

D N S S E C

HOWTO

Secure the Domain Name System

Meta Rhein Main Chaos Days
28. Mai 2005

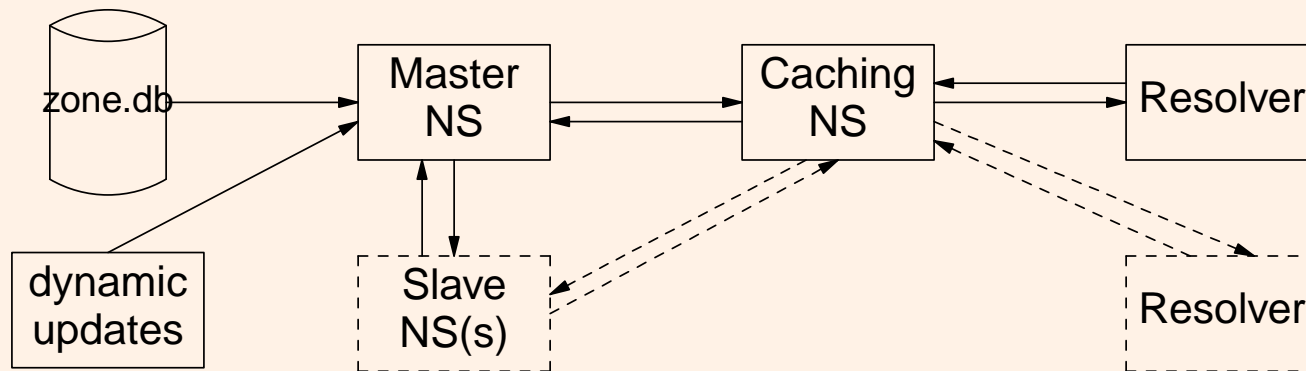
Holger.Zuleger@hznet.de

Agenda

- Voraussetzung: Grundlagen in Kryptographie, DNS, BIND, no djv :-(
- Einführung
 - Warum DNSsec ?
 - Was ist DNSsec ?
- Authentisierte Zonen Transfers
- Signierte Zonen
 - Schlüsselerzeugung
 - Signieren
 - Secure Resolver
- Secure Entry Points: Insellösung oder Hierarchien
- DNSsec und Privacy: NSEC und die Folgen
- DNSsec Tools

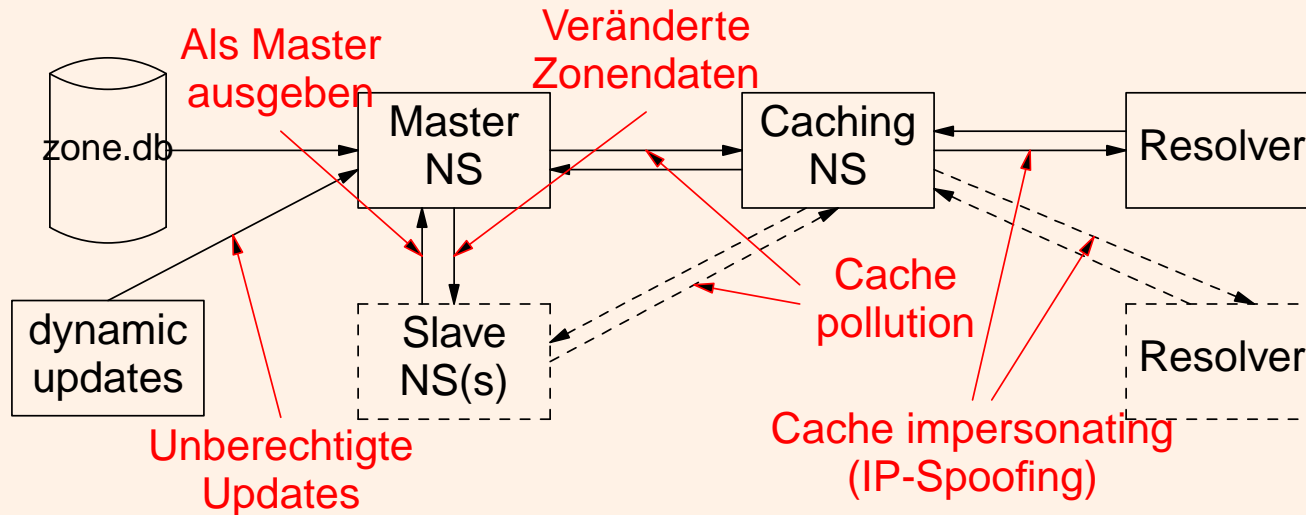
Warum DNSsec ?

- DNS (RFC1034, RFC1035) ist unsicher designed! ('86, erste Angriffe '90)



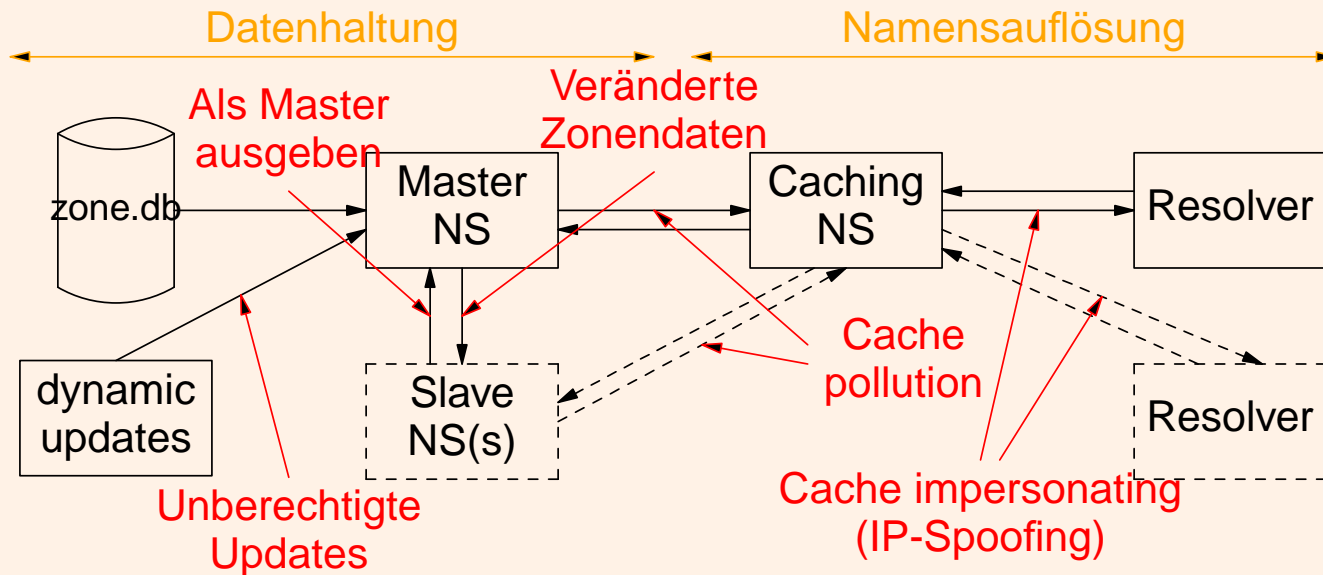
Warum DNSsec ?

- DNS (RFC1034, RFC1035) ist unsicher designed! ('86, erste Angriffe '90)



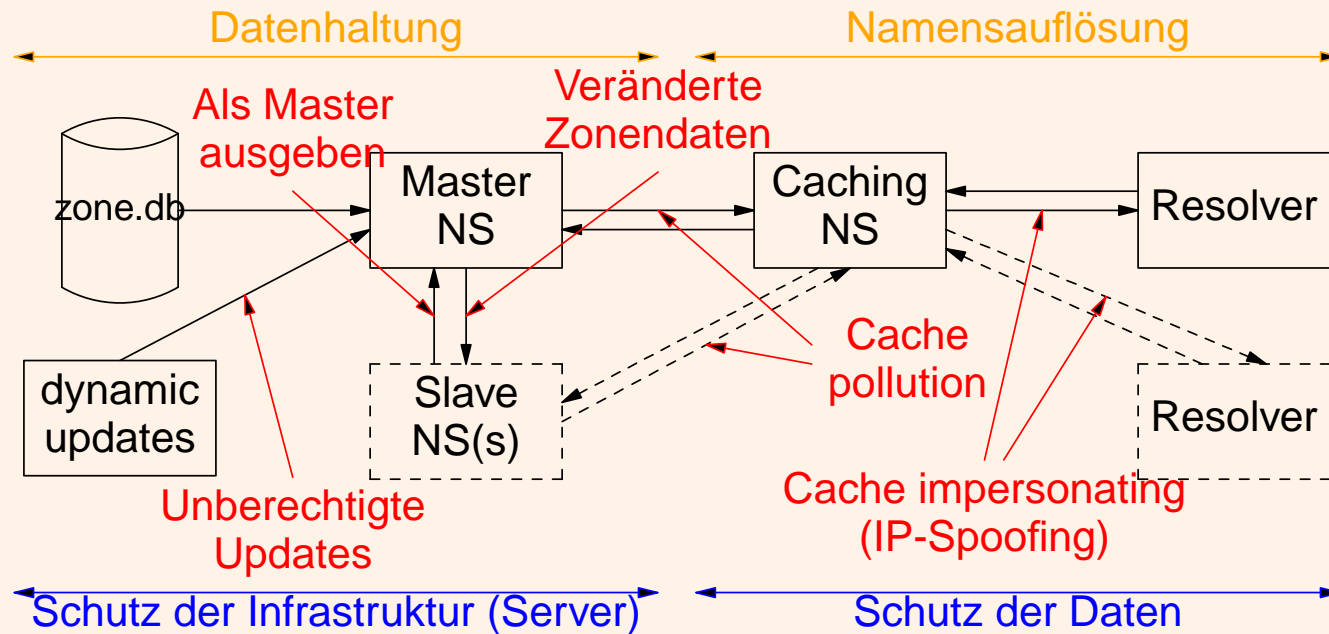
Warum DNSsec ?

- DNS (RFC1034, RFC1035) ist unsicher designed! ('86, erste Angriffe '90)



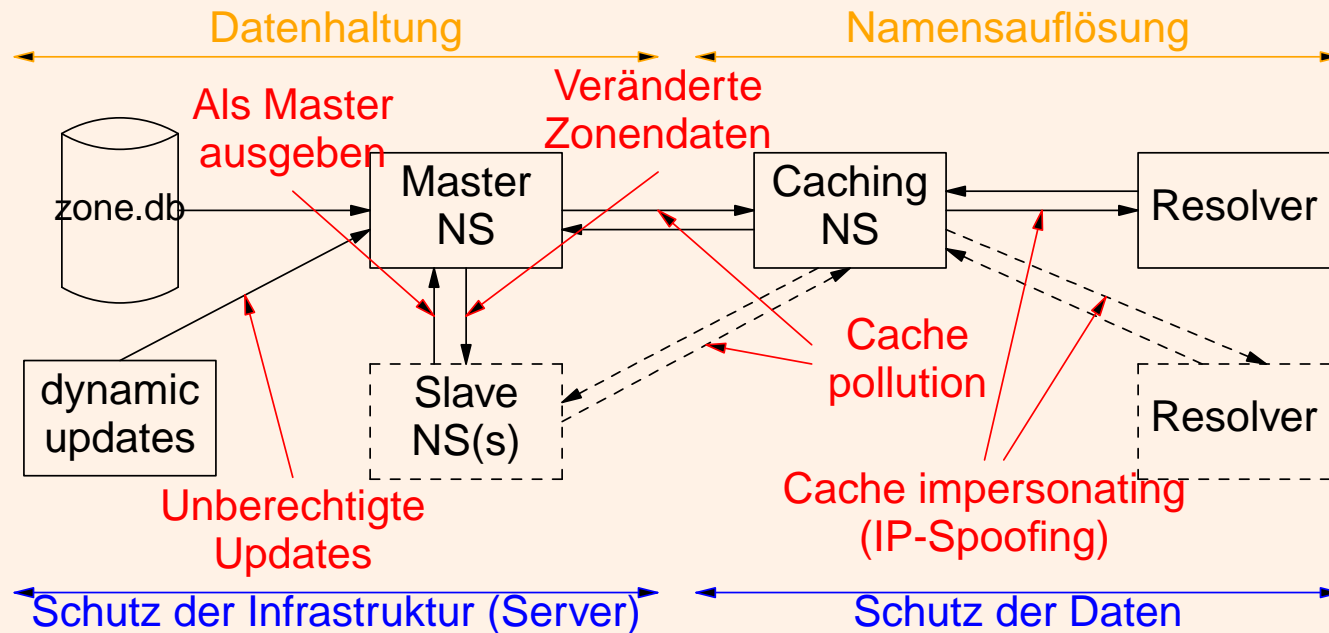
Warum DNSsec ?

- DNS (RFC1034, RFC1035) ist unsicher designed! ('86, erste Angriffe '90)



Warum DNSsec ?

- DNS (RFC1034, RFC1035) ist unsicher designed! ('86, erste Angriffe '90)



- Schutz der (Server) Infrastruktur
Authentisierung über symmetrische- oder asymmetrische Keys (Public-Keys)
- Schutz der Daten (Resource Records)
Signierte Datensätze (Public-Keys)

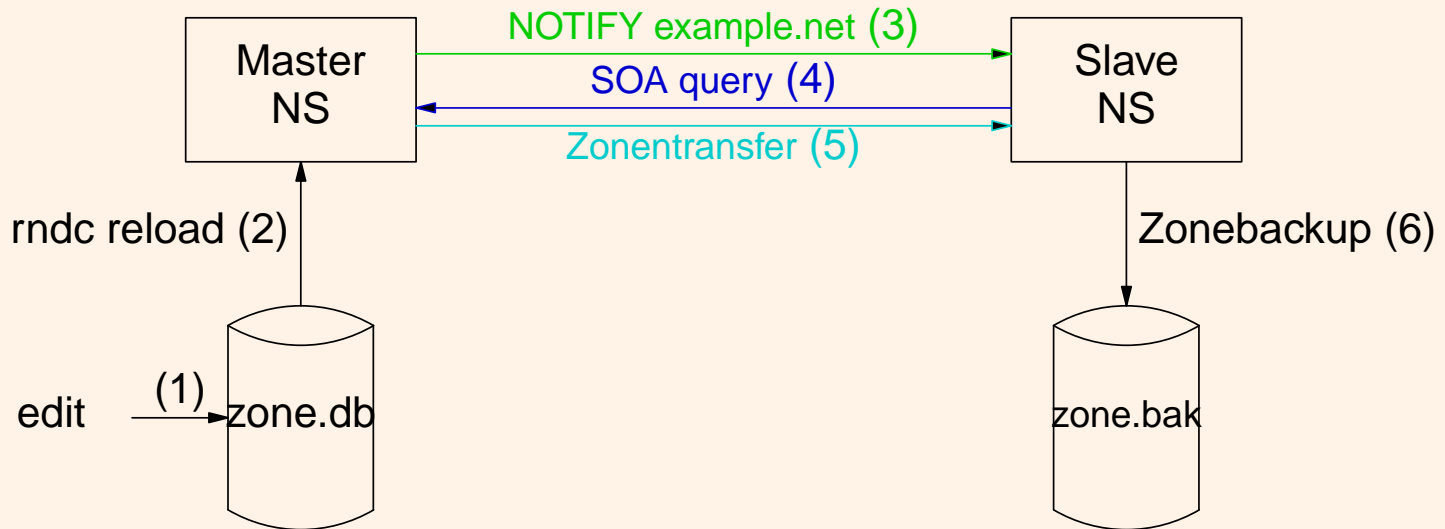
Was ist DNSsec ?

- Beginn der Entwicklung 1995; RFC2535 veröffentlicht 1999
- Seit März 2005 überarbeitete Version
RFC4033, RFC4034, RFC4035 und RFC3658 (DS-Record)
- Secure DNS adressiert verschiedene Bereiche:
 - a. Authentisierte Zonentransfers
 - b. Sichere dynamische Updates (RFC3007)
Interessantes Thema... aber nicht jetzt!
 - c. Signierte Zonen
Kryptographische Signatur über die Zoneninhalte
 - d. Authentisierte Querys
TSIG zwischen Stub-Resolver und Caching DNS
- Implementierung: bind-9.3.x, NSD 2.1.x
- Bedeutung nimmt zu / Mehr Dienste im DNS
(Anti SPAM (MARID), ENUM, SSH-Fingerprints, SRV-Records)

DNSsec – Resource Records

- TSIG (RFC2848) Pseudo-RR
Transaktions Signaturen (Hashed MD5)
Authentisierte Zonentransfers sowie sig. Querys und Updates
- TKEY (RFC2930) Pseudo-RR
Verfahren zur Aushandlung von symmetrischen Keys für TSIG
(Diffie-Hellman, Sig(0), GSSAPI)
- SIG(0) (RFC2931) Pseudo-RR
Transaktions Signaturen (Public-Key: RSA-MD5, RSA-SHA1, DSA)
Authentisierte dynamische Updates und TKEY Request
- RRSIG, DNSKEY, NSEC (RFC4034)
Ehemals: SIG, KEY, NXT (RFC2535, RFC3845)
Signierte Resource Records (Public Key Verfahren)
- DS (RFC3658)
Delegation der Vertrauensbeziehung

Zonentransfer



Angriffsszenarien:

- Verkörperung des Master (IP-Spoofing)
- Einschleusen falscher Daten beim Zonentransfer (man in the middle)

Transaktions Signaturen (TSIG)

- Shared Secret (HMAC-MD5) zwischen zwei(!) Hosts.
- Zur Zeit primär zwei Anwendungen:
 - Authentisierter Zonentransfer zwischen Master und Slave.
 - Dynamische Updates (z.B. DHCP-Server zu Master NS)
- Keine Verschlüsselung!
- Öffentliche Daten!
- Authentisierung und Integritäts-Check.
 - Bist du der Master Server?
 - Sind die Daten beim Transport modifiziert worden?
 - Sind die Daten aktuell?
- Voraussetzung: Synchronisierte Uhren (max. diff. 5 Min)!

TSIG Konfiguration

5 Schritte zu sicheren Zonentransfers:

- Uhren synchronisieren (ntp)
- Generieren eines Shared Keys
 - Als Algorithmus ist zur Zeit nur HMAC-MD5 möglich
 - Keylänge muß zwischen 1 und 512 liegen
 - Der Name wird per Konvention aus den beiden FQDNs gebildet

```
$ dnssec-keygen -a HMAC-MD5 \  
                -b 128 -n HOST \  
                ns1.example.net-ns2.example.net  
Kns1.example.net-ns2.example.net.+157+19512
```

Output: Zwei Dateien (leider)!

```
Kns1.example.net-ns2.example.net.+157+19512.key  
Kns1.example.net-ns2.example.net.+157+19512.private
```

TSIG Konfiguration (2)

- Definition des Key in `named.conf` (Master und Slave)

```
key "ns1.example.net-ns2.example.net." {  
    algorithm hmac-md5;  
    secret "/EB/vJJkokIoSGD6Wjmyeg==";  
};
```

Das Secret holt man aus der `*.private` Datei

```
$ grep Key: Kns1.example.net-ns2.example.net.+157+19512.private  
Key: /EB/vJJkokIoSGD6Wjmyeg==
```

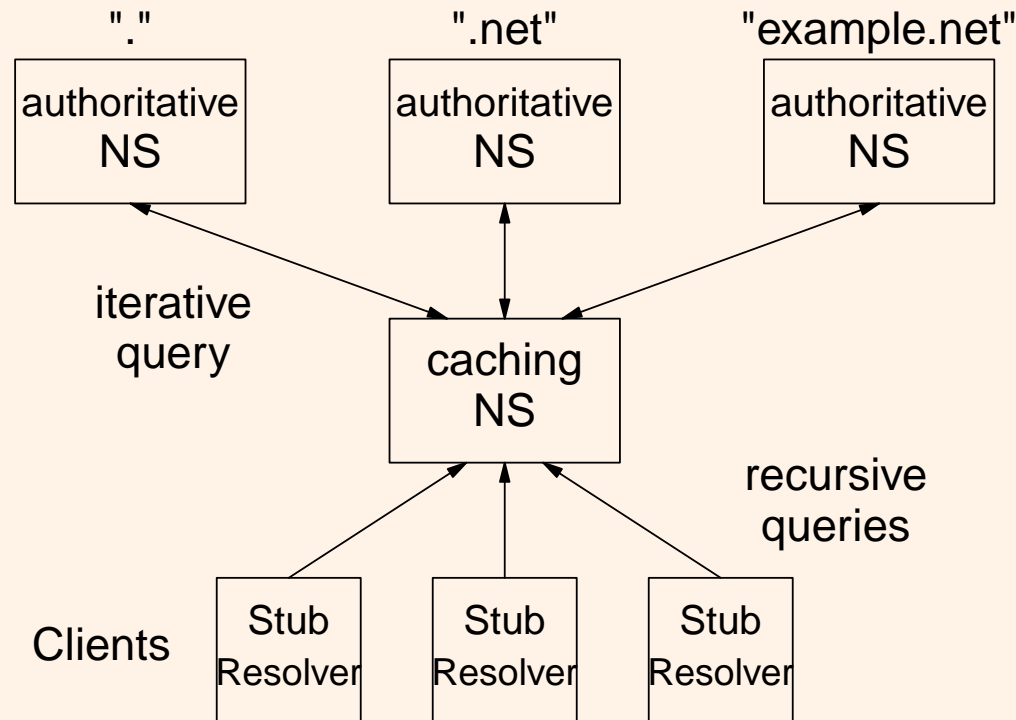
- Binding: Host to Key (mind. beim Slave)

```
server 1.2.3.4 {  
    keys { key ns1.example.net-ns2.example.net.; };  
};
```

- Access Control (Optional):

```
zone "example.net" IN {  
    ...  
    allow-transfer { key ns1.example.net-ns2.example.net.; }; # Master  
    allow-transfer { none; }; # Slave  
};
```

Authentisierte Namensauflösung



- Host zu Host Authentisierung ist nicht praktikabel
- Stattdessen: Signatur des Zoneninhalts (authenticated data origin)

Signierte Zonen

Authoritative NS (Master)

- Generieren eines asymmetrischen Keys (Zonenkey)
- Einfügen des (öffentlichen) Keys in die Zone
- Erhöhen der Seriennummer im SOA Record
- Signieren der Zone (Offline)
- Reload der Zone

Resolving NS

- DNSsec einschalten
- Öffentlicher Zonenkey als „Secure Entry Point“ (SEP) hinterlegen

DNSKEY – Schlüssel zu signierten Zonen

Kommando `dnssec-keygen` mit den folgenden Parametern:

- Schlüsselalgorithmus und Keylänge
 - DSA (Keysize 512 bis 1024 Bit)
 - RSAMD5 (Keysize 512 bis 4096 Bit)
 - RSASHA1 (Keysize 512 bis 4096 Bit)
- Namenstyp: ZONE
- **Schlüsselname** = Domainname

```
$ dnssec-keygen -a RSASHA1 -b 512 -n ZONE sec.example.net
Ksec.example.net.+005+62759
      ^           ^
Algorithmus -----+ |
Key ID -----+-----+
```

Zwei Dateien:

```
-rw-r--r--  1 hoz hoz   125 Aug 07 12:31 Ksec.example.net.+005+62759.key
-rw-----  1 hoz hoz   549 Aug 07 12:31 Ksec.example.net.+005+62759.private
```

Einfügen des Keys in die Zone

- Der öffentliche Teil des Keys steht als RR in der Datei K*.key:

```
sec.example.net. IN DNSKEY 256 3 5 AQPUSMEKBKBSYO/xd...
                        ^   ^  ^  ^
Flags: Bit8 == Zonenkey  --+   |   |   |
Protokoll (3 == DNS)  -----+   |   |
Algorithmus  -----+   |   |
Schlüsselmaterial (gekürzt) -----+
```

- Dateiname des Keys ändert sich bei Neugenerierung

```
$ cat Ksec.example.net.+00*.key > keys.db
```

- Einfügen der Keys in die Zone (\$INCLUDE Anweisung)

```
$ cat zone.db
@ 7200 IN SOA ns1.example.net. hostmaster.example.net. ....
      IN NS      ns1.example.net.
      IN NS      ns2.example.net.
$INCLUDE keys.db
....
```

- Erhöhen der Seriennummer !

RRSIG – Unterschriebene Resource Records

- Signieren der Zone durch `dnssec-signzone`

```
$ dnssec-signzone -o sec.example.net zone.db
zone.db.signed
```

- Sortieren der RR-Sets
- Einfügen der NSEC Records
- Signieren jedes RR-Sets und Einfügen des Signatur Records (RRSIG)

```
$ cat zone.db.signed
```

```
...
sec.example.net. 7200 IN NS      ns1.example.net.
                  7200 IN NS      ns2.example.net.
                  7200 IN RRSIG NS 1 2 7200 (
Sig. Lifetime           20040906100802 20040807100802
Keytag+Name             62759 sec.example.net.
Signaturdaten         AK9adL3Ov7VkVLYoan/5CHUO...== )
```

- **Wichtig:** Vor Ablauf der Signatur erneut signieren!

Reload der Zone

- DNSsec im Authoritative NS einschalten:

```
options {  
    recursion no;  
    dnssec-enable yes;  
};
```

- Name des Zonenfiles in der `named.conf` eintragen:

```
zone "sec.example.net" in {  
    type master;  
    file "sec.example.net./zone.db.signed";  
    allow-transfer { key ns1.example.net-ns2.example.net.; };  
    notify yes;  
};
```

- Zone neu laden

```
$ rndc reload sec.example.net
```

- Meldungen kontrollieren

```
$ tail -f /var/log/named  
07-Aug-2004 13:38:43.198 general: info: zone sec.example.net/IN: \  
                                loaded serial 12 (signed)
```

Secure Resolver (Caching NS)

- BIND 9.3.x benutzen
- Secure DNS in `named.conf` einschalten

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

- Secure Entry Point in der „trusted-keys“-Section hinterlegen

```
trusted-keys {
    "sec.example.net." 256 3 5 "AQPUSMEKKBKBSYO/xdnL/j..."
};
```

- Wie ?

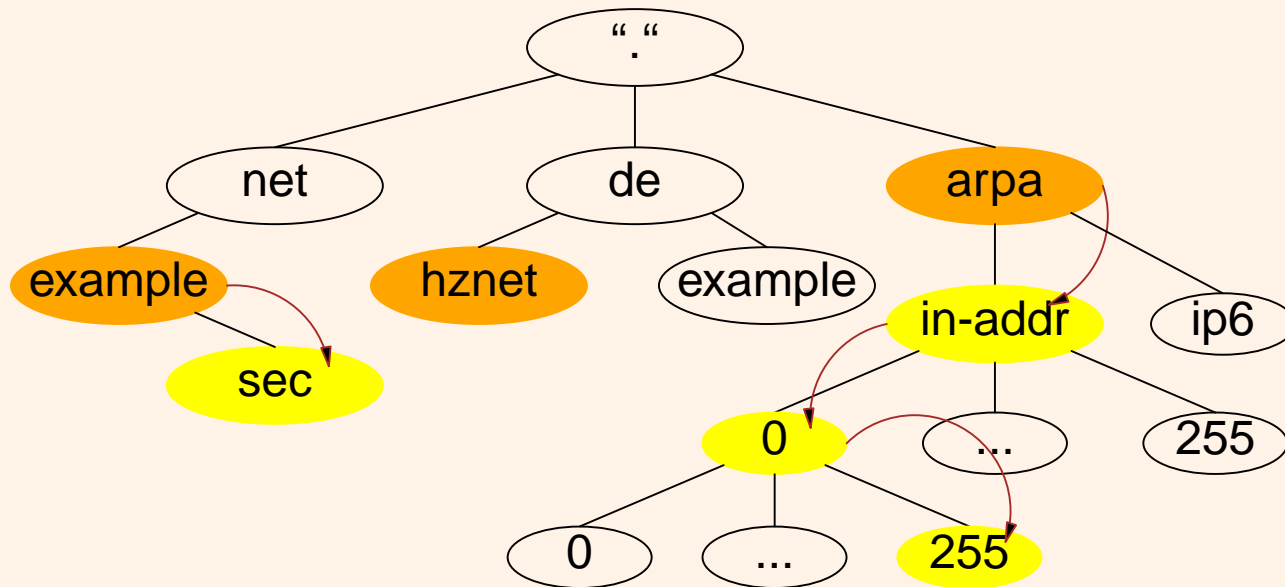
```
$ { echo "trusted-keys {";
    cat Ksec.example.net.+005+*.key |
    sed -n 's/^\([a-zA-Z]*\) IN DNSKEY \([0-9]*\) \([0-9]\) \
        \([0-9]\) \(.*)/"\1\" \2 \3 \4 "\5"/p'
    echo "};"
} >> named.conf
```

Oder besser: <http://www.hznet.de/dns/create-trusted-key-section>

What next?

- „Secure Entry Points“ müssen auf sicherem Wege übermittelt werden!
Wie? Skalierbarkeit?
- Jeder (secure) Resolver benötigt den öffentlichen Zonenkey von **allen** signierten Zonen!
Lösung: Delegation Signer d.h. Aufbau einer Hierarchie ?
oder: Delegated Verification ? (Sichere Inseln)
- Ändert sich der Zone Signing Key müssen alle Resolver angepasst werden. Lösung: Key Signing Keys (KSK)
- Wie kann eine negative Antwort signiert werden ?
Lösung: NSEC Resource Records ?
oder: Online signing ?
oder: NSEC3 ??
- Signatur von Wildcard Records

Chain of Trust / Secure Entry Points



- Chain of Trust durch **DS Records**
- Wenige(r) „**Secure Entry Points**“
- Idealfall: Nur SEP der Rootzone notwendig
- Minimaler Schlüsselaustausch durch Key Signing Keys

KSK + ZSK

- Key Signing Keys (SEP)
 - Wird lediglich zum Signieren der Zonenkeys verwendet
 - Wenig genutzt (kleine Menge zu signierender Daten)
 - Große Schlüssellänge (DSA 1024, RSA 2048)
 - Lange Lebensdauer, d.h. selten geändert (5 bis 10 Jahre)
 - Änderung des KSK muß kommuniziert werden!
 - KSK über ein Bit im Flagfeld gekennzeichnet (RFC3757)
- Zone Signing Keys
 - Wird zum Signieren der Zonendaten verwendet
 - Häufig genutzt (große Menge zu signierender Daten)
 - Kleine Schlüssellänge (RSA 512 Bit)
 - Kurze Lebensdauer (Wochen bzw. Monate)
 - Änderung des Schlüssels muß nicht kommuniziert werden

KSK + ZSK (Konfiguration)

- Generieren des KSK (Option -f KSK)

```
$ dnssec-keygen -f KSK -n ZONE -a DSA -b 1024 sec.example.net  
Ksec.example.net.+003+16004
```

- Generieren eines zweiten ZSK

```
$ dnssec-keygen -n ZONE -a RSASHA1 -b 512 sec.example.net  
Ksec.example.net.+005+57764
```

- Einfügen der Keys in die Zone

```
$ cat Ksec.example.net.00[135]+*.key > keys.db
```

- Erhöhen der Seriennummer!

- Signieren + Neuladen

```
$ dnssec-signzone -o sec.example.net zone.db  
zone.db.signed  
$ rndc reload sec.example.net
```

- KSK in der trusted-key Section des Resolver eintragen!

DS – Delegation Signer

- Delegation: Einfügen eines Verweises auf den KSK in der Parentzone
dnssec-signzone kreiert dsset- und keyset-Datei

- Die `keyset`-Datei enthält die DNSKEY-RR der Key Signing Keys
Diese werden in der secure Zone (!) eingefügt

```
$ cat keyset-sec.example.net.
sec.example.net. 7200 IN DNSKEY 257 3 3 (
                        62uVBWg9spPDjXVaaXNaEwjLlNaKEqfwz4+A...
                        ) ; key id = 16004
```

- Die `dsset`-Datei enthält die DS-RRs als Verweis auf die KSKs
Die DS-Records werden in der Parent-Zone (!) eingefügt.

```
$ cat dsset-sec.example.net.
sec.example.net. IN DS 16004 3 1 55FBEE63...
Key Tag -----^ ^ ^ ^
Algorithm Number -----+ | |
Digest Type (SHA1) -----+ +-- Hash des DNSKEY
```

- Parent kann aus der `keyset`-Datei den DS-Record generieren

DS – Secure the Parent

- Der Parent muß seine Zone signieren!
- Wir brauchen Schlüsselmaterial für den Parent (KSK, ZSK, usw.)
- Signieren der Parent Zone

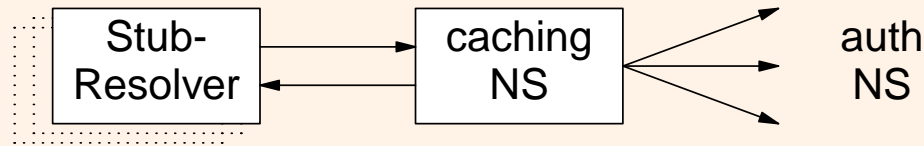
```
$ dnssec-signzone -g -o example.net zone.db  
zone.db.signed
```

- Das Ergebnis:

```
$ORIGIN example.net.  
sec      7200    IN NS    ns1.example.net.  
         7200    IN NS    ns2.example.net.  
         7200    IN DS    16004 3 1 (55FBEE63... )  
         7200    IN RRSIG DS 1 3 7200 20040906133208 (  
         20040807133208 65516 example.net.  
         dCzVu1NC7s/EB8e7Ynsl.... )
```

- Der Parent signiert nicht die Delegation (NS-Records)
Lediglich der DS Record wird durch den Parent signiert!
- Resolver benötigt den SEP des Parent in der trusted-key Section

Stub-Resolver / Clients



Zwei Modi:

- a. Signaturprüfung durch den Caching NS (Resolver)
 - EDNS0: do-Flag in der Anfrage setzen
 - EDNS0: UDP-Size 4096
 - In der Antwort sollte AD-Bit gesetzt sein (verified secure/insecure)

- b. Eigenprüfung der Signatur durch den Stub-Resolver
 - Stub-Resolver benötigt Trust-Anchor (SEP)
 - Zusätzlich bei der Anfrage das CD-Flag setzen
 - Die Antwort enthält auch die Authority Section
 - AD-Bit nicht gesetzt

Stub-Resolver (dig)

```
$ dig @secResolver +multiline +dnssec a.sec.example.net
; <>> DiG 9.3.0rc3 <>> @secResolver +multiline +dnssec a.sec.example.net
;; global options: printcmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 42021
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 4, AUTHORITY: 5, ADDITIONAL: 11

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;a.sec.example.net.                IN A

;; ANSWER SECTION:
a.sec.example.net.                5147 IN A 1.2.3.4
a.sec.example.net.                5147 IN RRSIG A 1 4 7200 20040906133347 (
                                20040807133347 10809 sec.example.net.
                                EZ0P5FVLaaRYx09Gh5VWVJzySt9CTPDhRgwAE522+L93
                                27XecpZQwsKileKFdoExPQqQAWJJo4c9vUIZ+3tSBw== )
a.sec.example.net.                5147 IN RRSIG A 1 4 7200 20040906133347 (
                                20040807133347 32998 sec.example.net.
                                Ju5aWfSFGHpp1+spF4/PVmB6vOZ/LBJQJqjGF/Du/tyS
                                gNvUdsGkNn0EN2hxp8Z6FByTOKrV1w4SQZufBs0EVw== )

;; Query time: 107 msec
;; SERVER: 1.2.3.105#53(secResolver)
;; WHEN: Sat Aug 07 15:37:51 2004
;; MSG SIZE rcvd: 2458
```

NSEC ...

Wie kann eine negative Antwort (offline) signiert werden?

- NSEC (früher NXT) Record (Next SECure Record)
 - Alle Records sortieren
 - Jedes Label erhält NSEC als Zeiger auf nächstes Label
 - Signieren der Zone

- Beispiel (Ohne RRSIG)

```
example.net.      SOA  ns1.example.net.  ...
                  NS   ns1.example.net.
                  NS   ns2.example.net.
                  NSEC a.example.net. NS SOA RRSIG NSEC DNSKEY

a.example.net.   A    1.2.3.4
                  NSEC b.example.net. A RRSIG NSEC

b.example.net.   A    1.2.3.5
                  NSEC example.net. A RRSIG NSEC
```

- Funktioniert auch mit Wildcards

... und die Folgen

- Ermöglicht einfaches Auslesen aller Labels einer Zone (Zonewalk)

```
$ dig +noall +answer nsec example.net
example.net.      7171  IN NSEC      a.example.net.  A RRSIG NSEC

$ dig +noall +answer nsec a.example.net
a.example.net.   7171  IN NSEC      b.example.net.  A RRSIG NSEC

$ dig +noall +answer nsec b.example.net
b.example.net.   7171  IN NSEC      example.net.   A RRSIG NSEC
```

Siehe auch: DNSSEC Walker (<http://josefsson.org/walker/>)

- Alternativen zu NSEC werden zur Zeit noch diskutiert
 - Evaluating DNSSEC Transition Mechanisms
draft-ietf-dnsext-dnssec-trans-02.txt
 - DNSSEC Hash Authenticated Denial of Existence
draft-ietf-dnsext-dnssec-nsec3-01.txt
 - Minimally Covering NSEC Records and DNSSEC On-line Signing
draft-ietf-dnsext-dnssec-online-signing-00.txt

DNSsec Tools

- KROd – Key Rollover Daemon (www.idsa.prd.fr/index.php?page=kro&lang=en)
 - Automatischer ZSK Rollover
 - Automatischer KSK Rollover
inkl. KSK Schlüsselaustausch mit dem Parent
- DNSSEC Key Maintenance Tools (www.ripe.net/disi/code.html)
 - Verwaltung eines Key Storage
 - Halbautomatischer KSK und ZSK Rollover
- DNSsec Tools (www.dnssec-tools.org)
 - Zone signing and key management tool
- Zone Key Tool (www.hznet.de/zkt/)
 - Automatischer ZSK Rollover
 - Automatisches Resigning der Zone (inkl. SOA incr.)
 - Liest secure Zonen aus named.conf

Zone Key Tool (ZKT)

- Unterstützt Keymanagement und Zone Signing

```
$ dnssec-zkt
$ dnssec-signer -N /etc/named.conf
```

- Einfaches Konfigfile (Auszug aus `dnssec.conf`)

```
# zone specific timing values
ResignInterval: 3d      # (259200 seconds)
Sigvalidity:    30d     # (2592000 seconds)
Max_TTL:        6h      # (21600 seconds)
Propagation:    5m      # (300 seconds)

# signing key parameters
KSK_lifetime:    0
KSK_algo:        DSA    # (Algorithm ID 3)
KSK_bits:        1024
ZSK_lifetime:    10d    # (864000 seconds)
ZSK_algo:        RSAMD5 # (Algorithm ID 1)
ZSK_bits:        512
```

- Vollautomatischer ZSK Rollover
- Automatisches inkrementieren der Seriennummer im SOA-Record
Unterstützt sequentielle Seriennummer und YYYYmmDDxx Format

ZKT – Konfiguration

- Pro Zone ein Verzeichnis anlegen (Name = Zonename)

```
$ mkdir example.net.  
$ cd example.net.
```

- Anlegen der Zonendatei (zone.db)

```
$ head -15 zone.db  
$TTL      7200  
; Be sure that the serial number below is left  
; justified in a field of at least 10 spaces!!  
;           0123456789;  
@ IN SOA ns1.example.net. hostmaster.example.net. (  
           63           ; Serial  
           43200      ; Refresh  
           1800       ; Retry  
           2W        ; Expire  
           7200      ) ; Minimum  
  
           IN NS      ns1.example.net.  
           IN NS      ns2.example.net.  
  
$INCLUDE dnskey.db  
...
```

ZKT – Konfiguration(2)

- Anlegen einer (leeren) zone.db.signed Datei

```
$ touch zone.db.signed
$ ls -l
-rw-r----- 1 hoz      users    916 2005-05-26 14:42 zone.db
-rw-r--r--  1 hoz      users      0 2005-05-26 14:43 zone.db.signed
```

- Signieren der Zone

```
$ dnssec-signer -v -o example.net.
parsing zone "example.net." in dir "."
  No active KSK found: generate new one
  No active ZSK found: generate new one
  Resigning necessary: Modified keys
  Writing key file "./dnskey.db"
  Incrementing serial number (64) in file "./zone.db"
  Signing zone "example.net."

$ ls -l
-rw-r--r--  1 hoz      users    121 2005-05-26 14:45 Kexample.net.+001+16809.key
-rw-----  1 hoz      users    545 2005-05-26 14:45 Kexample.net.+001+16809.private
-rw-r--r--  1 hoz      users    581 2005-05-26 14:45 Kexample.net.+003+59110.key
-rw-----  1 hoz      users    688 2005-05-26 14:45 Kexample.net.+003+59110.private
-rw-r--r--  1 hoz      users   1136 2005-05-26 14:45 dnskey.db
-rw-r--r--  1 hoz      users     71 2005-05-26 14:45 dsset-example.net.
-rw-r--r--  1 hoz      users    702 2005-05-26 14:45 keyset-example.net.
-rw-r-----  1 hoz      users    916 2005-05-26 14:45 zone.db
-rw-r--r--  1 hoz      users   4080 2005-05-26 14:45 zone.db.signed
```

ZKT – Konfiguration(3)

- Keystatus anzeigen

```
$ dnssec-zkt -a .
```

Keyname	Tag	Typ	Sta	Algorit	Generation	Time	Age
example.net.	18710	KSK	act	DSA	May 26 2005	15:07:24	13m42s
example.net.	57705	ZSK	act	RSAMD5	May 26 2005	15:07:24	13m42s

- Zonefile in named.conf ändern

```
zone "example.net." in {  
    type master;  
    file "example.net./zone.db.signed";  
};
```

- Force resigning und reload der Zone auf den Nameserver

```
$ dnssec-signer -r -f -v -N named.conf  
parsing zone "example.net." in dir "./."  
Resigning necessary: Option -f  
Writing key file "././dnskey.db"  
Incrementing serial number (65) in file "././zone.db"  
Signing zone "example.net."  
Reload zone "example.net."
```

- Testen: dig is your friend

References

Miek Gieben, Internet Protocol Journal (Vol. 7, Issue 2, June 2004)
„DNSSEC: The Protocol, Deployment and a Bit of Development“

Nominum

BIND v9 Administrator Reference Manual

Olaf Kolkman, Ripe-NCC DISI

„DNSSEC Howto Version 1.3“

RFCs 1034, 1035, 2535, 2848, 2930, 2931, 3007, 3655, 3658, 3757,
3833, 3845

4033 (DNS Security Introduction and Requirements)

4034 (Resource Records for the DNS Security Extensions)

4035 (Protocol Modifications for the DNS Security Extensions)

Drafts DNSSEC Operational Practices

draft-ietf-dnsop-dnssec-operational-practices-04.txt

Links <http://www.dnssec.net>

<http://www.ietf.org/html.charters/dnsexext-charter.html>



Fragen ?

Fragen ?

<http://www.hznet.de/dns/dnssec-mrmcd050528.pdf>

Fragen ?

<http://www.hznet.de/dns/dnssec-mrmcd050528.pdf>

Herzlichen Dank für Eure Aufmerksamkeit

CONTENTS

.....	1	ZKT – Konfiguration(2)	30
Agenda	2	ZKT – Konfiguration(3)	31
Warum DNSsec ?	3	References	32
Was ist DNSsec ?	4	33
DNSsec – Resource Records	5		
Zonentransfer	6		
Transaktions Signaturen (TSIG)	7		
TSIG Konfiguration	8		
TSIG Konfiguration (2)	9		
Authentisierte Namensauflösung	10		
Signierte Zonen	11		
DNSKEY – Schlüssel zu signierten Zonen	12		
Einfügen des Keys in die Zone	13		
RRSIG – Unterschriebene Resource Records	14		
Reload der Zone	15		
Secure Resolver (Caching NS)	16		
What next?	17		
Chain of Trust / Secure Entry Points	18		
KSK + ZSK	19		
KSK + ZSK (Konfiguration)	20		
DS – Delegation Signer	21		
DS – Secure the Parent	22		
Stub-Resolver / Clients	23		
Stub-Resolver (dig)	24		
NSEC	25		
... und die Folgen	26		
DNSsec Tools	27		
Zone Key Tool (ZKT)	28		
ZKT – Konfiguration	29		