren des Keys in der nächst höheren Ebene gibt es nicht.
- Wir wollen unseren Key aber von Zeit zu Zeit ohne Aufwand tauschen.
- Abhilfe: Wir ziehen eine weitere Ebene ein:

  1) Wir signieren unsere Records mit einem Zone Signing Key (ZSK), den wir häufig (monatlich?) tauschen können

  2) Der kurzlebige ZSK wird vom langlebigen Key Signing Key (KSK) signiert

  3) Erst der KSK wird in der nächst höheren Ebene publiziert

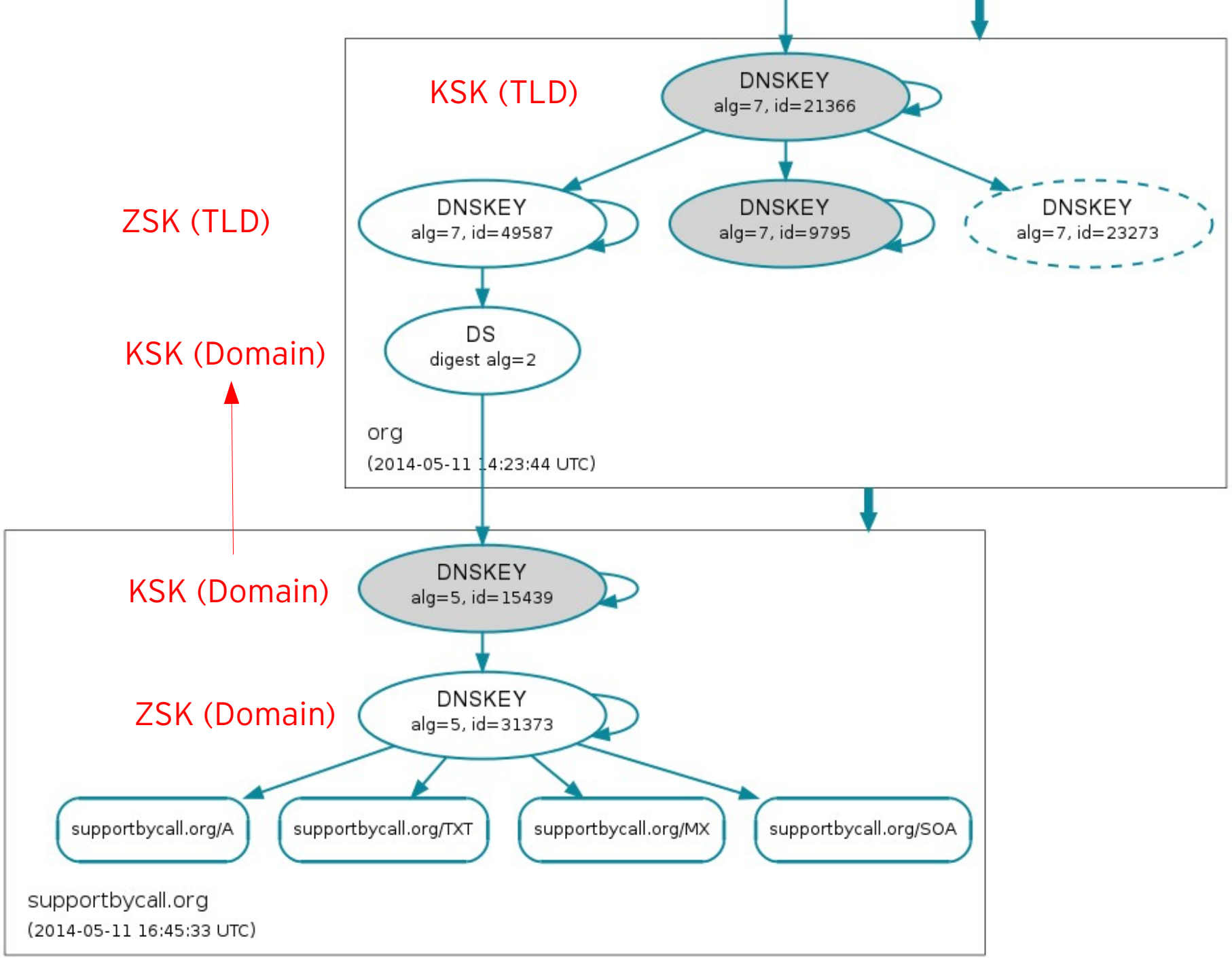
- => Wir haben also jeweils ZWEI Keys (ZSK + KSK).

Es gibt zwei Keys: KSK und ZSK

- Darum: DNSSEC kennt zwei verschiedene Key-Typen!

- ZSK (256): Zone-Signing Keys, unterschreibt Records in Zonen
  - Häufiger Tausch sinnvoll
  - Wird lokal vom KSK unterschrieben, vergleichsweise einfach automatisierbar
  - Empfehlung: Monatlich tauschen

- KSK (257): Key-Signing Keys, unterschreibt andere (ZSK-) Keys
  - Tausch aufwändig weil Upstream zu veröffentlichen
  - Wird nur genutzt um Unterschlüssel zu unterschreiben
  - Empfehlung: 1 x pro Jahr tauschen


KSK (TLD)
DNSKEY
alg=7, id=21366
ZSK (TLD)
DNSKEY
alg=7, id=49587
DNSKEY
alg=7, id=9795
DNSKEY
alg=7, id=23273
KSK (Domain)
DS
digest alg=2
org
(2014-05-11 14:23:44 UTC)
KSK (Domain)
DNSKEY
alg=5, id=15439
ZSK (Domain)
DNSKEY
alg=5, id=31373
supportbycall.org/A
supportbycall.org/TXT
supportbycall.org/MX
supportbycall.org/SOA
supportbycall.org
(2014-05-11 16:45:33 UTC)

Howto:
Einrichtung von DNSSEC mit Bind 9.x

Schlüsselerzeugung

- Erzeuge die zwei Schlüssel (KSK und ZSK)

```
/var/lib/bind/keys # dnssec-keygen -f KSK -n ZONE -a RSASHA1 -b 1024
supportbycall.org
Generating key pair.........................++++++ .....++++++
Ksupportbycall.org.+005+46864

/var/lib/bind/keys # dnssec-keygen -e -n ZONE -a RSASHA1 -b 1024
 supportbycall.org
Generating key pair.........................++++++ ........................++++++
Ksupportbycall.org.+005+00158

/var/lib/bind/master.dnssec/RSASHA1 # ls -la
-rw-r--r--  1 root  root   449 Mar 15 20:42 Ksupportbycall.org.+005+00158.key
-rw-------  1 root  root  1014 Mar 15 20:42 Ksupportbycall.org.+005+00158.private
-rw-r--r--  1 root  root   446 Mar 15 20:42 Ksupportbycall.org.+005+46864.key
-rw-------  1 root  root  1010 Mar 15 20:42 Ksupportbycall.org.+005+46864.private
```

Anpassung in Bind

- KSK und ZSK-Key im Key-Dir erzeugen
- BIND-Schreibrechte auf Directory/Files (Dynamische Updates)

Vorgehensweise aussuchen:

- Dynamische Zonen (Bind 9.7)
  - Verwaltung als dynamische Zone per nsupdate
  - Bind signiert beim Zone-Reload
- Inline Signing (Bind 9.9)
  - Verwaltung ganz normal in ASCII-Dateien
  - Bind signiert beim Zone-Reload

Bind 9.7: Dynamische Zonen-Updates

- Zone dynamische verwaltbar anlegen:

```
options {
     dnssec-enable yes;
     dnssec-validation yes;
     dnssec-lookaside auto;
};
zone "supportbycall.org." {
     type master;
     auto-dnssec maintain;
     update-policy local;
     file "/var/cache/bind/master.dnssec/supportbycall.org";
     key-directory "/var/lib/bind/keys";
};
```

Bind 9.7: Dynamische Updates per nsupdate

- Damit Bind alle Zonen automatisch signieren kann, sollten diese dynamisch per nsupdate gepflegt werden

```
root@ns:~# dig supportbycall.org SOA
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 3, ADDITIONAL: 5
supportbycall.org.      3600    IN      SOA     ns.jpberlin.de. root.jpberlin.de.
2014021323 40000 7200 604800 86401

root@ns:~# nsupdate  -l
> update add slac.supportbycall.org 3600 A 192.168.10.10
> send
> quit

root@ns:~# dig supportbycall.org SOA
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 3, ADDITIONAL: 5
supportbycall.org.      3600    IN      SOA     ns.jpberlin.de. root.jpberlin.de.
2014021324 40000 7200 604800 86401
```

Bind 9.9: Inline-Signing

- Zone als inline-signing markieren:

```
options {
     dnssec-enable yes;
     dnssec-validation yes;
     dnssec-lookaside auto;
};
zone "supportbycall.org." {
     type master;
     auto-dnssec maintain;
     inline-signing yes;
     file "/var/cache/bind/master.dnssec/supportbycall.org";
     key-directory "/var/lib/bind/keys";
};
```

- Zone in ASCII editieren, reloaden, fertig!

Die unsignierte Zone im Überblick

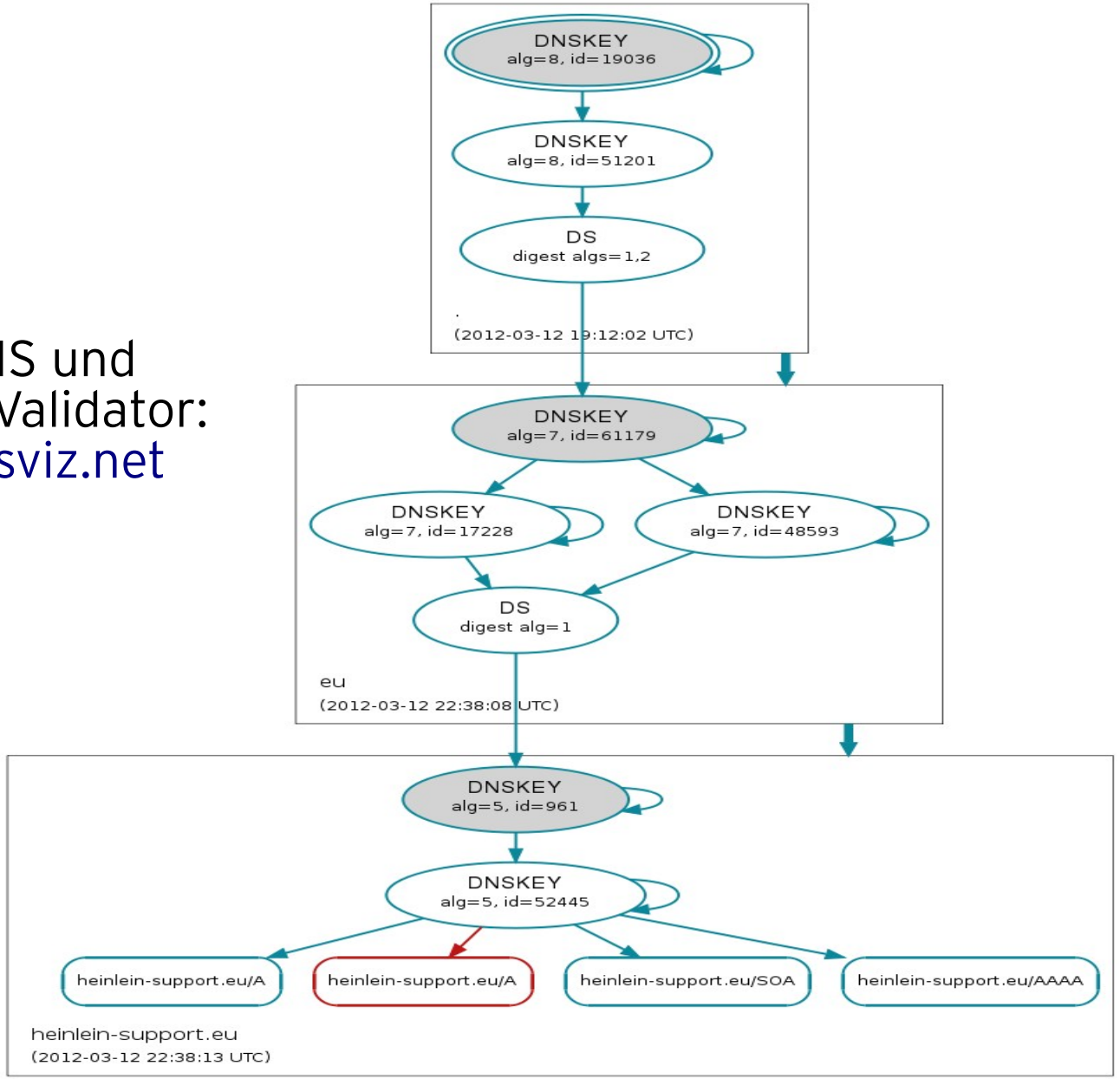
```
supportbycall.org   IN SOA  ns.jpberlin.de. root.jpberlin.de. (
                    2014021331 ; serial
                    [...]
            NS      ns.jpberlin.de.
            NS      ns2.jpberlin.de.
bla         A       192.168.5.5
```

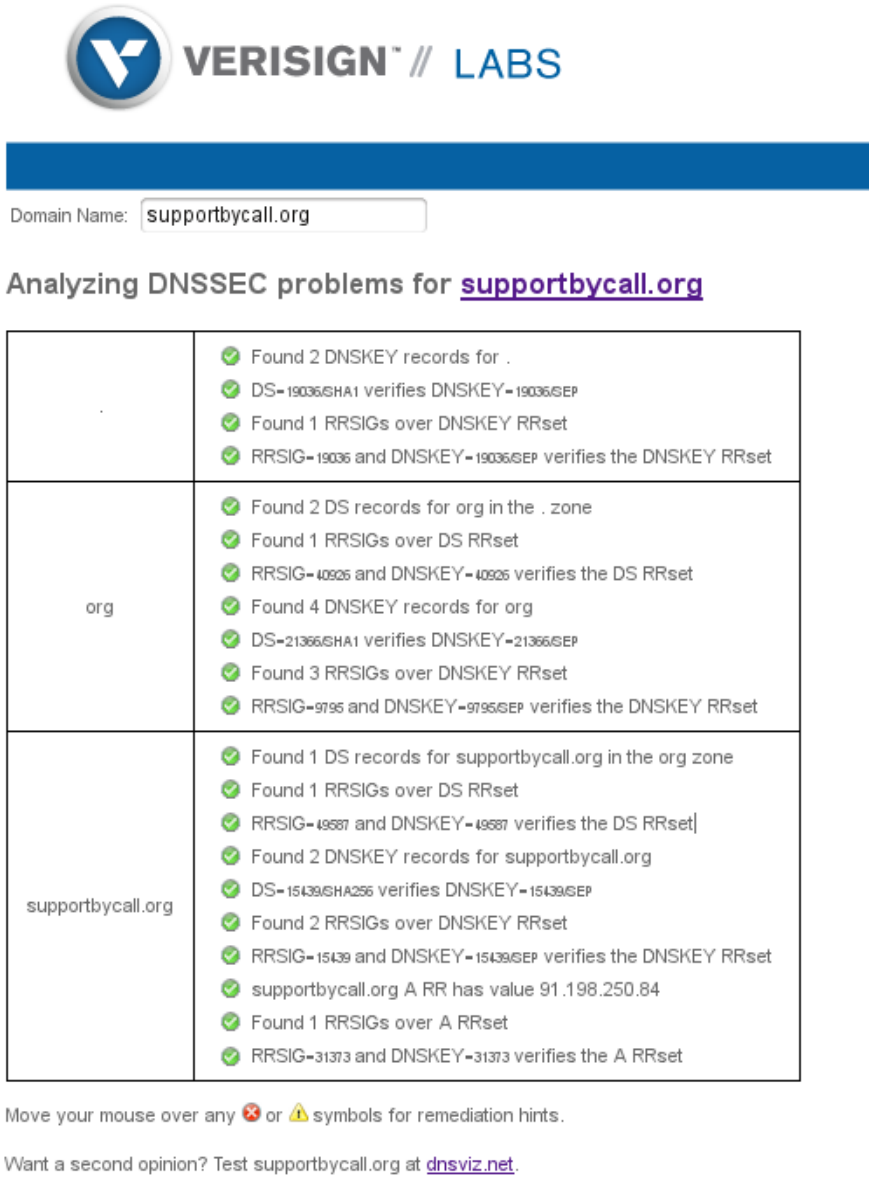
Die signierte Zone im Überblick

```
supportbycall.org.   IN SOA  ns.jpberlin.de. root.jpberlin.de. (
                     2014021331 ; serial
                     [...]
             NS      ns.jpberlin.de.
             NS      ns2.jpberlin.de.
             RRSIG   NS 5 2 3600 20140611224239 (
                     20140512224001 31373 supportbycall.org. RCvHznuQVC2KWtk+RZ
             DNSKEY  256 3 5 (
                     F6am+W4bk87E=                    ) ; key id = 31373
             DNSKEY  257 3 5 (
                     N/DrX56+g1a4sx4ekH               ) ; key id = 15439
             RRSIG   DNSKEY 5 2 3600 20140611234001 ( 20140512224001 15439 supportbycall.org.
                           OUHmzQTlZuaYqvg4yQbVJClxD
             NSEC    bla.supportbycall.org. A NS SOA DS RRSIG NSEC DNSKEY
bla          A       192.168.5.5
             RRSIG   A 5 3 4600 20140611224239 (
                     20140512224001 31373 supportbycall.org. 4I1S4b5GRgU3xH7O4
```

DNSSEC testen und debuggen

- Prima DNS und DNSSEC-Validator: http://dnsviz.net





→ Ebenso: http://dnssec-debugger.verisignlabs.com/

DNSSEC mit dig abfragen

```
; <<>> DiG 9.7.3 <<>> bla.supportbycall.org +dnssec
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 48617
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 3, ADDITIONAL: 5

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;bla.supportbycall.org.          IN      A

;; ANSWER SECTION:
bla.supportbycall.org.  4600    IN      A       192.168.5.5
bla.supportbycall.org.  4600    IN      RRSIG   A 5 3 4600 20140611224239 20140512224001 31373
supportbycall.org. K5JgILUrc2XctYjkjH4I1S4b5GRgU3xH7O4PY4k9yfX9qjdSz8rnhg+s glrc3U/=
```

- → DNSSEC-Antworten haben DO-Flag gesetzt
- → Kaputtes DNSSEC führt zu NXDOMAIN

Die DNS-Flags im Überblick

- aa - authoritative answer
  - Daten kommen direkt vom authoritativen NS => Daten vertrauenswürdig
- ad - authenticated data
  - Recursive Nameserver hatte Trust Anchor => Daten vertrauenswürdig
- do - dnssec ok
  - Antwort ist valide lt. DNSSEC.

- rd - recursion desired
  - Client bittet den Server um rekursive Auflösung
- ra - recursion available
  - Server würde Client rekursive Auflösung ermöglichen

Das Problem von Keys und TTL

Der Key-Rollover

- Werden Keys kompromittiert, müssen Sie ausgetauscht werden
  - Webserver: Zertifikatstausch, Server-Restart, Fertig!
  - Nameserver: Records mit alten Keys/Signaturen werden bis zu einer Woche lang gecacht
- Ein falscher Key-Austausch sorgt für ungültige Signaturen
  - Ungültige Signaturen = Ungültige DNS-Records
  - Ungültige DNS-Records = keine DNS-Records = Downtime
- Und eine Woche Downtime kann sehr lang sein!
  - Auch 1 Tag oder auch nur 5 Minuten Downtime sind inakzeptabel

So werden Keys getauscht

- Rechtzeitig vor Ablauf alten Keys werden parallel bereits neue Keys veröffentlicht
  - Rechtzeitig = TTL-Zeit vor Ablauf!
- Nach dem Key-Wechsel müssen die alten Keys noch aktiv bleiben
  - Gecachte Records müssen mit alten Keys noch validiert werden können
  - Alte Keys also über die TTL-Zeit behalten
- Der Administrator muß permanent an den Key-Rollover denken!
  - Empfohlener Zyklus: 1 x im Monat!
  - TTL davor und danach Keys tauschen = fortlaufend zu tun haben

Wir brauchen eine fortlaufende Rotation

- Neue Keys müssen mindestens <TTL> vorher in die Zone aufgenommen werden, damit sie sicher bekannt sind.
- Erst dann darf mit ihnen signiert werden.
- Anschließend müssen Sie mindestens <TTL> in der Zone verbleiben, falls noch gecachte signierte Records kursieren.

Automatisches Key-Rollover mit Bind

- Werden die Zonen in Bind dynamisch verwaltet, kann Bind neue Schlüssel automatisch reinnehmen, aktivieren, deaktivieren, rausnehmen.
- Die Schlüssel-Dateien haben dazu Meta-Daten mit Timestamps zu den jeweiligen Events
- „dnssec-settime“ setzt diese Metadaten in den Schlüssel-Dateien
- Bind liest diese Timestamps ein und agiert entsprechend.

Zeit-Verwaltung mit dnssec-settime

```
flash:~/DNSSEC # cat Ktest.+007+28022.key
; This is a zone-signing key, keyid 28022, for test.
; Created: 20140508170312 (Thu May  8 19:03:12 2014)
; Publish: 20140515170515 (Thu May 15 19:05:15 2014)
; Activate: 20140508170312 (Thu May  8 19:03:12 2014)
; Inactive: 20140518170559 (Sun May 18 19:05:59 2014)
test. IN DNSKEY 256 3 7 AwEAAbcyWv2cweB2DVMc45gYF2suDCqJTU/kPwvxzyh7hd8DFLdrsBbU
ADMJwa31FExo01U3JCLOa38kUMOF40DQ01UiSID60t6sua9Mk2ECRuLG
IgQ4YpXT3hQ69Tp5IBU3hMByox1QVOBVBTCE60ZqvyVyO48zn1tylZ0V qh3Ym4KF

flash:~/DNSSEC # dnssec-settime -I+10d  Ktest.+007+28022.private
./Ktest.+007+28022.key
./Ktest.+007+28022.private

flash:~/DNSSEC # cat Ktest.+007+28022.key
; This is a zone-signing key, keyid 28022, for test.
; Created: 20140508170312 (Thu May  8 19:03:12 2014)
; Publish: 20140515170515 (Thu May 15 19:05:15 2014)
; Activate: 20140508170312 (Thu May  8 19:03:12 2014)
; Inactive: 20140522230736 (Fri May 23 01:07:36 2014)
test. IN DNSKEY 256 3 7 AwEAAbcyWv2cweB2DVMc45gYF2suDCqJTU/kPwvxzyh7hd8DFLdrsBbU
ADMJwa31FExo01U3JCLOa38kUMOF40DQ01UiSID60t6sua9Mk2ECRuLG
IgQ4YpXT3hQ69Tp5IBU3hMByox1QVOBVBTCE60ZqvyVyO48zn1tylZ0V qh3Ym4KF
```

NSSEC und NSEC3

- Vorhandene Records können nicht ausgetauscht werden - gut.
- Aber was ist, wenn Abfragen und damit Records unterdrückt werden?
- Wie beweist man, daß es etwas NICHT gibt?

- NSEC-Records enthalten eine Liste vorhandener Records und den alphabetisch darauffolgenden Hostnamen
  - Der letzte Record verweist wieder auf den alphabetisch Ersten

```
supportbycall.org.          NSEC    bla.supportbycall.org. A NS SOA DS RRSIG NSEC DNSKEY
                            RRSIG   NSEC 5 2 86401 20140611224239 (
bla.supportbycall.org.      NSEC    huhu.supportbycall.org. A RRSIG NSEC
                            RRSIG   NSEC 5 3 86401 20140612084024 (
huhu.supportbycall.org.     NSEC    supportbycall.org. MX RRSIG NSEC
                            RRSIG   NSEC 5 3 86401 20140612084024 (
```

- Gegenproblem: Jeder kann sehen, was es gibt :-(
  - „Zonewalking“

NSEC3PARAM

- NSEC3 hasht die Hostnamen
- Der Ressource Record NSEC3PARAM definiert die benutzten Hash-Verfahren und den verwendeten SALT.

```
;                                       <---  SALT --->
supportbycall.org.      NSEC3PARAM 1   0 100 0123456789ABCDEF
```

  - Hash Algorithm: The cryptographic hash algorithm used.
    - 0 is Reserved.
    - 1 is SHA-1.
    - 2-255 Available for assignment.
  - Flags: "Opt-out" (indicates if delegations are signed or not).
  - Iterations: How many times the hash algorithm is applied.
  - Salt: Salt value for the hash calculation.

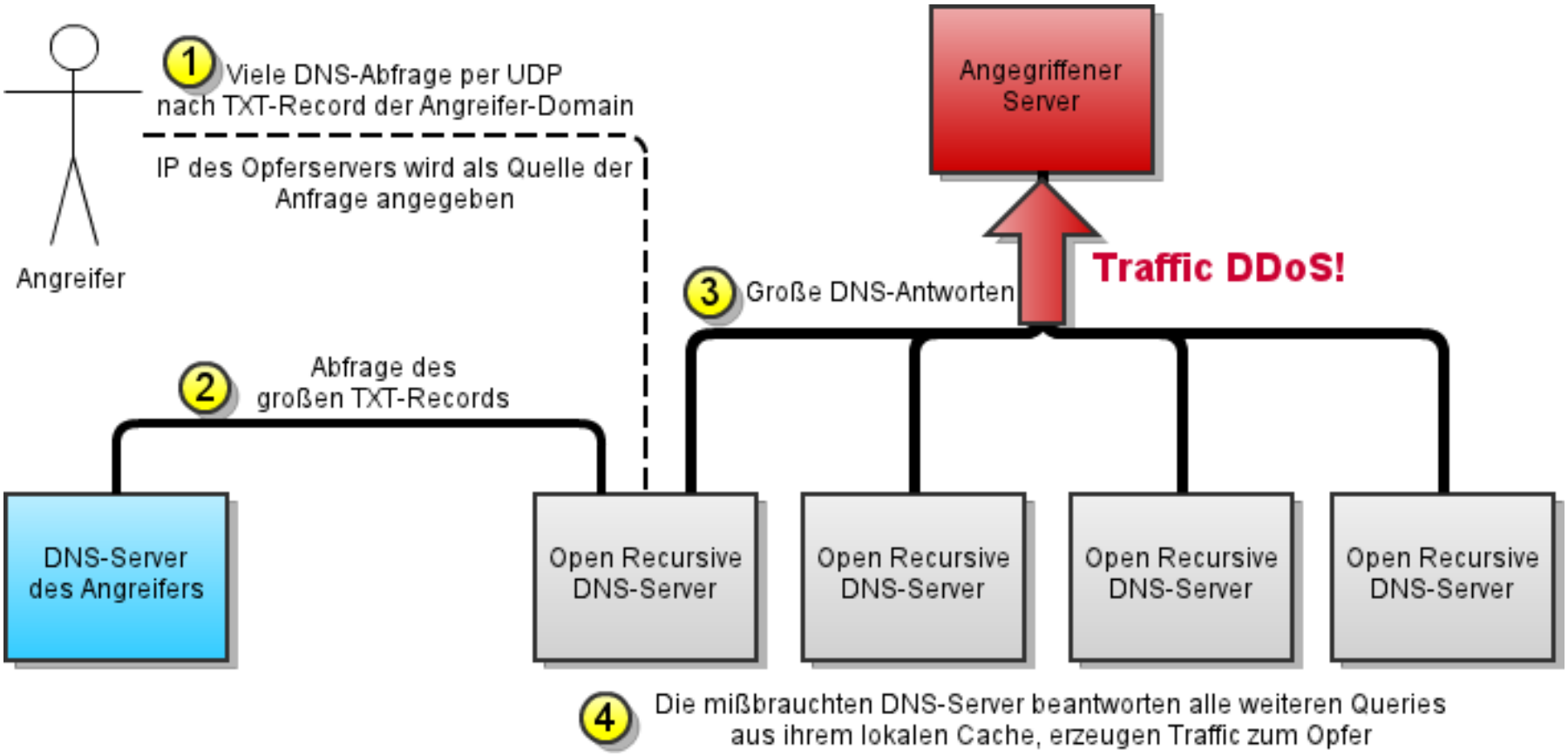
Zusammenfassung

Zusammenfassung: Die DNSSEC-Records im Überblick

- DNSSEC führt einige neue Record-Typen ein:
  - DNSKEY:
    - Public-Key + Gültigkeiten
  - RRSIG: Resource Record Signature
    - Signatur für einen oder mehrere Records
  - DS: Delegation Signer
    - Delegation von DNSSec für Subzonen
  - NSEC: Next Secure
    - Liste aller Records einer Zone (beweist, daß es etwas nicht gibt!)
  - NSEC3: Next Secure V3
    - Wie NSEC, aber gehashte Hostnamen
  - NSEC3PARAM:
    - Technische Parameter zu den Hashverfahren

Bonustrack:
DNSSEC und Security

- Im Sommer 2013 veranlasste ein Jugendlicher einen 300/400 Gbit/sec DDOS auf Spamhaus
- Genutzt wurde eine DNS-Amplitudenattacke
  - Kleines UDP-Paket mit gespooftem Absender
  - Erzeugt große Datenmenge als Antwort


Angegriffener Server
1
Viele DNS-Abfrage per UDP nach TXT-Record der Angreifer-Domain
IP des Opferservers wird als Quelle der Anfrage angegeben
Angreifer
3
Große DNS-Antworten
Traffic DDoS!
2
Abfrage des großen TXT-Records
DNS-Server des Angreifers
Open Recursive DNS-Server
Open Recursive DNS-Server
Open Recursive DNS-Server
Open Recursive DNS-Server
4
Die mißbrauchten DNS-Server beantworten alle weiteren Queries aus ihrem lokalen Cache, erzeugen Traffic zum Opfer

- DNS-Amplituden-Attacken erzeugen besonders große Datenmengen als Antwort. Laut Cloudflare:

- DNS-Amplituden-Attacke: Faktor 50
- NTP-Amplituden-Attacke: Faktor 200
- SNMP-Amplituden-Attacke: Faktor 500
- ???

- DNSSEC: Minimale Anfrage, große Datenmenge, über UDP

- Darum: Open Recursive Namesever vermeiden
  - => Analog zum Open Relay eines Mailservers: Client-IPs von extern dürfen nur die eigenen authoritatiuven Zonen abfragen

```
options {
  [...]
        allow-recursion { x.x.x.x/x, 10.0.0.0/8; 192.168.0.0/16;
127.0.0.0/8; ::1; };
  [...]
}
```

- Natürlich und gerne stehe ich Ihnen jederzeit mit Rat und Tat zur Verfügung und freue mich auf neue Kontakte.

Peer Heinlein

Mail: p.heinlein@heinlein-support.de

Telefon: 030/40 50 51 - 42

- Wenn's brennt:
  - Heinlein Support 24/7 Notfall-Hotline: 030/40 505 - 110

Ja, diese Folien stehen auch als PDF im Netz...
http://www.heinlein-support.de/vortrag

Soweit, so gut.

Gleich sind Sie am Zug:
Fragen und Diskussionen!

Wir suchen neue Kollegen für:

Helpdesk, Administration, Consultanting!

Wir bieten:
Spannende Projekte, Kundenlob, eigenständige Arbeit, keine Überstunden, Teamarbeit

...und natürlich: Linux, Linux, Linux...

http://www.heinlein-support.de/jobs

Und nun...

- Vielen Dank für's Zuhören...
- Schönen Tag noch...
- Und viel Erfolg an der Tastatur...

Bis bald.

heinlein

Heinlein Support hilft bei allen Fragen rund um Linux-Server

HEINLEIN AKADEMIE

Von Profis für Profis: Wir vermitteln in Training und Schulung die oberen 10% Wissen: geballtes Wissen und umfangreiche Praxiserfahrung.

HEINLEIN HOSTING

Individuelles Business-Hosting mit perfekter Maintenance durch unsere Profis. Sicherheit und Verfügbarkeit stehen an erster Stelle.

HEINLEIN CONSULTING

Das Backup für Ihre Linux-Administration: LPIC-2-Profis lösen im CompetenceCall Notfälle, auch in SLAs mit 24/7-Verfügbarkeit.

HEINLEIN ELEMENTS

Hard- und Software-Appliances für Archivierung, IMAP und Anti-Spam und speziell für den Serverbetrieb konzipierte Software rund ums Thema E-Mail.